REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

IMAGE LEAF CLASSIFICATION FOR PLANT DISEASES DETECTION USING GREY WOLF OPTIMIZATION TECHNIQUE

Amenah Nazar jabbar JABBAR

Master`s Thesis

Supervisor

Asst. Prof. Dr. Hakan KOYUNCU

Istanbul, 2023

IMAGE LEAF CLASSIFICATION FOR PLANT DISEASES DETECTION USING GREY WOLF OPTIMIZATION TECHNIQUE

Amenah Nazar jabbar JABBAR

Information Technologies

Master`s Thesis

ALTINBAŞ UNIVERSITY

2023

The thesis titled IMAGE LEAF CLASSIFICATION FOR PLANT DISEASES DETECTION USING GREY WOLF OPTIMIZATION TECHNIQUE prepared by AMENAH NAZAR JABBAR JABBAR and submitted on 17/04/2023 has been **accepted unanimously** for the degree of Master of science in Information Technologies.

Asst. Prof. Dr. Hakan KOYUNCU

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Hakan KOYUNCU	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	Department of Computer Engineering, Altınbaş University	_____
Assoc. Prof. Dr. Hakan AYDIN	Department of Computer Engineering, İstanbul Topkapı University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master`s Thesis.

Submission date of the thesis to the Institute of Graduate Studies: ____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Amenah Nazar Jabbar JABBAR

Signature

DEDICATION

This thesis is dedicated to:

My supportive angel, my reflection, who I see myself through, my beloved mother. They say that the first love of a girl's life is her father... For the one who I felt love and confidence through my dear father. My wonderful sister, who shared my journey with me and was by my side step by step, all thanks to you and your family. My exceptional, loving brothers, my support in life, who make the world make sense. My dear uncle, who gave me the greatest support and helped me achieve one of my dreams. And finally, my gorgeous best friends, who believed in me, encouraged me, and were there for me.

ABSTRACT

IMAGE LEAF CLASSIFICATION FOR PLANT DISEASES DETECTION USING GREY WOLF OPTIMIZATION TECHNIQUE

JABBAR, Amenah Nazar Jabbar

M.Sc., Information Technologies, Altınbaş University,

Supervisor Asst. Prof. Dr. Hakan KOYUNCU

Date: April / 2023

Pages: 103

Plant ailments have the potential to significantly affect agriculture and cause substantial financial losses. They can affect crop yields and quality, leading to lower profits for farmers and higher prices for consumers. In some cases, plant diseases can also lead to food shortages and other economic and social consequences. Thus, it's critical to create efficient plans for managing and avoiding plant diseases. The identification and diagnosis of plant illnesses using learning techniques (ML and DL) can greatly reduce the harm and monetary losses brought on by plant diseases.

In this thesis a method for plant disease detection is proposed using several approaches. Three types of plant leaves are used in this thesis Peppers (two types), Potato (three types) and Tomato (Nine types). image resizing, and data augmentation are used as a pre-processing. Three type of feature extraction Histogram of Gradient (HOG), Local Binary Patterns (LBP) and Haralick feature are used. In order to choose the best features that characterize, a Grey Wolf Optimization (GWO) is employed as a feature selection method. Binary classification is carried by using Support Vector Machines (SVM) and K-Nearest Neighbour (KNN) used multi classification in Machin learning while deep learning is used also for classification all three types of planet leaf.

The proposed method had SVM algorithm for pepper plant achieved an accuracy of 93.14% at 1800 sample size and 10 k-fold, while the KNN algorithm achieved an accuracy of 89.30% for potato plant at 2202 sample size, 10 k-fold and accuracy of 95.18% for tomato plant at 12000 sample size ,10 k-fold. The CNN algorithm achieved an accuracy of 98.67%for pepper plant, 99.85%for potato plant and 91.80%.

Keywords: Grey Wolf Optimization (GWO), K-Nearest Neighbour (KNN), Support Vector Machines (SVM), Local Binary Patterns (LBP), Histogram of Gradient (HOG), Convolutional Neural Network (CNN), Grey Level Co-Occurrence Matrix (GLCM).

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

ML	:	Machine Learning
DL	:	Deep Learning
CNN	:	Convolution Neural network
KNN	:	K-Nearest Neighbour
LBP	:	Local Binary Pattern
HOG	:	Histogram of Oriented Gradient
SVM	:	Support Vector Machine
GWO	:	Grey Wolf Optimization
GLCM	:	Grey Level Co-occurrence Matrix

1. INTRODUCTION

1.1 BACKGROUND

Plants are an essential source of energy and an excellent solution to the issue of global warming. Still, they are prone to several illnesses that have serious adverse effects on the environment, society, and the economy. It may result in a severe decline in the yield and quality of crops [1]. Agriculture provides a living for 50% of the world's population. The leftovers from harvesting are utilized to increase the country's income and serve as a source of sustenance (for example, wax, oil, jute, etc.) [2].

In plant systems, emerging, re-emerging, and endemic disease damage is significant and can result in financial loss. Additionally, crop diseases both directly and indirectly aid in the spread of infectious illnesses that affect humans and harm the environment. The spread of these diseases' harms both the plant's ability to function normally and the economy by severely limiting the number of crops that can be grown [3]. Plant diseases can affect several components, including fruits, stems, and leaves. The most common bacterial, fungal, and viral diseases in plants are Alternaria, Anthracnose, bacterial spot, canker, and others. Environmental changes cause viral illness. Fungal disease is caused by fungus in leaf's, and bacterial infections are caused by germs in the leaves or plants. [4]. The output and quality of the world's agricultural products are frequently threatened by plant diseases, which also account for a sizeable amount of production expenses. Global food output is said to have decreased by at least 10% due to plant disease losses [5].

People are becoming more aware of the detrimental consequences extensive usage of chemical pesticides has on both the environment and their health. Foods made organically are preferred by consumers. Additionally, regulatory bodies like the European Union (EU) are tightening restrictions on the use of chemicals in agricultural goods imported into their markets. It is anticipated that these discoveries would decrease the use of pesticides in agriculture, highlighting the need of early and precise identification of pests and illnesses [6]. As plants are exposed to the outside environment and are very vulnerable to infections, plant disease prevention and management have been a topic of intense discussion for a long time. To effectively combat plant diseases, accurate and quick disease identification is often crucial. This is because accurate diagnosis frequently results in the implementation effective protective measures [7]. Plant diseases commonly reduce yields, quality, and economic

growth. This case clearly demonstrates that professional observation with the unaided eye requires a lot of work, and as a result, automation to identify leaf diseases consumes a significant amount of research [8].

It is now necessary to identify and diagnose plant diseases using information technology. To prevent such losses, a precise system for plant disease identification and diagnosis employing the best ML techniques is required. ML systems for plant disease identification need to excel in two key areas: speed and accuracy. Techniques such as automated plant disease identification and categorization must be developed by employing leaf image processing methods. Farmers will find this a beneficial strategy that will warn them just before the sickness spreads widely [9].

1.2 TYPE OF PLANT DISEASES

Crop leaves are particularly vulnerable to illness. Pathogens, including viruses, bacteria, fungus, and deficiencies, all have an impact on crops. Consequently, the pathogens causing the illness are divided into two groups: Saprophytes, which eat dead tissue, and autotrophs, which feed on living tissue. The disease's symptoms, which harm crop development and growth, are plain to see. The first sign of illness in plants is discoloured leaves.

Additionally, the texture and form of the leaves may be used to identify several illnesses. As a result, processing photographs of the leaves may be used to identify several illnesses, including mildew, rust, and powdery mildew [10]. There are three types of Viruses and contagious plants diseases, as shown in Fig. 1 [11]:

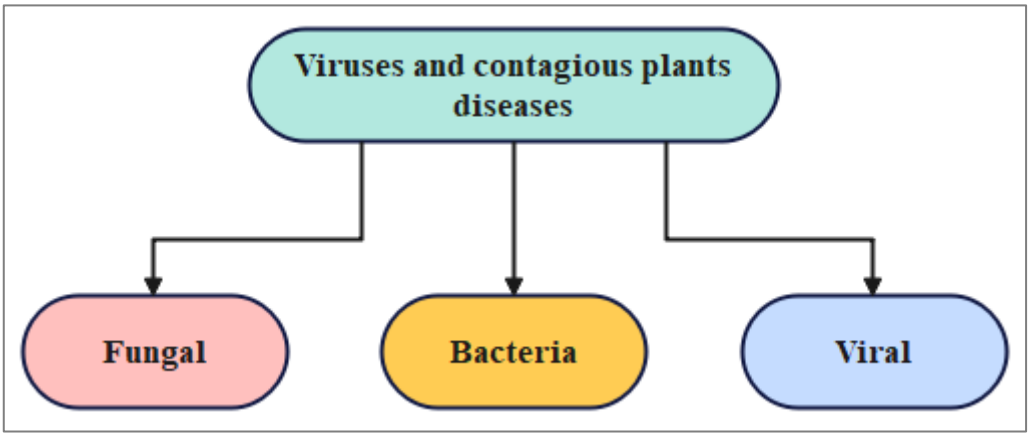


Figure 1.1: Types of Viruses and Contagious Plant Diseases.

a. Fungal Diseases

Fungi cause the majority of vegetable illnesses. Plants become infected by fungus when it stresses them and kills their cells. Fungal infections can come from various sources, including contaminated seeds, unsuitable soil, agricultural waste, neighbouring crops, and weeds. Their focus points are the organic apertures, like the stomata in plants. Fungal growth can also occur inside plants when artificial wounds are created by pruning, harvesting, flooding, insects, or other illnesses [12].

b. Bacteria Diseases

Many of plant's sections are susceptible to bacterial infections. Bacteria can inwardly impact even plants without exhibiting any apparent signs. Bacterially contaminated plants display symptoms such as cankers, leaf spots, overgrowth, scabs, wilts, and others. A bacterially infected plant may function as a source of infection for other plants nearby, rapidly dispersing the illness [13].

c. Viral Diseases

The most challenging viral infections to identify are those that affect plant leaves. The majority of viruses are concealed, which means that until they reach a specific level, they cannot be noticed. Mixing up viral infections with nutritional deficiency and pesticide damage is expected. Numerous common carriers can easily transmit viruses, such as aphids, leafhoppers, whiteflies, cucumber beetles, and insects [14].

1.3 PROBLEM STATEMENT

Since identifying plant diseases is an important activity in agriculture, plant diseases are typically detected visually, manually, or through microscopic examination. However, since visual identification depends on the viewer's psychological and visual state, it could be inaccurate and lead to some errors.

Plant diseases pose a severe threat to agricultural products, affecting food security and reducing farmers' profits. Early diagnosis of plant diseases enables treatment of affected plants through appropriate feeding methods to treat diseases early and to reduce losses in production and profit. However, it is possible to address this issue using over-the-counter fungicides and pesticides but doing so has a highly negative impact on both the human health

and environment. Early detection of plant diseases is essential to help growers take precautions to save affected plants.

1.4 THESIS AIM

Thesis goal is to build a diagnosis system that can find plant diseases by analysing images of affected plant leaves. The proposed system can detect abnormal cases of several types of plant diseases, helping farmers to identify plant diseases in their early stages. This system is trained to recognize multiple plant diseases using multiple classification techniques.

1.5 THESIS OUTLINES

The thesis is organized as follows:

Chapter One: It presents an overview of plants, their diseases, and the species that infect their leaves.

Chapter Two: It presents a background review of some similar work.

Chapter Three: Highlights the system's intended overall design.

Chapter Four: Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and DL based on the Convolutional Neural Networks (CNN) method are used in the proposed machine learning (ML) system.

Chapter Five: This chapter explains the results gained from implementing the proposed system.

Chapter Six: This chapter presents the conclusions that emerged from the application of the system, as well as recommendations for future work of the proposed system for plant diseases, taking in account the correct accuracy.

2. LITERATURE REVIEW

2.1 CHAPTER INTRODUCTION

A plant serves humanity by providing food and green vegetation cover that reduces the severity of global warming. It is also important in many areas of human life. and the fact that plant diseases cause serious negative effects on the environment, society, creatures' health and the economy. It has become necessary to find accurate and, at the same time, quick solutions to solve this problem. This section presents a theoretical background on plant disease detection via image processing and previous research that has dealt with this field.

2.2 RELATED WORKS

In 2020, P. Sindhu et al. [1], introduced a cloud-based and Internet of Things (IoT) methodology to diagnose illnesses affecting wheat leaves by using (ODNN). IoT devices are used for image acquisition. In order to identify the contaminated areas in the leaf photos, K-means clustering was applied. The thresholding approach was then used to remove unnecessary green regions from sick areas. From diseased leaves, characteristics including texture, colour, and form are retrieved. The ODNN model is applied to classify wheat leaf images using GWO Algorithm optimize parameters of DNN. This combination of DNN-OGWO achieve 95.92% precision, 96.41% recall and 96.96% accuracy.

In 2017, R. Meena Prakash, et al. [5], reported the use of image processing methods to categorize plant leaf diseases. The proposed system employed Grey-Level Co-Occurrence Matrix (GLCM) for the extraction of texture characteristics and K-means clustering for the segmentation of leaf pictures. The SVM was used to classify diseases in citrus leaves with accuracy of 90%.

In 2022, S. Harakannanavar et al. [12], recommended an ML technique for tomato plant leaf disease detection. the tomato sample is enhanced using histogram equalization after the tomato leaf pictures are downsized to 256x256 pixels. The dataspace is divided into Voronoi cells by the K-means clustering. The contour tracing approach is used to retrieve the leaf sample's boundary. The properties of leaf samples are extracted using a number of descriptors, such as Principal Component Analysis, the Discrete Wavelet Transform, and GLCM. The collected features are classified using Convolutional Neural Network (CNN),

Support Vector Machine (SVM), and K-Nearest Neighbour. Each classifier's accuracy is as follows: SVM (88%), K-NN (97%), and CNN (99.6%).

In 2020, Waheed, Abdul, et al**.** [13], present an optimized DenseNet model to recognize and classify corn leaf diseases with accuracy of 98.06%.

In 2020, N. Kaur and V. Devendran [14], Various plans have been put up for the automated detection of plant diseases. In order to classify leaf diseases with various sample classes and sizes, a convolutional low mask was used to extract texture features, with grey wolves' method for optimizing segmentation. In the end, the Potato, tomato, and bell pepper diseases are classified using the SVM classifier.

In 2019, Cecilia Sullca et al. [15], Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), Support Vector Machines (SVM) and Random Forest (RF) are combined with image processing methods to determine which is best for building a model that recognizes whether a blueberry plant is affected by a disease or pest, or whether it is healthy. Many filleters, including Medianblur and Gaussianblur, were employed to remove noise from blueberry images, and a weighted filter was utilized to improve the details. The Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) are used for feature extraction. Convolutional Neural Networks (CNN) can classify whether or not the blueberry plant is infected with an accuracy of 84%, according to the results.

In 2019, Mohammed A. Hussein et. al [16], A two-phase technique for detecting plant diseases has been suggested. The knowledge base is first built by processing a collection of training samples through a number of processing stages. These actions include obtaining a collection of colour and texture data using pre-processing methods including cropping, resizing, and fuzzy histogram equalization; training an SVM classifier utilizing the knowledge base for identifying and diagnosing plant leaf diseases; and more. The trained classifier is employed in the second step to identify and classify plant leaf ailments. This model has an accuracy rate of 88.1%.

In 2016, Erika Fujita et al. [17], Two CNNs are used to classify cucumber leaf diseases. These CNNs are trained with two datasets containing infected and healthy cucumber leaf images taken under good and bad conditions. This classification system achieved an average accuracy of 82.3% under the 4-fold cross-validation method.

In 2020 Anagnostis et al [18]. In the preceding work, a reliable CNN model was developed. The recommended method may categorize leaf images depending on whether they are contaminated with anthracnose and then determine if a tree is sick. Moreover, Fourier created a rapid transition to emphasize the highlights, some of the photos were utilized in RGB and grayscale modes, and CNN's design was chosen depending on how it would appear on the screen. In any event, the 98.7% accuracy of the suggested model produced positive outcomes.

In 2019 Geetharamani and Pandian [19]. A novel model that relies on a highly contentious neural network (CNN) has been created and put out to diagnose a leaf ailment in plants. An open data collection with 39 distinct classes and unique plant leaf photos was used to build the suggested model. Gamma management, image flipping, infusion, rotation, magnification, and principal component analysis were six categories of information augmentation techniques employed (PCA). Taking into account the outcomes, it was discovered that the application of knowledge can boost the model's exposure. In any case, the recommended model had a 96.4% accuracy rate and delivered good outcomes. In the end, the idea of plant leaf disease was excluded from the suggested model.

In 2019 Shanwen and et al [20]. To detect plant leaf disease, a three-channel CNN model was developed. The diminishing light of each CNN is learnt, and each channel guards one of the three darker zones of the RGB sick foliage. In order to further validate the illness, the most conspicuous portions of the fully connected common layer are joined at the following convolutional layer and pooling layer. The element is then used by the SoftMax layer to group the information photos into pre-established categories. As a consequence, the suggested methodology can precisely identify plant illnesses and extract the active component from intricately afflicted leaves. With an accuracy rate of 94.2%, the suggested system produced good results.

In 2019 Balakrishna and Rao [21]. Offered two techniques for growing and building an unwelcome tomato leaf. The KNN approach determines if the tomato leaf is healthy or undesirable in the early stages. Unfortunately, the PNN and KNN technique characterizes the tomato leaf in the next stage. To categorize objects, GLCM-like highlights, Gabor highlights, and shadow highlights are employed. The project focuses on the 600 voices and tragic characters in the writers' database. According to the findings, it is shown that the PNN

increases accuracy to 91.88%, however this percentage is insufficient and has to be combined with another algorithm in order to provide better results.

In 2017 Yang et al [22]. A fresh approach using convolutional neural networks (CNNs) to detect rice illness. With a dataset containing 500 regular photographs of sick and healthy rice leaves and stems taken from a rice test field, CNN is equipped to identify 10 underlying rice diseases. The findings indicate that the suggested framework accurately predicted outcomes with a precision of 95.48%.

Table 2.1: Comparison of Related Works**.**

Authors	**Classification Methods**	**Accuracy**
P. Sindhu et al. **[1]**	DNN-OGWO	96.96%
R. Meena Prakash et al. **[5]**	SVM	90%
S. Harakannanavar et al. [12]	SVM	88%
	KNN	97%
	CNN	99.6%
Waheed, Abdul et al. **[13]**	DenseNet	98.06%
N. Kaur and V. Devendran **[14]**	SVM	-----
Cecilia Sullca et al. **[15]**	CNN	84%
Mohammed A. Hussein et. al **[16]**	SVM	88.1%
Erika Fujita et al. **[17]**	CNN	82.3%
Anagnostis et al **[18]**	CNN	98.7%
Geetharamani and Pandian **[19]**	CNN	96.4%
Shanwen and et al **[20]**	CNN	94.2%
Balakrishna and Rao **[21]**	KNN, PNN	91.88%
Yang et al **[22]**	CNN	95.48%

2.3 SUMMARY OF LITERATURE REVIEW

This section summarized some prior research on the use of deep learning (DL) and machine learning (ML) approaches for the identification of plant leaf diseases.

3. METHOD AND MATERIALS

3.1 CHAPTER INTRODUCTION

This section explains the steps that will be followed to build the proposed system, as well as the basic methods used to prepare the data, extract its features, optimize it, and classify it.

3.2 IMAGE RESIZE

Image resizing is the process of altering an image's proportions, either by scaling it up or down, or by cropping and resampling it to a different aspect ratio. This is a typical procedure in digital image processing, and it can be carried out for a variety of reasons, including file size reduction, visual quality enhancement, and image adaptation for certain use cases. Image resize can be achieved using software tools, online platforms, or programming libraries [23].

There are several techniques for image resize, including:

a. Nearest Neighbour Interpolation: By applying the value of the closest pixel from the old picture to the new image, this approach creates a look that is blocky or pixelated.
b. Bilinear Interpolation: This technique uses the averaging of weights of the 4 nearest pixels in the original image to estimate the new pixel value in the resized image, resulting in smoother transitions.
c. Bicubic Interpolation: vs bilinear interpolation, this method produces even smoother transitions since it estimates the new pixel value using the averaging of weights of the 16 closest pixels in the original picture.
d. Lanczos Interpolation: This technique uses a more complex mathematical function to estimate the new pixel values, resulting in even higher quality resizing.
e. Resampling: This technique adjusts the image resolution, rather than its physical size, to produce a resized image with fewer or more pixels.
f. Cropping: This technique involves removing parts of the original image to change its aspect ratio, while preserving its resolution.

There are several benefits of image resize:

i. Improved computation time: Resizing images to a smaller size can speed up the processing time for computer vision tasks, particularly for large or high-resolution images.
ii. Reduced storage requirements: Resizing images to a smaller size can also reduce the storage requirements, making it more efficient to store and transfer the images.
iii. Better handling of aspect ratios: Resizing can be used to adjust the aspect ratio of an image to match the required dimensions, making it easier to display the image in various applications.
iv. Improved model performance: Resizing can be used to adjust the resolution of an image to match the input size requirements of a ML model, improving its performance.
v. Better visualization: Resizing can be used to adjust the size of an image for visualization purposes, making it easier to see details or patterns in the image.
vi. Improved image quality: By using high-quality interpolation methods during resizing, it is possible to raise the standard of the resized image, reducing the loss of details or introduction of artefacts.

Image resize is a useful technique for pre-processing and transforming images, allowing for better handling, computation, and analysis of the data.

3.3 DATA AUGMENTATION

It is a technique in ML and computer vision where existing data is transformed in a controlled manner to create additional, synthetic data. To improve the resilience and generalizability of machine learning models, the training data set's size and diversity must be increased.

Data augmentation performed by applying a set of predefined operations to the existing data, such as random rotations, translations, scaling, flipping, noise addition, and colour transformations. The model gains the ability to recognize the same objects or patterns under various circumstances thanks to these procedures, which imitate variances that may occur in real-world data [24].

Data augmentation can be performed on both image and non-image data, such as text, audio, and time-series data. In computer vision, data augmentation is commonly used for training object detection, segmentation, and classification models, as well as for improving the performance of generative models, such as Generative Adversarial Networks (GANs) [25].

Here are some common data augmentation techniques [26]:

a. Random Cropping: creating a new image by cropping a random area of the original picture.
b. Random Flipping: flipping the picture either vertically or horizontally to produce a new picture.
c. Random Rotation: Rotating the image by a random degree to create a new image.
d. Random Scaling: Scaling the image by a random factor to create a new image.
e. Colour Jittering: modifying an image's colour, contrast, saturation, or brightness at random to get a new one.
f. Gaussian Noise: Adding random Gaussian noise to the image to create a new image.
g. Random Translation: Translating the image by a random offset to create a new image.
h. Image Deformation: Warping or deforming the image to create a new image.
i. Cutout: masking off a section of the image at random to get a new image.

Depending on the particular needs of the work and the kind of the data, these strategies can be applied alone or in various combinations. The goal of data augmentation is to create a diverse and representative training set, which will help the model learn to generalize to unseen data and improve its performance [25].

There are several benefits of using data augmentation in machine learning:

i. Increased dataset size: Data augmentation generates new synthetic samples of the data already available, effectively increasing the size of the training set. This can help overcome the limitations of having a small or limited amount of data.
ii. Improved generalization: Data augmentation helps the model learn to recognize the same objects or patterns under different conditions, reducing overfitting and improving generalization to new, unseen data.
iii. Better model robustness: Data augmentation can train the model to become more resilient to noise, changes in lighting, and other real-world elements by exposing it to a larger variety of variances and distortions.
iv. Reduced overfitting: Data augmentation can minimize overfitting, which happens when a model is too tightly fitted to the training set of data, leading to poor generalization to untrained data.

v. Increased diversity: By transforming the existing data in controlled ways, data augmentation can improve the training set's variety, helping the model learn to recognize a wider range of variations and distributions.

vi. Computationally efficient: Data augmentation can be performed on the fly during training, reducing the need to manually acquire and store large amounts of additional data.

Generally speaking, Data augmentation is a useful technique for enhancing the functionality and generalization of ML models, especially when working with incomplete or unbalanced data [27].

3.4 FEATURE EXTRACTION

The capacity of humans to examine and classify items and settings is something that everyone benefits from every day [28]. In computer vision, we employ a variety of classifiers, each with their own set of properties and attributes [29]. The initial stage in comprehending a complicated scene is to distinguish the items, followed by the scene's categorization. Feature extraction is a phase in the dimensionality reduction procedure. It is simpler to extract the top features from massive datasets by choosing variables and combining them into features. While precisely representing the actual dataset, these functions are simple to understand and use [30]. Enhancing performance in areas like estimated visualization, accuracy, and comprehensibility of learned knowledge is the aim of feature extraction and selection algorithms. [31]. The benefit of feature selection is that it prevents the loss of crucial information related to a single feature, but if only a small subset of features is needed, and the original features are extremely diverse, there is a risk of information loss because some of the features will need to be dropped.

Yet, dimensionality reduction, also known as feature extraction, usually allows the feature space to be shrunk without sacrificing any knowledge of the original feature space. A drawback of feature extraction is the fact that the linear combination of the original features can occasionally be incoherent and that information about how much an original feature contributes is frequently lost [31].

3.4.1 Local Binary Pattern (LBP)

LBP was created by Ojala et al. [32] as an effective texture pattern descriptor for defining the LPB in an image. Applications for image processing frequently use it. The LBP uses a 3x3-bit block size, with the centre pixel acting as a threshold for the neighbouring pixel and the centre pixel's LBP code created by encoding the determined threshold value into a decimal value. LBP is described mathematically as follows [32]:

$$LBP = \sum_{i=0}^{p-1} s(n_i - G_c)2^i \tag{3.1}$$

$$(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases} \tag{3.2}$$

where P is the number of nearby pixels, n_i is the i th neighbouring pixel, and c is the centre pixel. The acquired LBP code is used to extract the histogram features of size 2P [33]. Therefore, eight neighbouring pixels have a histogram feature vector length of 256. Fig. 3.1 depicts the LBP process with a G_c value of 10 and eight surrounding pixels.

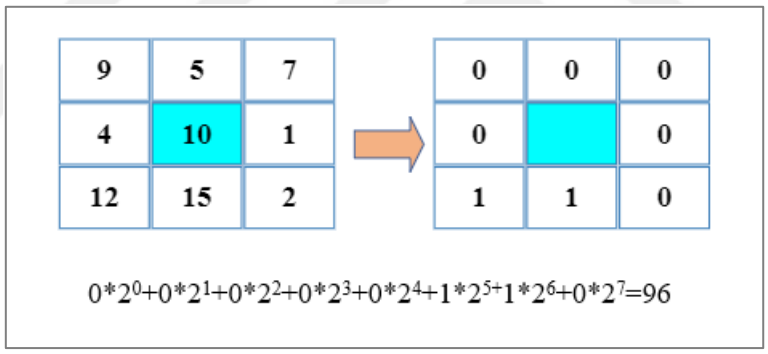


Figure 3.1: The Operation of LBP.

3.4.2 Histogram of Oriented Gradients (HOG)

The histogram of oriented gradients (HOG), also known as the Canny Edge Detector and Scale Invariant and Feature Transform SIFT, is a feature descriptor. It is used for object detection in computer vision and image processing. The technique counts how many times a gradient orientation appears in a certain region of a photograph. This technique is equal to Edge Orientation Histograms and Scale Invariant Feature Transformation (SIFT) [34]. The HOG description places a lot of attention on an item's form or structure. It performs better than other edge descriptors because it computes features based on both the magnitude and

angle of the gradient. It creates histograms for each section of the picture based on the sizes and orientations of the gradient. The procedures to determine HOG Features are [35]:

Step 1. Select an input image.

Step 2. Resize the input image, the common aspect ratio is (1:2).

Step 3. Calculate the gradient (G), considering a 3x3 block calculate the G_x using Eq. 3 and G_y using Equ. 4, for each pixel in the block.

$$Gx = I(r, c+1) - I(r, c-1) \tag{3.3}$$

$$Gy = I(r-1, c) - I(r+1, c) \tag{3.4}$$

Step 4. For each pixel calculate the magnitude using Eq. 5 and angle using Equ. 6.

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \tag{3.5}$$

$$Angle(\theta) = \left|tan^{-1}(G_y/G_x)\right| \tag{3.6}$$

Step 5. Divide the obtained gradient matrices (magnitude and angle matrix) into 8x8 blocks and calculate 9-point histogram for each block using Equ. 7 and the step size will be calculated using Equ. 8. The 9-point histogram creates a 9-bins histogram each bin has 20-degree angle range.

$$No.ofbins = 9(range[0^o, 180^o]) \tag{3.7}$$

$$Step - size(\Delta\theta) = 180^o/No.ofbins = 20^o \tag{3.8}$$

Each ith bin will have boundaries from $[\Delta\theta.i, \Delta\theta.(i+1)]$, and the center value of each ith bin will be $C_i = \Delta\theta(i + 0.5)$.

Step 6. Calculate the ith bin using Equ. 9, and then multiply it with Magnitude(μ) calculate the value of ith bin (V_i) using Equ.10. The value of ith+1 bin will be calculated using the Magnitude(μ) and the centre of ith bin as shown in Equ.11.

$$i = \left\lfloor\left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right)\right\rfloor \tag{3.9}$$

$$V_i = \mu \, . \, i \tag{3.10}$$

$$V_{i+1} = \mu \, . \left[\frac{\theta - C_i}{\Delta\theta} \right] \tag{3.11}$$

Step 7. An array is used as a bin for a block, and values of V_i and V_{i+1} are appended to the array at the index of the ith and ith+1.

Step 8. When the histogram computation for all blocks is completed, the 9-point histogram matrix is combined to generate a new block. To build the feature vector, this combing is done in an overlapping way with an 8-pixel stride.

3.4.3 Haralick Texture Features

Characteristics of the Haralick texture grey level co-occurrence matrix (GLCM) calculations are a popular way to represent image texture since they are straightforward to implement and produce a collection of comprehensible texture descriptors [36]. The procedures required to calculate Haralick texture characteristics in an image are outlined below. Create a co-occurrence matrix at the grey level as the initial step. The statistical characteristics of the co-occurrence matrix are used to create the Haralick features.

a. Grey-Level Co-occurrence Matrix (GLCM)

Texture analyses use the Grey Level Co-Occurrence Matrix (GLCM). Consider two pixels at a time, referred to as the reference and neighbour pixels. Before computing the GLCM, define a specific spatial connection between the reference and neighbouring pixels. For example, the neighbour may be 1 pixel to the right of the current pixel, 3 pixels above, or 2 pixels diagonally (one of NE, NW, SE, or SW) from the reference [37], as shown in Fig. 3.2

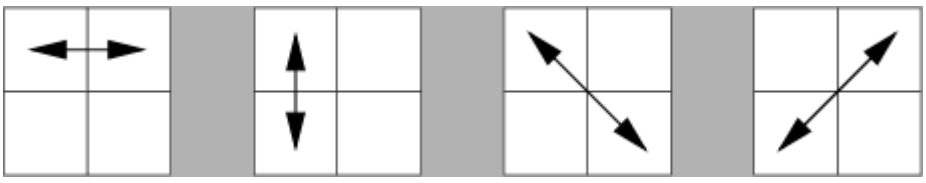


Figure 3.2 Adjacency in GLCM.

The GLCM matrix has $N_g \times N_g$ dimensions, where N_g is the number of grey levels in the image. The matrix element $P(i, j)$ is formed by counting the number of times a pixel with value i is next to a pixel with value j and then dividing the total number of such

comparisons done. Each item represents the probability that a pixel with value *i* will be discovered adjacent to a pixel with value *j*. Table 3.1 shows the statistical properties of GLCM [38].

Table 3.1: Statistical Properties of GLCM

Statistical properties	Description
$R = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i,j)$	The sum of all elements of the frequency matrix is the normalizing value R
$P(i,j) = \frac{P(i,j)}{R}$	co-occurrence probability matrix
$P_x(i) = \sum_{j=1}^{N_g} P(i,j)$	i-th entry in the marginal-probability matrix obtained by summing the rows of p(i,j).
$P_y(j) = \sum_{i=1}^{N_g} P(i,j)$	j-th entry in the marginal-probability matrix obtained by summing the columns of p(i,j).
$P_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \delta_{i+j,kP(i,j)}$	k=2, 3, …, 2Ng
$P_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \delta_{\lvert i-j \rvert,kP(i,j)}$	k=0, 1, …, Ng-1

where the Kronecker delta function $\delta_{n,m}$ is defined by:

$$\delta_{n,m} = \begin{cases} 1 & when\ n = m \\ 0 & when\ n \neq m \end{cases} \tag{3.12}$$

The second-order statistical information of an image's spatial connection is contained in the GLCM. Haralick presented 14 common statistical characteristics derived from GLCM, known as Haralick texture features [39], as illustrated in Table 3.2.

Table 3.2: Equations of Haralick Texture Features.

Features	Equations	Where
Angular 2nd Moment	$\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \left(\frac{P(i,j)}{R}\right)^2$	
Contrast	$\sum_{k=0}^{N_g-1} k^2 P_{x-y}(k)$	

Table 3.2: Equations of Haralick Texture Features "Tables Continued".

Correlation	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(ij)P(i,j)-\mu_x\,\mu_y}{\sigma_x\,\sigma_y}$	
Variance	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-\mu)^2P(i,j)$	
Inverse Difference Moment	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{1}{1+(i-j)^2}P(i,j)$	
Sum of Average	$\sum_{i=2}^{2N_g} iP_{x+y}(i)$	
Sum of Entropy	$\sum_{i=2}^{2N_g} P_{x+y}(i)log\left(P_{x+y}(i)\right)$	
Sum of Variance	$\sum_{i=2}^{2N_g}(i-SEn)^2P_{x+y}(i)$	
Entropy	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}P(i,j)\,log\big(P(i,j)\big)$	
Difference Variance	$Var\big(P_{x-y}\big)$	
Difference Entropy	$-\sum_{i=0}^{N_g-1}P_{x-y}(i)log\left(P_{x-y}(i)\right)$	
Maximal Correlation Coefficient	(2nd largest eigenvalue (Q))1/2	$Q(i,j)=\sum_k\frac{P(i,k)P(j,k)}{P_x(i)P_y(k)}$
Measures of Correlation #1	$\frac{Ent-H_1}{Max(H_x,H_y)}$	$H_1=-\sum_i\sum_j P(i,j)log\left(P_x(i)P_y(j)\right)$ $H_x=-\sum_i P_x(i)log\big(P_x(i)\big)$ $H_y=-\sum_j P_y(j)log\left(P_y(j)\right)$
Measures of Correlation #2	$\big[1-exp\big(-2(H_2-Ent)\big)\big]^{1/2}$	$H_2=-\sum_i\sum_j P_x(i)P_y(j)log\left(P_x(i)P_y(j)\right)$

3.5 FEATURES STANDARDIZATION

In practice, several types of variables might exist in the same dataset. The possibility of wide variations in the spectrum of variables is a major worry. Utilizing the original scale can give variables with a broad range more weight. To address this issue, certain strategies for feature rescaling to independent variables or data features are included in the data pre-processing stage [40]. The fundamental goal of feature standardization is to lessen the effects of different extracted feature values in order to get a classification that is consistent [41].

Scaling down characteristics to a comparable scale is known as normalization. As a result, the model's usability and training stability are improved [42]. There are several feature scaling methods in machine learning, the most commonly one is Standardization scaling, often referred to as Z-score normalization. As a result of standardization (or Z-score normalization), the features are rescaled to guarantee that the mean and standard deviation are between 0 and 1, as illustrated in Equ. 13 [43].

$$x' = \frac{x - \mu}{\sigma} \tag{3.13}$$

Where, μ reflects the feature value's mean, and σ reflects the feature values' standard deviation. This approach for rescaling feature values with a distribution value ranging from 0 to 1 is beneficial for optimization algorithms such as gradient descent, which are utilized in machine learning algorithms that weight inputs. Rescaling is also employed in algorithms that require distance measurements, such as the K-Nearest-Neighbours method (KNN) [43].

3.6 GREY WOLF OPTIMIZATION (GWO)

In 2014, Mirhalili Seyedali et al. [29] introduced the Grey Wolf Optimization Method, a novel swarm intelligence optimization technique. The tracking, enclosing, and hunting procedures, which were modelled after the predation behaviour of the Grey Wolf population, are used by the system to seek and optimize. GWO which involves picking a few wolves to steer the overall direction of the wolf population through competition inside the wolf population at each population update, includes both cooperation and competition within the wolf population [29]. These wolves are divided into four groups and live in a society, according to the strict social dominance hierarchy order depicted in Fig. 3.3 wolves named Alpha (α), Beta (β), Delta (δ), and Omega (ω) wolves [30].

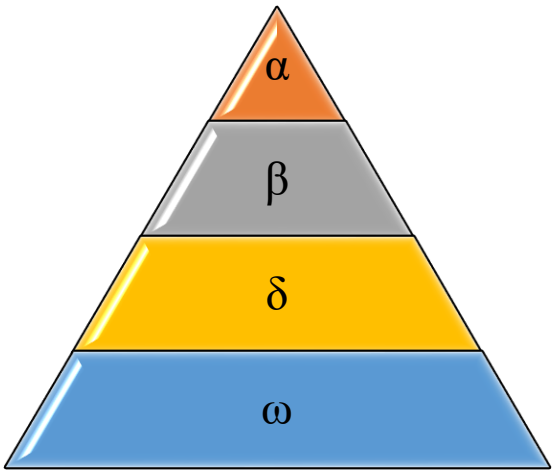


Figure 3.3: Grey Wolves' Social Hierarchy**.**

Alpha (α) is a group's overall leader, which issues commands to the three wolves below it. Beta (β) can direct the lower-ranked wolves and provide recommendations to the leader. The duty of delta (δ) is to decide what to do and put plans into action. Omega (ω) stands for the other wolves, who follow instructions and carry out tasks. The essential phases of grey wolf hunting are listed below. [31]:

i. Pursuing and tracking the prey
ii. Pursuit and encirclement of the prey, and
iii. Hunting the prey

3.6.1 Mathematical Model for Grey Wolf Optimization (GWO)

In developing GWO, take into account that the most suitable solution is the α to mathematically imitate the social structure of wolves. Hence, β and δ respectively, are names for the second and third-best solutions. All of the remaining potential answers are regarded as being omega. The hunting (optimization) in the GWO algorithm is led by α, β, and δ. These three wolves are followed by the ω wolves [44].

a. Encircling

As previously stated, grey wolves surround prey during hunting. The following equations are presented to mathematically describe encircling behaviour [45]:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{x_p}(t) - \vec{x}(t)\right| \tag{3.14}$$

$$\vec{x}(t+1) = \vec{x}_p(t) - \vec{A} \cdot \vec{D} \tag{3.15}$$

where t shows the most recent iteration, $\vec{A}$ and $\vec{D}$ are vectors of coefficients, $\overrightarrow{x_p}$ is the prey's location vector, and $\vec{x}$ shows a grey wolf's location vector. This is how the vectors $\vec{A}$ and $\vec{C}$ are calculated:

$$\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} - \vec{a} \tag{3.16}$$

$$\vec{c} = 2 \cdot \overrightarrow{r_2} \tag{3.17}$$

Where $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors in the range [0,1] and components of $\vec{a}$ are linearly decreased from 2 to 0 during the duration of repetitions.

b. Hunting

Grey wolves can locate and encircle prey, whereas the alpha (α) wolf normally looks for prey. While beta (β) and delta (δ) wolves may partake in hunting from time to time. There is, however, no predetermined concept of the best location (prey). To mathematically mimic a hunting approach, alpha is supposed to be the best possible answer, while beta and delta have a superior understanding of the likely location of prey. As a result, the three best solutions acquired so far are kept, and the remaining wolves (including omegas (ω)) are required to update their locations based on the position of the best response. In this regard, the following equations are offered [45]:

$$\vec{D}_a = \left|\vec{C}_1 \cdot \vec{x}_a - \vec{x}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{x}_\beta - \vec{x}\right|, \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{x}_\delta - \vec{x}\right| \tag{3.18}$$

$$\vec{x}_1 = \vec{x}_\alpha - \vec{A}_1 \cdot \left(\vec{D}_\alpha\right), \vec{x}_2 = \vec{x}_\beta - \vec{A}_2 \cdot \left(\vec{D}_\beta\right), \vec{x}_3 = \vec{x}_\delta - \vec{A}_3 \cdot \left(\vec{D}_\delta\right) \tag{3.19}$$

$$\vec{x}(t+1) = \frac{\vec{x}_1 + \vec{x}_2 + \vec{x}_3}{3} \tag{3.20}$$

A search agent can adjust its location in an n-dimensional search space using these equations in order to account for alpha, beta, and delta. The final position would also be randomly distributed inside a circle defined by the alpha, beta, and delta coordinates of the search space. To put it another way, the wolf's α, β and δ make an estimation of the location of the prey, while the remaining wolves update their locations at random around the prey [46].

c. Attacking

As previously stated, the grey wolves complete the hunt by attacking the target when it stops moving. The value of $\vec{a}$ is reduced in order to mathematically simulate approaching the prey. It is worth mentioning that decreases $\overrightarrow{A}$ fluctuation range. In other words, $\overrightarrow{A}$ is a random value in the range $[-2\alpha, 2\alpha]$, with a decreasing from 2 to 0 during the period of repetitions. When random values of $\overrightarrow{A}$ are in the range [-1, 1], the future position of a search agent can be anywhere between its present position and the position of the prey. Fig 3.4 shows that |A| < 1 forces the wolves to attack towards the prey [47].

Using the operators described thus far, the GWO algorithm enables its search agents to update their positions in relation to the positions of the alpha, beta, and delta and to move closer to the prey. On the other hand, the GWO algorithm is vulnerable to stalling in local solutions using these operators. The recommended encircling strategy does, in fact, show some exploration, but GWO needs more operators to emphasize exploration [45].

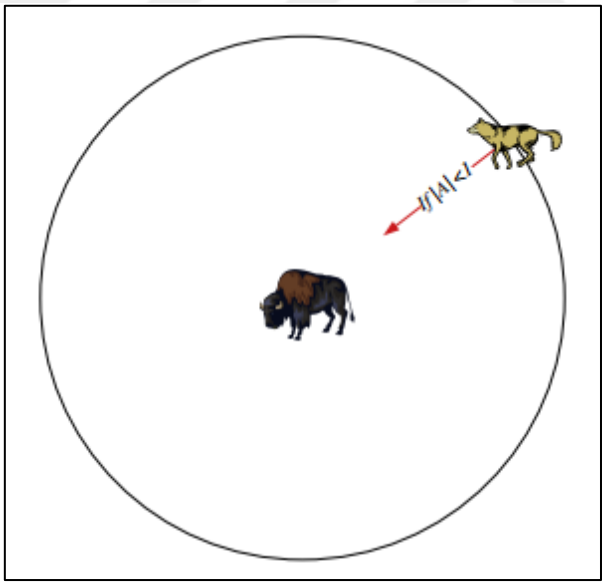


Figure 3.4: Attacking Prey.

d. Searching

Grey wolves usually search using the alpha (α), beta (β), and delta (δ) positions. They divide to look for prey and then converge to attack prey. $\overrightarrow{A}$ employs random values larger than 1 or less than -1 to force the search agent to diverge from the prey in order to mathematically describe divergence. This encourages exploration while also allowing the GWO algorithm

to explore widely. Fig 3.5 further illustrates that |A| > 1 causes the grey wolves to deviate from the prey in the hopes of finding a fitter prey [45, 47].

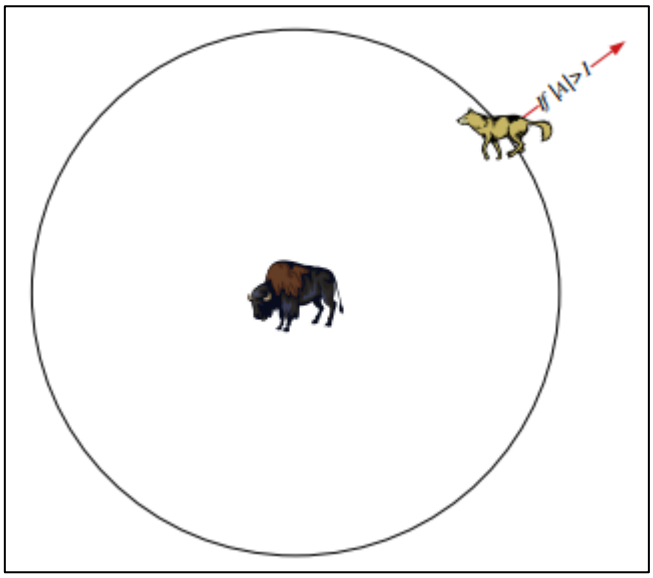


Figure 3.5: Searching for Prey.

$\overrightarrow{C}$ that shown in Equ. 24 is another component of GWO that encourages exploration, this vector contains random values between [0, 2]. This component offers random prey weights to stochastically emphasize (C > 1) or deemphasize (C < 1) the influence of prey in determining the distance in Eq. 21. This helps GWO behave more randomly during optimization, promoting exploration and local optimum avoidance. It is worth noting that, in contrast to A, C does not decrease linearly. C is supposed to always provide random values in order to emphasize exploration across all iterations, not just the initial one. When local optima stagnate, this component can be quite helpful, especially in the most recent versions [45, 48].

3.7 K-NEAREST NEIGHBORS (KNN)

The K Nearest Neighbour Method is a supervised learning method used in ML for regression and classification. It is a versatile method that may be applied to resample datasets and impute missing values [49]. As suggested by the name, it predicts the class or continuous value of a new data point using K Nearest Neighbours (data points). The KNN method assumes that neighbouring items of a similar nature exist. In other words, related items are close by. By computing the separation between data points on a graph, the KNN algorithm categorizes the data. [50].

For example, let (X_i, C_i) is a data point where (i=1, 2, …, n), X_i is feature values, and C_i is class labels for X_i for each i. Let p is unknown label point, the steps below are to find which class the point p is belong [56]:

Step 1. Calculate the distance D(p, X_i), Where i=1, 2, …, n.

Step 2. Arrange the calculated N distance in Ascending order.

Step 3. Choose K value.

Step 4. From the sorted list, take the first K distances.

Step 5. Find the K-points that match to the K-distances.

Step 6. Let K_i indicate the number of points in the i^{th} class among the K-points.

Step 7. If $K_i > K_j \; \forall \; i \neq j$ then put p in class C_i.

3.7.1 KNN: Distance Metrics

The distance between these data points must be calculated in order to discover which data points are closest to a particular data point. Decision boundaries, which separate data points into discrete areas, are formed with the use of these distance measurements [52]. There are several distance measures that can be used, such as:

a. Euclidean distance: is the most generally used distance metric, which is confined to real-valued vectors. The straight-line distance between the current data point and the other points is calculated using Equ. 21 [53].

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \tag{3.21}$$

b. Manhattan distance: Another common distance metric that quantifies the absolute value between two places. It is also known as taxicab distance or city block distance because it is usually represented by a grid, indicating how one may get from one location to another through city streets, as shown in Equ. 22 [54].

$$(x, y) = \sum_{i=1}^{n} |x_i - \; y_i| \tag{3.22}$$

c. Minkowski distance: This distance metric is a synthesis of the Euclidean and Manhattan distance metrics. Other distance metrics can be calculated using the parameter p in Equ.

23. This formula represents Euclidean distance when p equals two, and Manhattan distance when p equals one [55].

$$\mathrm{d}(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|\right)^{1/p} \tag{3.23}$$

d. Hamming distance: This technique is frequently used to find areas where two vectors do not match when they are Boolean or string vectors. Hence, it is sometimes referred to as the overlap metric [56]. Equ. 24 may be used to express this:

$$D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right) \tag{3.24}$$

$$Where = \begin{cases} x = y & D = 0 \\ x \neq y & D \neq 1 \end{cases}$$

3.8 CONVOLUTION NEURAL NETWORK (CNN)

A CNN Fig 3.6 is a kind of deep learning neural network that is very effective at computer vision applications like image recognition. The visual cortex, which in the brain is in charge of processing visual data, served as the model for CNNs [57].

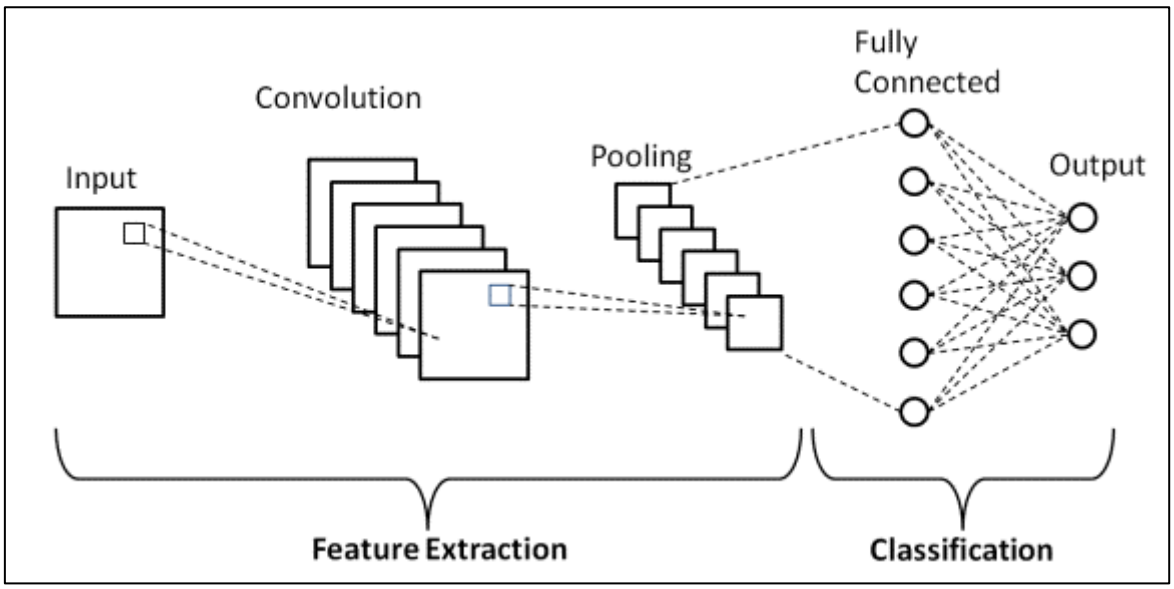


Figure 3.6: Convolution Neural Network Architecture.

The convolutional layer, which performs a convolution operation on the input data, is the fundamental component of a CNN. A tiny kernel or filter is moved over the input picture

during a convolution process, and the dot product between the kernel and the image is determined at each place. A new feature map is created as a consequence, emphasizing the edges and textures of the input picture [58].

Each layer of a convolutional algorithm may learn a separate characteristic of the picture and can be placed on top of the others. Following the convolutional layers, the feature maps' spatial dimension is generally reduced using one or more pooling layers. The feature map's spatial resolution is decreased, and the computational cost is decreased by using the pooling layer [59].

Following the pooling layers, there are usually a layer or layers that are fully connected that are used to categorize the picture using the attributes that the convolutional and pooling layers learnt. A probability distribution across all possible classes, showing the chance that the picture belongs to each class, is the final output of a CNN [57].

Several different image identification tasks, including as object detection, facial recognition, and picture segmentation, have seen great success using CNNs. They have also been used to other disciplines, including voice recognition and natural language processing, among others. The suitability of CNNs for these tasks and their widespread usage in both business and research stem from their capacity to learn hierarchical representations of the data [60].

The CNN architecture consists of several layers such as [58]:

a. Input Layer: The input layer is where the network receives the raw picture data.
b. Convolutional Layer: In order to extract characteristics like edges, textures, and patterns from the input picture, a collection of filters are applied in this layer. Each filter creates a feature map by sliding it over the input image.
c. Pooling Layer: This layer shrinks the feature maps' spatial dimensions by averaging or calculating the maximum value of a tiny window of pixels. By doing this, the computational effort is reduced, and the characteristics are strengthened against minor changes in the input data.
d. Fully Connected Layer: This layer, often referred to as the thick layer, joins all the neurons from the preceding layers and completes the categorization of the characteristics that the preceding layers have retrieved.
e. Output Layer: The output of the network is generally a probability distribution over a collection of classes when it reaches this layer.

During training, the CNN algorithm employs backpropagation to modify the weights and biases of the network. This is accomplished by calculating the error and comparing the output of the network to the expected output. The network then propagates the mistake back through it in order to update the layer weights and biases.

It's worth to mention that, CNNs are usually trained on large datasets, and the training process can take a significant amount of time and computational resources, but once the CNN is trained, it can perform the task it was trained for with high accuracy and efficiency. Also, with the recent advancements in hardware, it's become easier to train deep CNNs and use them in various applications [61].

3.9 SUPPORT VECTOR MACHINES (SVMS)

Supervised machine learning algorithms called SVMs are utilized for classification and regression problems. They are founded on the idea of locating the hyperplane that classifies the data into distinct groups or forecasts the result for regression tasks [62].

An SVM's objective in a binary classification job is to choose the hyperplane with the highest margin—that is, the distance between the nearest data points from each class and the hyperplane—that divides the data into two classes. Support vectors, or the data points closest to the hyperplane, are very important in establishing where the hyperplane is [62].

By merging the predictions from many binary classifiers that have been trained, SVMs may also be utilized for multi-class classification. Finding a function that fits the data and minimizes the variations between the expected and actual results is the objective of regression tasks.

One of the main benefits of SVMs is their ability to handle non-linearly separable data by converting the input data into a higher-dimensional space where a linear hyperplane may be utilized to split the classes. SVMs are also effective in terms of memory utilization and training time, making them appropriate for large-scale data sets [63].

The choice of the kernel function and hyperparameters might have an impact on the model's performance, which is one of the drawbacks of SVMs. The decision boundary of an SVM is specified in a high-dimensional space, which makes it challenging to comprehend the findings.

The steps to train an SVM for a classification or regression task are [64]:

a. Data pre-processing: The data must be cleaned, the inputs must be normalized or standardized, and the data must be divided into training and testing sets.
b. Choose a kernel function: The kernel function that determines the mapping of the input data into a higher-dimensional space must be chosen at this phase. Radial basis function (RBF) kernels, linear, polynomial, and other common kernel functions.
c. Train the model: The training process involves finding the hyperplane that separates the data into different classes or predicts the output for regression tasks. This is done by optimizing an objective function that balances the margin maximization and the classification or regression error.
d. Evaluate the model: The performance of the trained model is assessed using measures like accuracy, recall, precision, and mean squared error on the testing data.
e. Fine-tune the model: If the performance is not satisfactory, the model can be fine-tuned by adjusting the hyperparameters, such as the regularization parameter or the kernel parameters.
f. Deploy the model: The model may be used in a production setting to generate predictions on fresh data after it has been trained and assessed.

It's worth noting that the exact implementation details of the SVM algorithm can vary based on the software library or programming language being used.

3.10 CLASSIFICATION METRICS

Because classification algorithms produce distinct outputs, there must be metrics to assess how well these outputs perform. So, in order to evaluate classification models, there are several metrics to measure their performance, which are as follows [65]:

a. Accuracy: The most straightforward statistic to use and use is classification accuracy, which is calculated as the number of correct predictions divided by the total number of predictions multiplied by 100.

$$Accuracy = \frac{TP + TN}{N} * 100 \tag{3.25}$$

b. Precision: is the proportion of results produced by the system that accurately anticipated positive observations (True Positives) to all accurately predicted positive observations

(True Positives) and falsely forecasted positive observations (False Positives) (False Positives).

$$Precision = \frac{TP}{TP + FP} \tag{3.26}$$

c. Recall: The proportion of system-generated findings (True Positives) that accurately predicted positive observations to all observations in the real malignant class (Actual Positives).

$$Recall = \frac{TP}{TP + FN} \tag{3.27}$$

d. Confusion Matrix: Is a tabular representation of ground-truth labels vs. model predictions. The cases in a predicted class are represented by each row of the confusion matrix, whereas the cases in an actual class are represented by each column. Confusion Matrix isn't technically a performance statistic, but it serves as a foundation for other metrics to analyse the results.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Figure 3.7: Confusion Matrix [66].

3.11 SUMMARY OF MATERIALS AND METHODS

In this part, the steps followed for the suggested system were explained at the beginning. The size of the images of the data set used was changed, as well as the data was augmented, and this is considered the process of preparing the data. After completion, the features of this data were extracted by LBP, HOG and harailk, and these features were passed to the grey wolf algorithm, to be then classified in several ways which is SVM, KNN and CNN.

4. PROPOSED METHOD

4.1 CHAPTER INTRODUCTION

The stages that the suggested system has undergone are presented in this portion, as seen in Figure (4.1), the proposed system going through several phases which are as follow:

a. Phase One: Data Preprocessing using Image Resize and Data Augmentation and splitting the data after preprocessing into two parts: 80%training: 20%testing.
b. Phase Two: Feature extraction using three methods Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and Haralick Texture Features.
c. Phase Three: Features selection using Grey Wolf Optimizer (GWO).
d. Phase Four: Train three algorithms, Support Vector Machines (SVM), Convolutional Neural Network (CNN), and K-Nearest Neighbors (KNN).
e. Phase Five: Models Testing using testing data.

This chapter demonstrates the construction of every stage of the suggested system as well as the definition of each phase's inputs and the processing that took place to produce the necessary outputs that were subsequently utilized in the other phases. These stages are designed to treat the data with care that ensures the development of a model for detecting plant diseases.

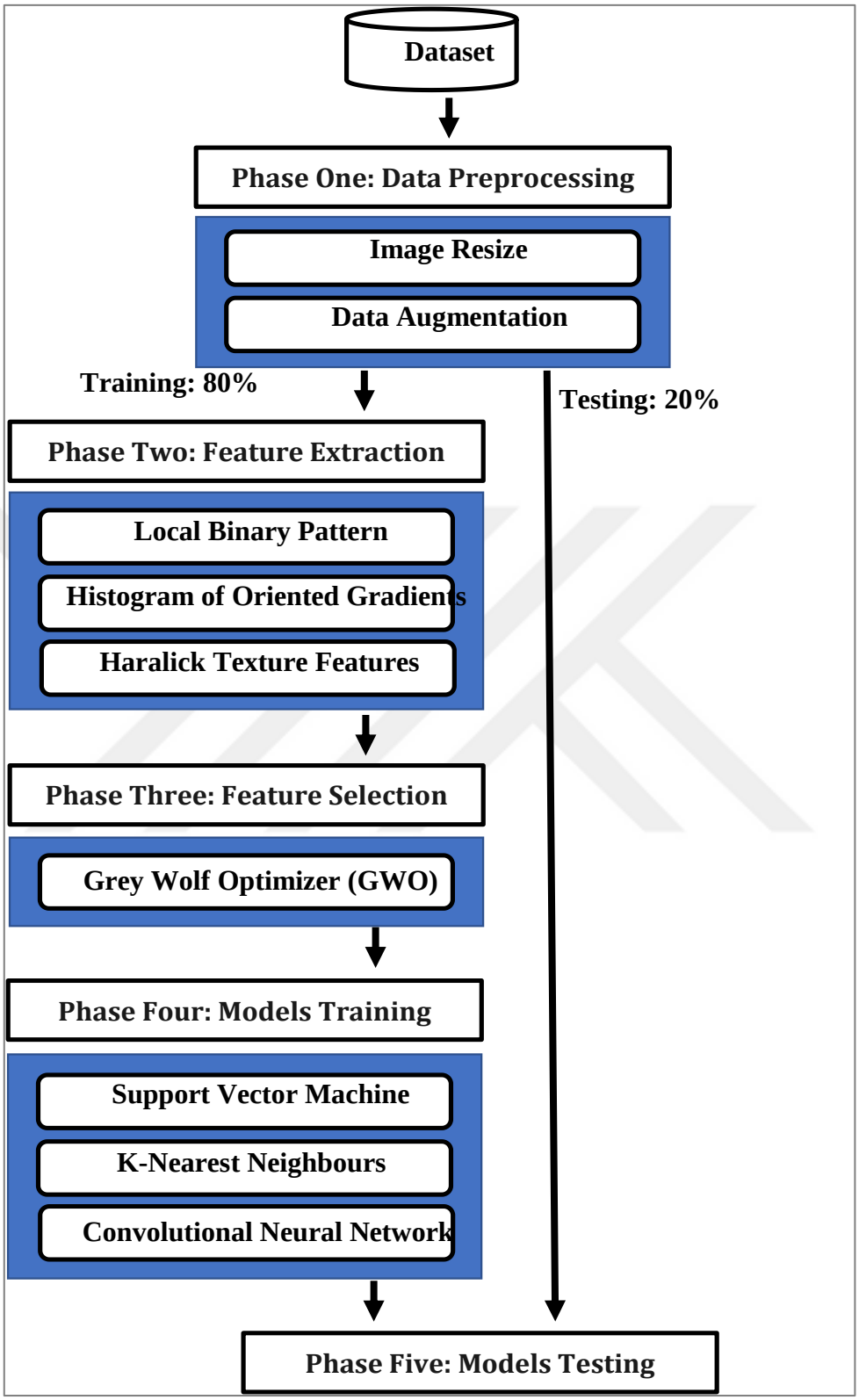


Figure 4.1: Proposed System Block Diagram**.**

4.2 DATASET

The PlantVillage dataset is a large collection of images of plant diseases. It was compiled by researchers at Penn State University and is intended to help develop machine learning models for detecting and identifying plant diseases from images. Almost 50,000 photos are included in the collection. of over 38 different species of plants, with diseases ranging from bacterial spot to rust to powdery mildew. Each image is annotated with the disease that is present in the image, as well as the plant species. The dataset is widely used in research on plant disease detection and has been used in a number of published papers. It is available for download from the PlantVillage website.

In this thesis only three types of planets were selected (Bell Pepper, Potato, and Tomato) as shown in table 4.1 The PlantVillage dataset includes 2,455 images of bell pepper plants, 2,152 images of potato plants, and 16,013 images of tomato plants.

Table 4.1: PlantVillage Dataset**.**

Diseases / **Plants**	**Fungi**	**Bacteria**	**Mold**	**Virus**	**Mite**	**Healthy**
Pepper (2455)	-	977	-	-	-	1478
Potato (2152)	2000	-	-	-	-	152
Tomato (16,013)	6084	2127	952	3582	1676	1592

Below is a sample of normal leaves and abnormal leaves a. Healthy leaf, b. Early Blight (Alternaria solani), c. Late Blight (Phytophthora Infestans), d. Septoria Leaf Spot (Septoria lycopersici), e. Yellow Leaf Curl Virus (Family Geminiviridae genus Begomovirus), f. Bacterial Spot (Xanthomonas campestris pv. vesicatoria), g. Target Spot (Corynespora cassiicola), h. Spider Mite (Tetranychus urticae).

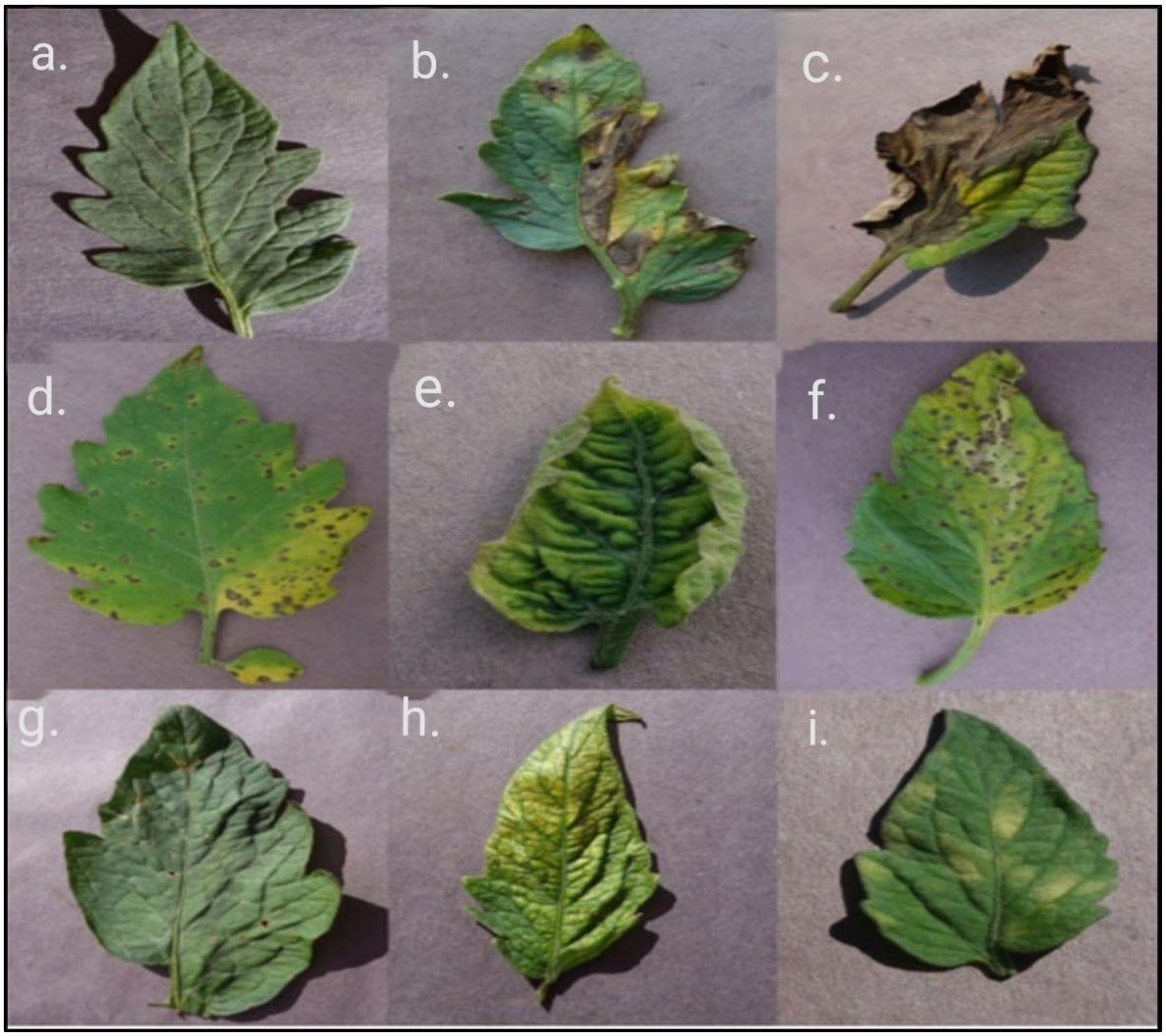


Figure 4.2: Examples of Each Plant Diseases**.**

4.3 DATA PRE-PROCESSING

Image pre-processing is the process of preparing images for analysis. It is a vital stage in many image processing and computer vision applications since the picture's quality and structure frequently affect whether the analysis that follows will be successful or not. The Data pre-processing phase involves two steps which are as follow:

4.3.1 Image Resize

Resizing an image means changing the dimensions of the image, either by altering how many pixels there are in the image or by changing image size file on disk. To ensure that all dataset images have the same size, the dataset images going to resize step as shown in algorithm 4.1 to set their size to [256 x 256].

Algorithm 4.1: Images Resize.

Input: Dataset
Output: ResizedDataset, InputSize
Step 1. InputSize ← [256 256 3] // *Define New Image Size* Step 2. For each image in Dataset // *Loop on Dataset Images* Step 3. ResizedDataset ← Resize (image, InputSize) Step 4. End For Step 5. Return ResizedDataset Step 6. Return InputSize

4.3.2 Data Augmentation

Data augmentation is a method for artificially growing a dataset by creating new data samples from the ones that already exist. This can be useful when you have a small dataset and want to make your machine learning model more robust and better able to generalize to new data.

There are many ways to perform data augmentation, but some common techniques include:

i. Rotating or flipping images: You can rotate or flip an image and use the transformed version as a new data sample.
ii. Cropping images: You can crop an image to create a new data sample with a different aspect ratio or to focus on a particular part of the image.
iii. Adding noise: You can add random noise to an image to create a new data sample that is slightly different from the original.
iv. Adjusting brightness or contrast: You can adjust the brightness or contrast of an image to create a new data sample with different lighting conditions.

Data augmentation can be a powerful tool for improving the performance of machine learning models, but it is important to use it carefully and make sure that the augmented data is still representative of the real-world data that the model will encounter.

In this case the X-reflection, Y-reflection, and rotation operations on the dataset images was used to increase the size of the available data as shown in algorithm 4.2.

Algorithm 4.2: Data Augmentation.

Input: ResizedDataset
Output: AugmentedDataset
Step 1. For each image in ResizedDataset // *Loop on Dataset Images* Step 2. AugmntedDataset ← X-reflection (image) Step 3. AugmntedDataset ← Y-reflection (image) Step 4. AugmntedDataset ← Rotation (270^o, image) Step 5. End For Step 6. Return AugmentedDataset

At this point, the Augmented Data is divided into two parts: 70% training: 30% testing.

4.4 FEATUR EXTRACTION

Feature extraction is a process in which certain characteristics or features of an image are extracted and made available for further processing. These features can include things like colour, texture, shape, or other properties of the image. The classification of plant leaf diseases was greatly aided by these characteristics. There are many different algorithms and techniques that can be used for feature extraction from images. Some common approaches include:

a. Colour histograms: A colour histogram is a graphical representation of the distribution of colours in an image. It can be used to extract features related to the colour content of the image.
b. Edge detection: Edge detection algorithms can be used to identify the boundaries of objects in an image. This can be useful for extracting features related to the shape of objects in the image.

c. Texture analysis: Texture analysis algorithms can be used to identify patterns or repeating structures in an image. This can be useful for extracting features related to the texture or surface characteristics of objects in the image.
d. Feature vectors: A feature vector is a set of numerical values that represent the characteristics of an image. These values can be used as input to machine learning algorithms or other image processing systems.

There are countless characteristics available in image processing that may be retrieved and used to classify images in accordance with requirements. To get better outcomes, selecting the appropriate feature set and comprehending which characteristics should be extracted are crucial. The features in this study were extracted using LBP, HOG, and Haralick Texture Features.

LBP is a texture descriptor that describes the texture of an image in computer vision and image processing. It works by dividing an image into small regions and comparing the values of neighbouring pixels within each region. Algorithm 4.3 show the steps of calculating LBP descriptor for each image in Augmented Dataset.

Algorithm 4.3: Local Binary Patterns (LBP).

Input: TrainingData
Output: LBPfeatures
Step 1. For each image in TrainingData // *Loop on Dataset Images* Step 2. The image must be divided into 8x8 "cells" Step 3. For each pixel in the cell Step 4. Comparing the value of current pixel to the values of its neighbours. Step 5. Encoding the result of these comparisons as a binary code, where: Step 6. "1" indicates that the value of the central pixel is greater than or equal to the value of its neighbour. Step 7. "0" indicates that it is less than the value of its neighbour. Step 8. Repeat this process for each pixel in the cell.

Algorithm 4.3: Local Binary Patterns (LBP) “Algorithms Continued”.

Step 9. LBPfeatures ← Combining the binary codes for all of the pixels in the cell to create a single "LBP code" for the cell.

Step 10. End For

Step 11. Return LBPfeatures

HOG is a feature descriptor used in image processing and computer vision to describe the structure and form of an item in an image. It works by dividing the image into small cells and calculating the distribution of intensity gradients within each cell. Algorithm 4.4 show the steps of calculating HOG descriptor for each image in Augmented Dataset.

Algorithm 4.4: Histogram of Oriented Gradients (HOG).

Input: TrainingData

Output: HOGfeatures

Step 1. For each image in TrainingData // *Loop on Dataset Images*

Step 2. Divide the image into (8x8) regions, called "cells"

Step 3. For each cell in image

Step 4. Calculate the gradient magnitude and direction for each pixel in the cell.

Step 5. Create a histogram of the gradient orientations for the cell, with each bin representing a range of orientations.

Step 6. HOGfeatures ← Normalize the histogram to account for differences in cell size and contrast.

Step 7. Repeat this process for each cell in the image to create a descriptor for the entire image.

Step 8. End For

Step 9. End For

Step 10. Return HOGfeatures

An image's texture may be described using the Haralick descriptor, a texture descriptor used in computer vision and image processing. Algorithm 4.5 show the steps of calculating Haralick descriptor for each image in Augmented Dataset.

Algorithm 4.5: Haralick Descriptor.

Input: TrainingData
Output: HaralickFeatures
Step 1. For each image in TrainingData // *Loop on Dataset Images* Step 2. Divide the image into (8x8) regions, called "cells" Step 3. For each cell in image Step 4. Calculate the co-occurrence matrix. Step 5. HaralickFeatures ← Calculate statistical measures (mean, variance, and correlation) of the elements in the co-occurrence matrix. Step 6. Repeat this process for each cell in the image to create a descriptor for the entire image Step 7. End For Step 8. End For Step 9. Return HaralickFeatures

4.5 FEATURE SELECTION

To increase the model's accuracy and interpretability, feature selection is the process of choosing a small subset of pertinent characteristics for usage in the modelling process. It is frequently used in data mining and machine learning to choose the most crucial characteristics for a certain job, like classification or regression. The choice of features for a ML model must be made carefully because it can significantly affect the model's performance. Feature selection can assist increase the model's precision, lessen overfitting, and enhance the model's interpretability.

The Grey Wolf Optimizer (GWO) is an optimization technique with naturalistic roots. It is employed to locate the global optimum of a particular optimization issue and is based on the social behaviour of grey wolves. The GWO algorithm works by simulating the behaviour of a pack of grey wolves in search of prey. The alpha wolf, beta wolf, and delta wolf are the algorithm's three major operators. The alpha wolf is the leader of the pack and is responsible for guiding the search. The beta wolf is the second-in-command and is responsible for exploring the search space. The delta wolf is the third-in-command and is responsible for exploiting the best solutions found so far.

Grey Wolf Optimizer (GWO) can be used for feature selection by treating the features as the dimensions of the search space and the objective function as a gauge of the model's effectiveness. The objective is to identify the subset of characteristics that improves the model's performance. To use GWO for feature selection, the search space, the objective function, and the stopping criteria must be defined. The search space would be the set of all possible feature subsets, and the objective function would be a measure of the model's performance on a given dataset using a particular subset of features. The stopping criteria could be based on the number of iterations, the improvement in the objective function, or some other measure. Once these parameters defined, the GWO algorithm can be used to search for the optimal subset of features. The alpha, beta, and delta wolves would correspond to different feature subsets, and their positions would be updated based on the objective function value and the positions of their neighbours. The algorithm would continue until the stopping criteria are met, at which point the best-performing feature subset would be returned. Algorithm 4.6 show how to use Grey Wolf Optimizer (GWO) for feature selection using LBP, HOG, and Haralick Texture Features.

Algorithm 4.6: Feature Selection Using GWO.

Input: LBPfeatures, HOGfeatures, HaralickFeatures
Output: GWOFeatures
Step 1. Load the LBP, HOG, and Haralick features Step 2. Concatenate the features $X = [\text{LBP HOG Haralick}];$

Algorithm 4.6: Feature Selection Using GWO "Algorithms Continued".

Step 3. Define the search space.

```
d = size(X, 2);

searchSpace = zeros(d, d);
```

Step 4. Define the objective function.

```
objFcn = @(subset) modelPerformance(X(:, subset), y);
```

Step 5. Define the stopping criteria.

```
maxIter = 100; // Max Iteration
```

Step 6. Initialize the positions of the alpha, beta, and delta wolves.

```
alpha = randi(d);

beta = randi(d);

delta = randi(d);
```

Step 7. Initialize the best solution.

```
bestSol = zeros(d, 1);

bestVal = -inf;
```

Step 8. Run the GWO algorithm.

```
For iter ← 1 to maxIter

// Calculate the objective function value for each wolf

alphaVal = objFcn(alpha);

betaVal = objFcn(beta);

deltaVal = objFcn(delta);

// Update the positions of the wolves

alpha = updatePosition (alpha, alphaVal, betaVal, deltaVal, searchSpace);
```

Algorithm 4.6: Feature Selection Using GWO" Algorithms Continued".

```
        beta = updatePosition (beta, alphaVal, betaVal, deltaVal,
searchSpace);

        delta = updatePosition (delta, alphaVal, betaVal, deltaVal,
searchSpace);

        // Update the best solution

        if alphaVal > bestVal

            bestSol = alpha;

            bestVal = alphaVal;

        end

        if betaVal > bestVal

            bestSol = beta;

            bestVal = betaVal;

        end

    if deltaVal > bestVal

     bestSol = delta;
        bestVal = deltaVal;

        end

        GWOFeatures ← bestVal

        end For

Step 9. Return GWOFeatures
```

4.6 MACHINE MODEL

A branch of artificial intelligence called "machine learning (ML)" focuses on creating algorithms and models that can learn from data and make predictions or judgments based on it. Algorithms for ML are created to automatically get better when they are exposed to fresh data. K-Nearest Neighbours (KNN) is a supervised machine learning technique used for classification and regression, and two different types of machine learning models will be employed in this thesis. On the basis of input data, they may learn to spot patterns and make predictions. These models use support vector machines (SVMs) to locate the hyperplane in a high-dimensional space that best divides several classes.

4.6.1 Support Vector Machine (SVM)

A supervised machine learning technique known as an SVM may be applied to classification or regression problems. Finding the hyperplane in a high-dimensional space that optimally separates several classes is the objective of an SVM.

When the data cannot be separated linearly, SVMs are very helpful because they may utilize the kernel method to move the data into a higher-dimensional space where it can be separated linearly. The linear kernel, polynomial kernel, and radial basis function (RBF) kernel are a few examples of typical kernels used in SVMs. Algorithm 4.7 show how to train an SVM classifier using feature data and training data.

Algorithm 4.7: Train SVM Classifier.

Input: GWOFeatures, TrainingData, TestingData
Output: SVMmodel
Step 1. Load the GWOFeatures and TrainingData. Step 2. Train the SVM classifier using linear kernel. SVMmodel = fitcsvm(X, y, 'KernelFunction', 'linear'); Step 3. Load the TestingData. Step 4. Predict labels for the test data.

Algorithm 4.7: Train SVM Classifier "Algorithms Continued".

results = predict (SVMmodel, Xtest);
Step 5. Return SVMmodel

4.6.2 K-Nearest Neighbors (KNN)

KNN is a supervised ML method used for regression and classification. Finding the K-number of training samples that are closest to a new sample allows it to be classified based on the majority class of its neighbours. The idea is that similar samples have similar class labels. It is simple to implement and effective in high-dimensional spaces. In KNN, the output is decided by the feature space's k nearest neighbours who cast the most votes for it. The algorithm provides a class label to an input data point based on the class of its closest neighbours after classifying the input data point. In this case there are 14 class, two for pepper, three for potato, and nine for tomato that is mean the value of K=14, algorithm 4.8 show the steps of K-Nearest Neighbours (KNN) training using training data and feature data.

Algorithm 4.8: Train KNN Classifier.

Input: GWOFeatures, TrainingData, TestingData
Output: KNNmodel
Step 1. Load the GWOFeatures, TrainingData, and TestingData.
Step 2. Convert the feature data and training data into matrices.
GWOFeatures = cell2mat(struct2cell(GWOFeatures));
TrainingData = cell2mat(struct2cell(TrainingData));
Step 3. Split the training data into features and labels.
X = GWOFeatures(TrainingData(:,1),:);
Y = TrainingData(:,2);
Step 4. Train the KNN classifier using Training Data.

Algorithm 4.8: Train KNN Classifier "Algorithms Continued".

```
KNNmodel = fitcknn (X, Y,'NumNeighbors', 14, 'Standardize', 1);
```

Step 5. Use the trained classifier to make predictions on the testing data.

```
XTest = GWOFeatures(TestingData(:,1),:);

YTest = predict(mdl,XTest);
```

Step 6. Return KNNmodel

In this algorithm, the fitcknn function is used to train the KNN classifier on the training data X and Y, with NumNeighbours set to 5. The Standardize option is set to 1, which indicates that the features should be standardized before training. The predict function is then used to make predictions on the testing data XTest, and the predicted labels are stored in YTest.

4.6.3 Convolutional Neural Network (CNN)

A branch of machine learning called "deep learning (DL)" is motivated by the structure and operation of the human brain. It includes training artificial neural networks, which are built of numerous layers of linked "neurons," using massive quantities of data.

DL models have been utilized to generate cutting-edge outcomes in a variety of applications, including image identification, processing, natural language and speech synthesis. DL models have the ability to learn and extract characteristics from raw data. CNNs, which are best suited for image classification tasks, and recurrent neural networks (RNNs), which are built to process sequential data such as time series or spoken language, are two examples of DL models. CNNs are a special kind of DL model that excel in image categorization tasks. They are made up of several linked layers of synthetic "neurons" and are intended to analyse data having a grid-like architecture, such as an image. Convolutional layers and pooling layers are the two primary types of layers found in CNNs. Convolutional layers perform a convolution operation to the input data, which uses a collection of trainable filters to extract features from the data. Data is sampled by pooling layers down by taking the maximum or average value across a predetermined window size.

Despite the ability of CNN to extract features automatically due to the existence of convolutional layers, the features that were extracted by the LBP, HOG, and Haralick

methods were relied upon after passing through a GWO. Algorithm 4.9 show how to train a convolutional neural network (CNN) using training data and feature data.

Algorithm 4.9: Train CNN Classifier.

Input: GWOFeatures, AugmentedDataset

Output: CNNmodel

Step 1. Load the GWOFeatures and TrainingData.

Step 2. Split the data into Training and Validation sets.

```
[X_train, X_val, y_train, y_val] = train_test_split(GWOFeatures,
        AugmentedDataset, test_size=0.3);
```

Step 3. Define the CNN architecture.

```
layers = [ imageInputLayer([256 256 3])
          convolution2dLayer(3,8,'Padding','same')
          batchNormalizationLayer
          reluLayer
          maxPooling2dLayer(2,'Stride',2)
          convolution2dLayer(3,16,'Padding','same')
          batchNormalizationLayer
          reluLayer
          maxPooling2dLayer(2,'Stride',2)
          convolution2dLayer(3,32,'Padding','same')
          batchNormalizationLayer
          reluLayer
          fullyConnectedLayer(10)
          softmaxLayer
          classificationLayer];
```

Algorithm 4.9: Train CNN Classifier "Algorithms Continued".

Step 4. Set the training options.

```
options = trainingOptions('sgdm', 'InitialLearnRate',0.01, ...
  'MaxEpochs',10, 'Shuffle','every-epoch', ...
  'ValidationData', {X_val,y_val}, 'Verbose',false, ...
  'Plots','training-progress');
```

Step 5. Train the CNN model.

```
CNNmodel = trainNetwork(X_train,y_train,layers,options);
```

Return CNNmodel.

Step 6. Return CNNmodel

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 CHAPTER INTRODUCTION

Plant diseases are a common issue affecting agriculture, horticulture, and forestry. They can be brought on by a number of things, including pathogens (parasites, viruses, fungus, and bacteria), the environment, and inappropriate care. Symptoms of plant diseases can include leaf spots, wilting, stunted growth, yellowing, and death. Use adequate cultural techniques, such as appropriate irrigation, crop rotation fertilizing, and preventing overcrowding of plants, to prevent and control plant diseases. When necessary, chemical, and biological control methods, such as fungicides and biological control agents, can be used to treat infected plants. In order to identify and treat plant diseases, information technology is crucial. Images of sick plants may be analysed using ML algorithms and computer vision techniques to find signs and recognize the illness. This is just one example of how it is utilized. However, in this work the CNN, KNN, and SVM algorithms are used to detect several plant diseases using images processing techniques. Three methods were used to extract plant leaf's images features which are LBP, HOG and Haralick texture features, the most powerful features are selected using Grey Wolf Optimization (GWO).

5.2 SYSTEM REQUIREMENTS

The suggested system was created in MATLAB R2021a and implemented on a personal computer that met the requirements listed below: Core TM i7 H, 12 Gen CPU, 32 GB of RAM, Asus GeForce RTX 3060, 16 GB VGA, 1 TB of NVMe SSD storage, and Microsoft Windows 11 Pro, x64 operating system.

5.3 DATASET DESCRIPTION

utilizing a publicly available dataset called Plant Village that includes 54,306 pictures of healthy and sick plant leaves., as displayed in tables 5.1 and 5.2 for pepper plant images, 5.3 and 5.4 for potato plant images, and 5.5 and 5.6 for tomato plant images, along with histogram samples of all images. These pictures are all offered in RGB format.

Table 5.1: Pepper-Plant Dataset Classes.

#	**Class**	**Label**
0	Pepper__bell___Bacterial__spot	C1
1	Pepper__bell___healthy	C2

Table 5.2: Pepper-Plant Class Sample Images with Its Histogram.

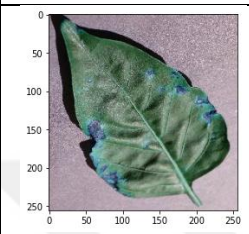
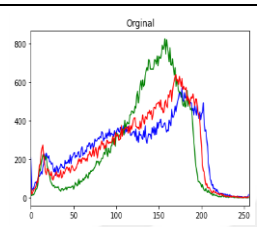
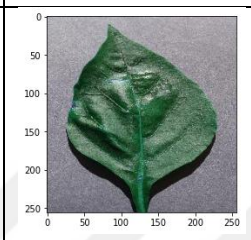
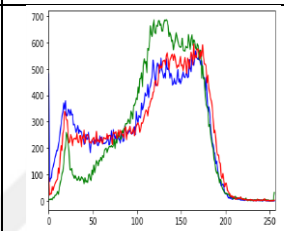
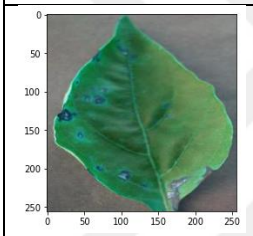
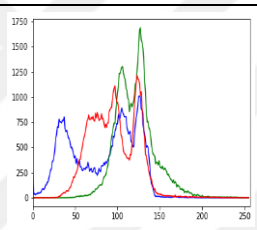
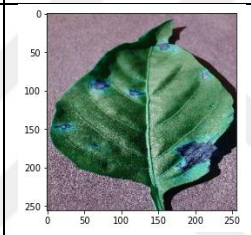
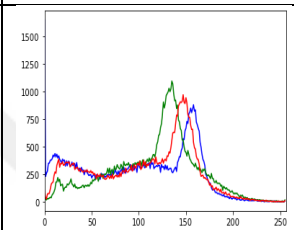
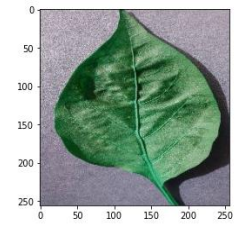
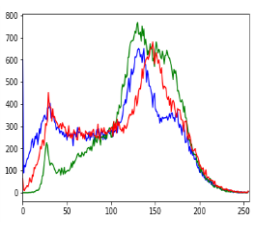
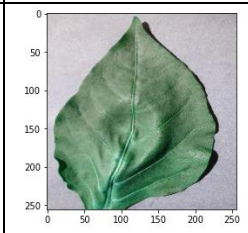
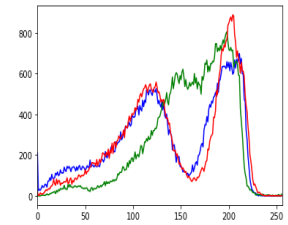
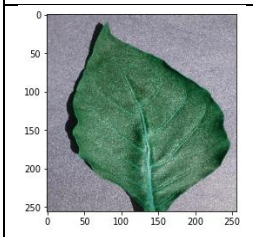
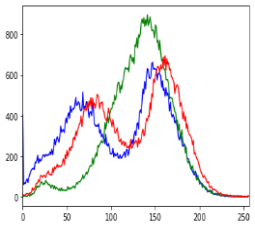
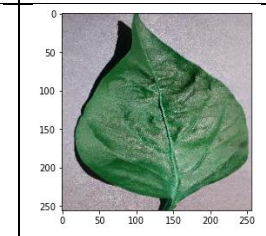
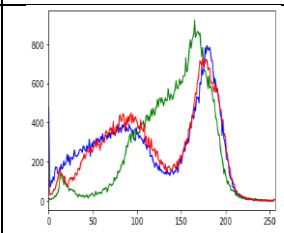
Class	Simple Image	Hist	Simple Image	Hist
C1				
				
C2				
				

Table 5.3: Potato-Plant Dataset Classes.

1	Potato___Early_blight	C1
2	Potato___healthy	C2
3	Potato___Late_blight	C3

Table 5.4: Potato-Plant Class Sample Images with Its Histogram.

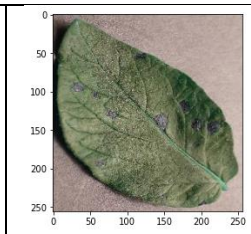
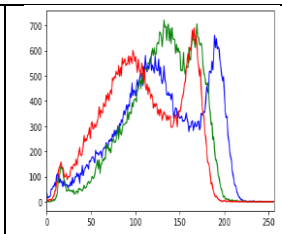
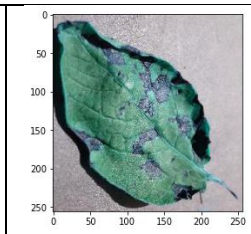
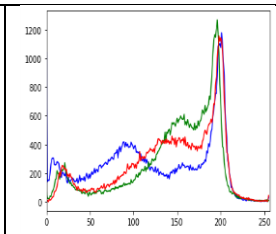
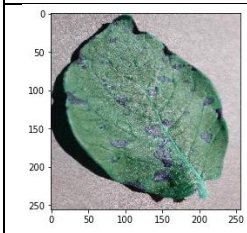
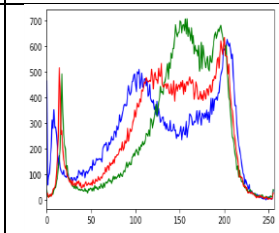
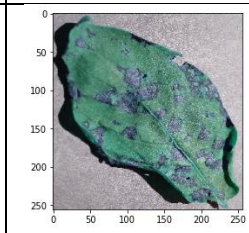
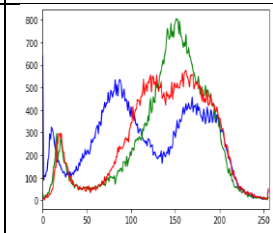
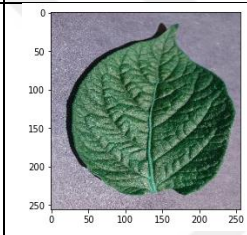
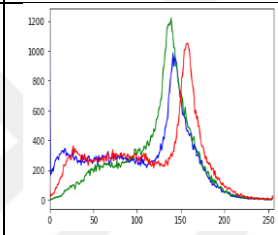
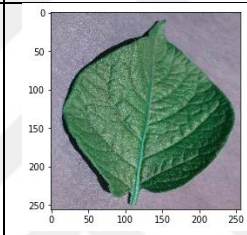
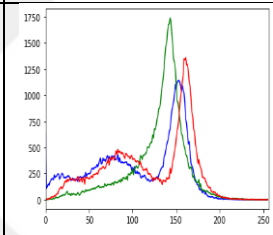
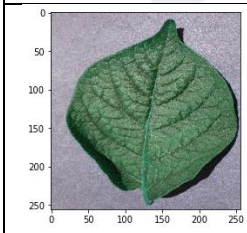
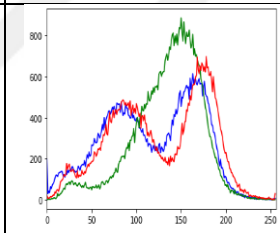
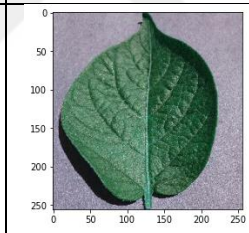
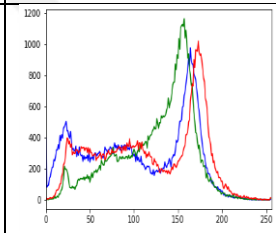
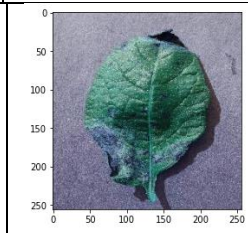
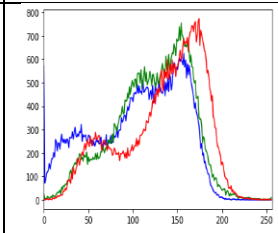
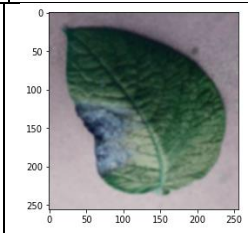
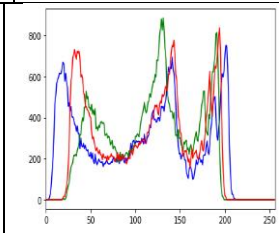
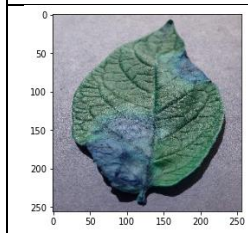
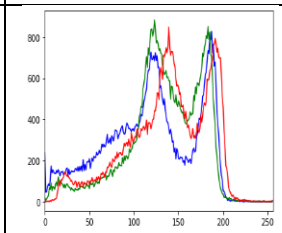
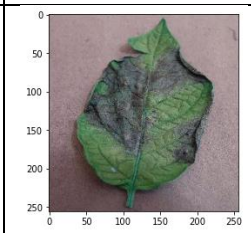
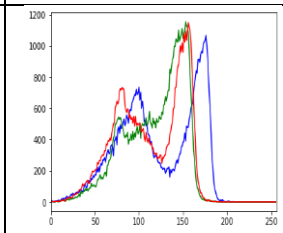
C1				
				
C2				
				
C3				
				

Table 5.5: Tomato -Plant Dataset Classes.

1	Tomato__Target_Spot	C1
2	Tomato__Tomato_YellowLeaf__Curl_Virus	C2
3	Tomato_Bacterial_spot	C3
4	Tomato_Early_blight	C4
5	Tomato_healthy	C5
6	Tomato_Late_blight	C6
7	Tomato_Leaf_Mold	C7
8	Tomato_Septoria_leaf_spot	C8
9	Tomato_Spider_mites_Two_spotted_spider_mite	C9

Table 5.6: Tomato-Plant Dataset Classes.

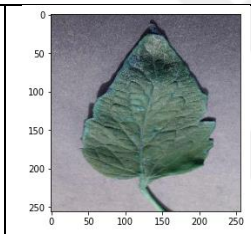
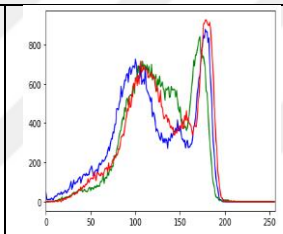
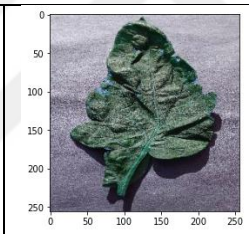
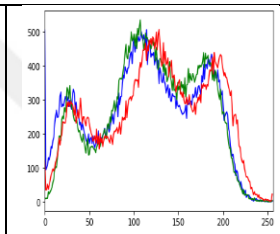
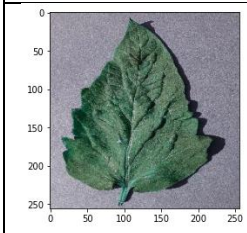
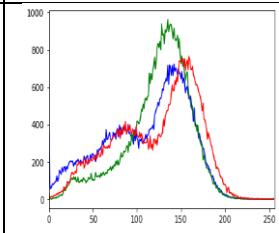
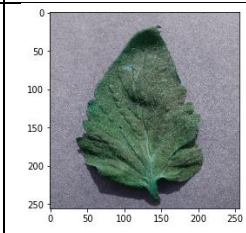
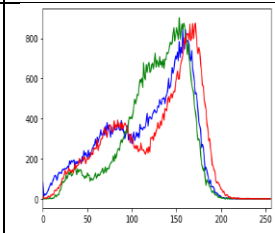
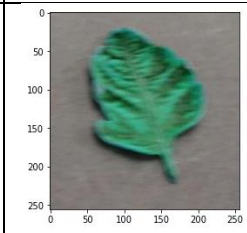
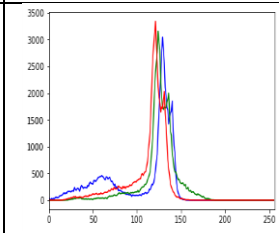
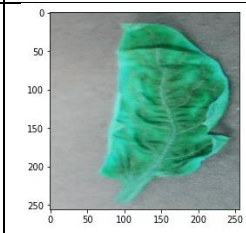
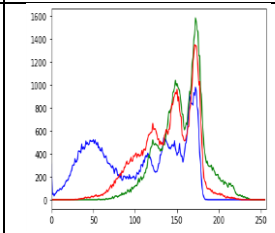
C1				
				
C2				

Table 5.6: Tomato-Plant Dataset Classes" Tables Continued".

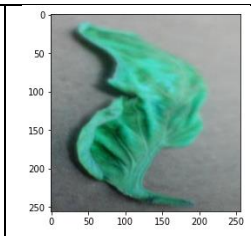
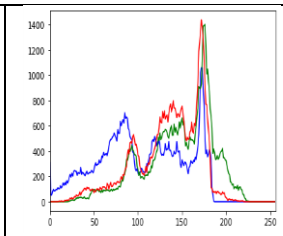
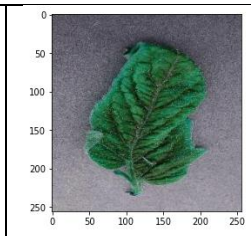
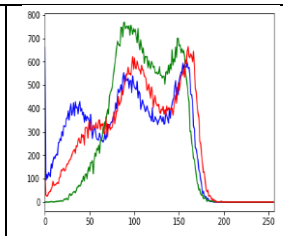
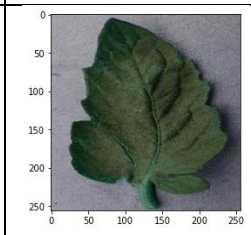
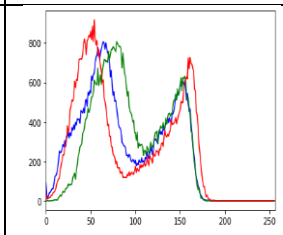
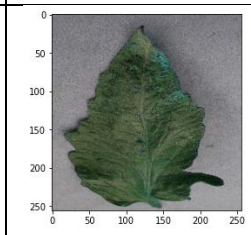
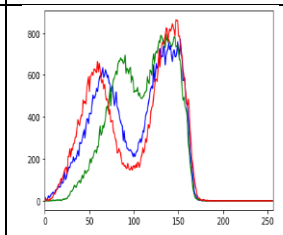
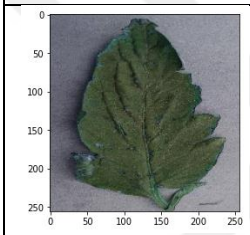
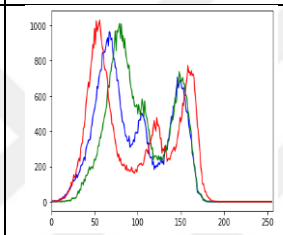
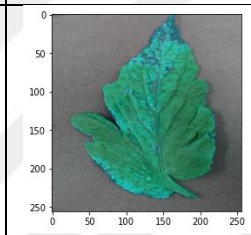
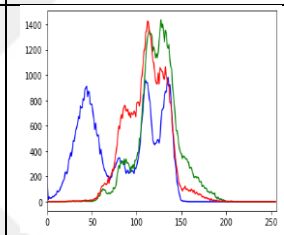
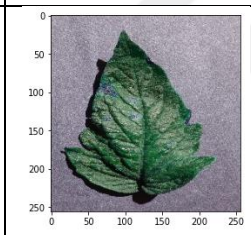
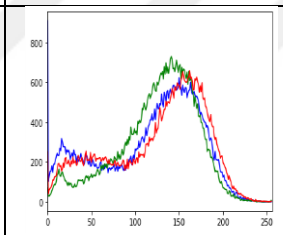
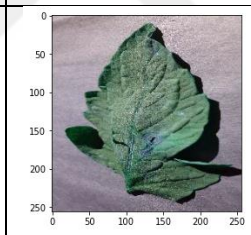
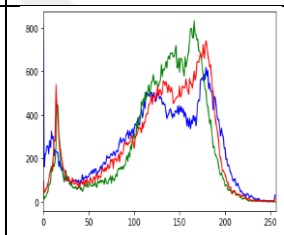
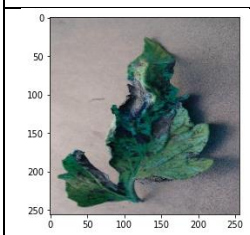
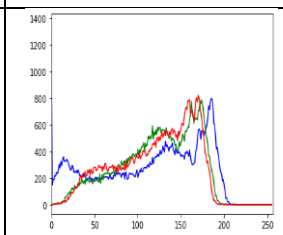
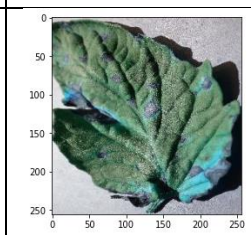
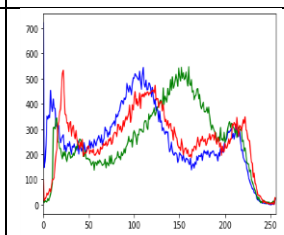
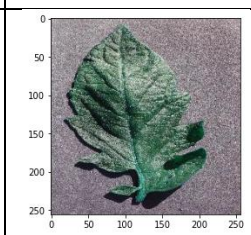
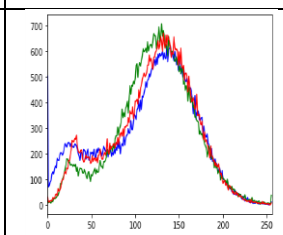
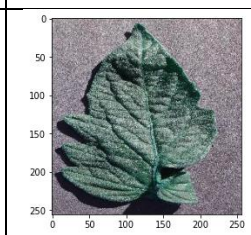
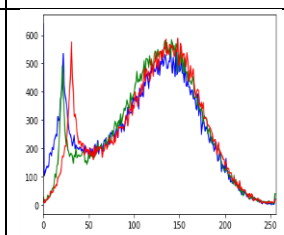
				
C3				
				
C4				
				
C5				

Table 5.6: Tomato-Plant Dataset Classes" Tables Continued".

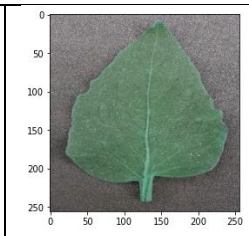
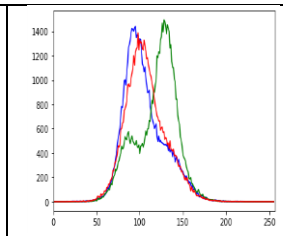
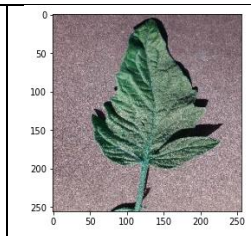
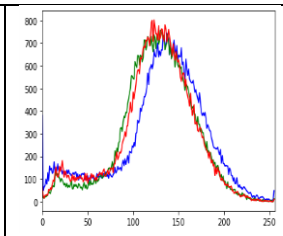
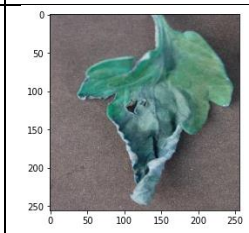
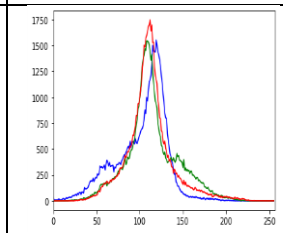
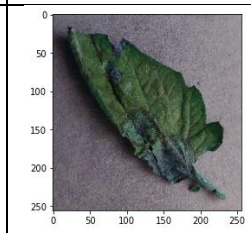
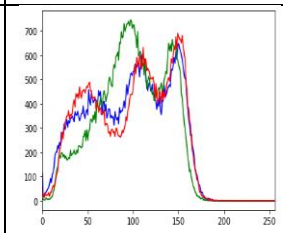
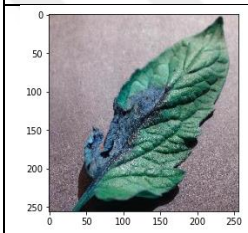
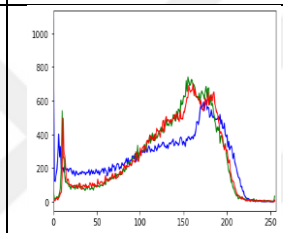
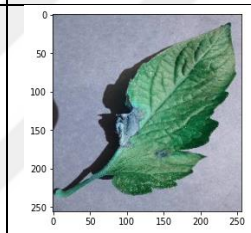
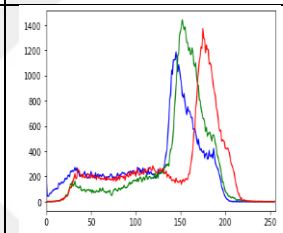
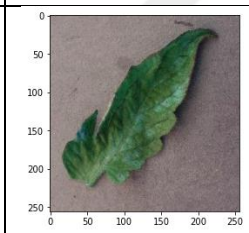
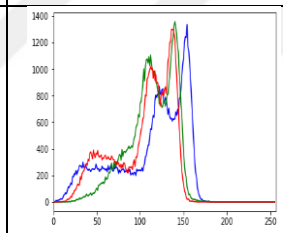
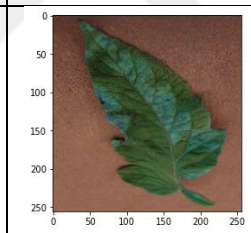
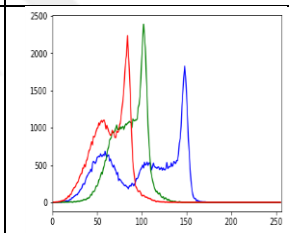
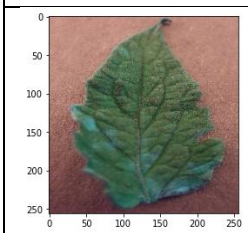
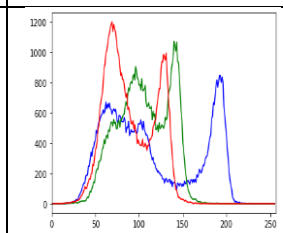
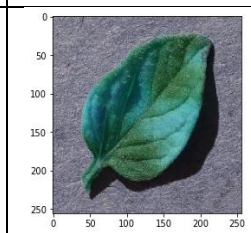
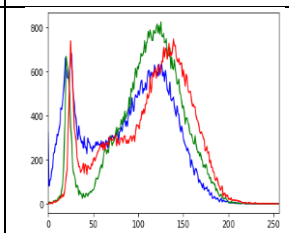
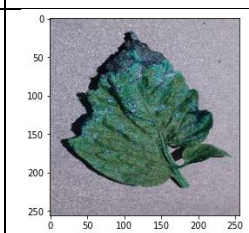
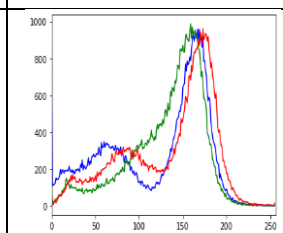
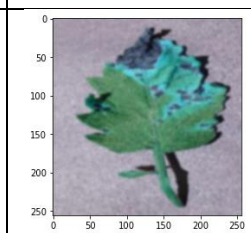
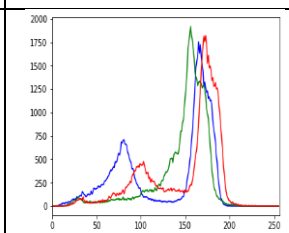
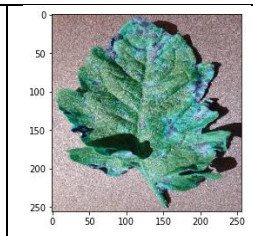
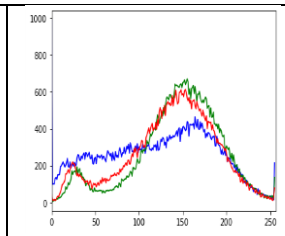
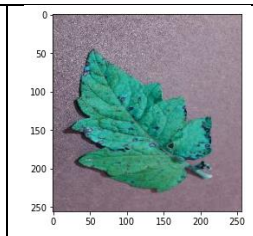
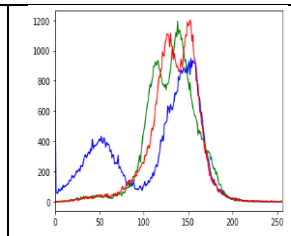
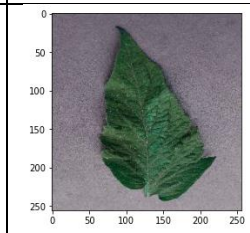
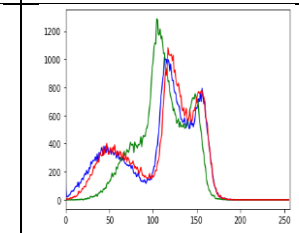
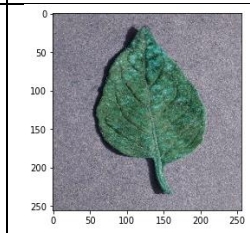
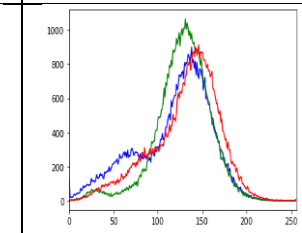
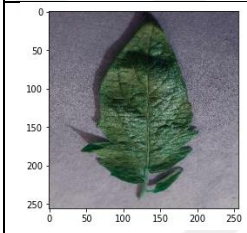
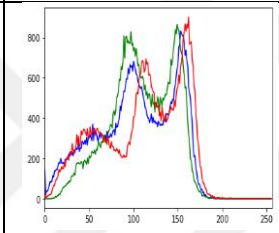
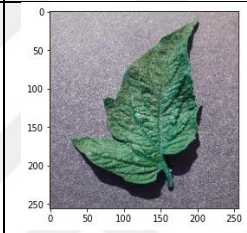
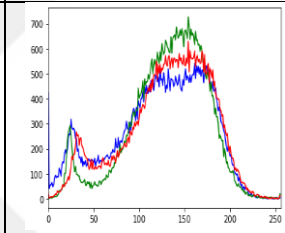
				
C6				
				
C7				
				
C8				

Table 5.6: Tomato-Plant Dataset Classes" Tables Continued".

				
C9				
				

5.4 SYSTEM IMPLEMENATION

Data pre-processing, feature extraction, feature selection, and KNN and CNN algorithms training are the four stages of this phase.

5.4.1 Data Pre-Processing

Image resizing, and data augmentation are steps in the data analysis process known as data pre-processing. This process is a crucial stage in data analysis because it makes sure the data is correct, comprehensive, and consistent and that it produces findings that are useful.

a. Image resizes.
When loading the dataset all images are going to be resized. During the loading process, every image in the dataset will be reduced in size to 256 pixels wide by 256 pixels high.

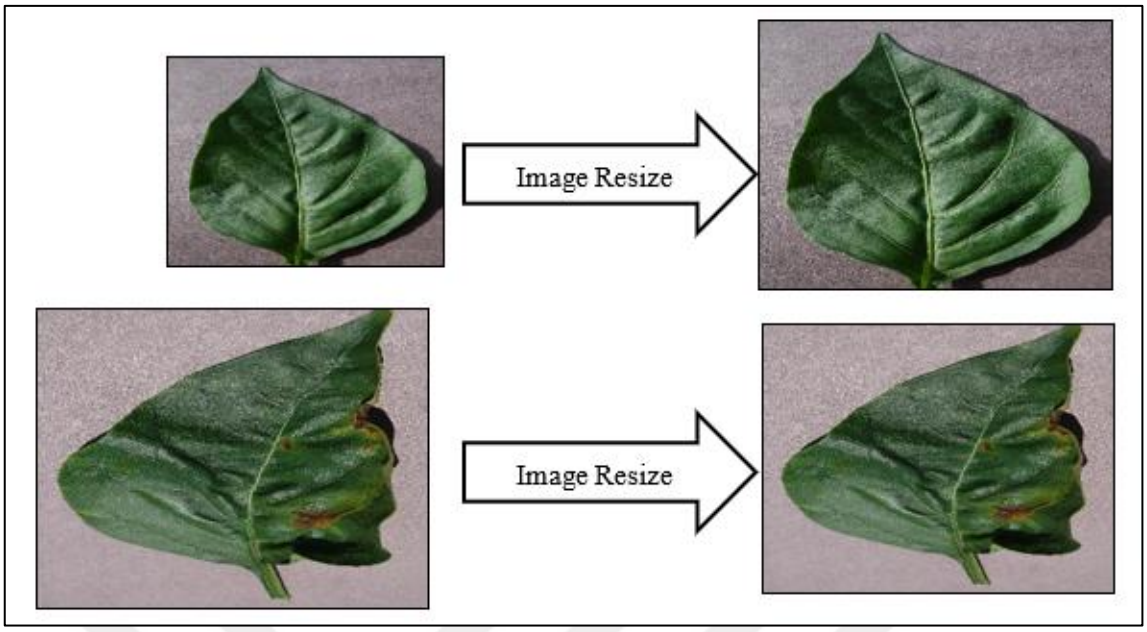


Figure 5.1: Image Resizing of Pepper Image Dataset.

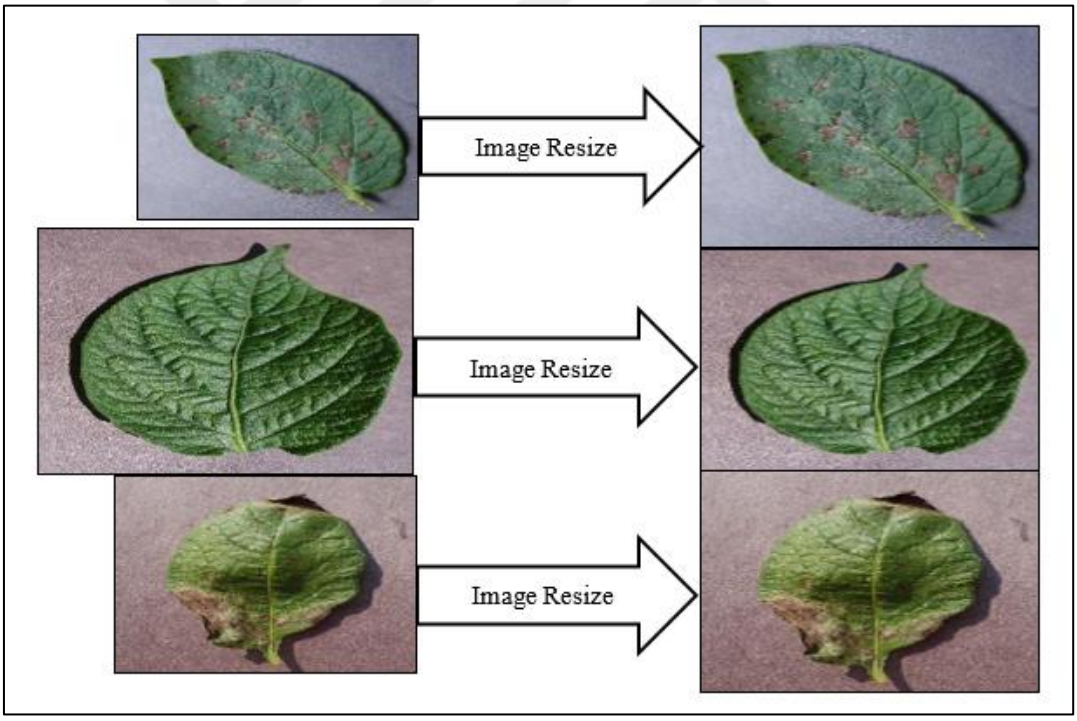


Figure 5.2: Image Resizing of Potato Image Dataset.

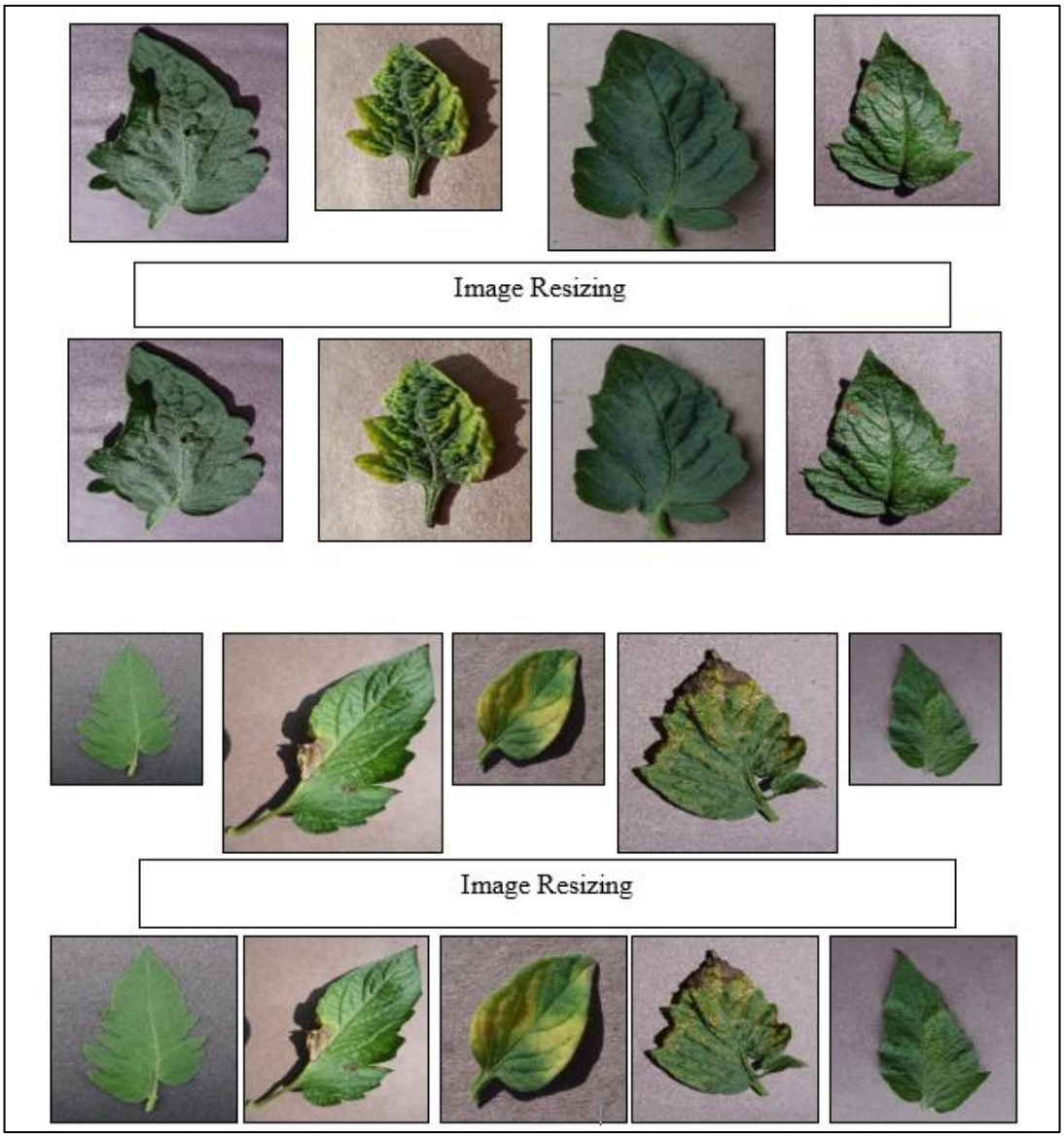


Figure 5.3: Image Resizing of Tomato Image Dataset.

b. Data Augmentation

A method used in ML and computer vision to expand the quantity and variety of a training dataset is known as data augmentation. Reduced overfitting, improved model generalization, and increased model resilience to changes in the data are the three objectives of data augmentation. Three transformation functions—X-reflection, Y-reflection, and Rotation—were used to enhance the quantity of data for each class in the dataset; Table 5.7 displays the results for each class.

Table 5.7: Augmentation Results.

Class	Original Size	Augmented Size
Pepper	2455	7,365
Potato	2152	6,456
Tomato	16,013	48,039
Total	20,620	61,851

5.4.2 Feature Extraction

The process of feature extraction in ML involves taking the pertinent data from the raw data and turning it into a collection of features that can be used to train a model. The objective of feature extraction is to convert unstructured data into a set of features that are instructive and pertinent to the issue at hand. In this study, the features from the dataset photos were extracted using three algorithms: LBP, HOG and Haralick texture features. Table 5.8 show the number of features that extracted from each class.

Table 5.8: Feature Extraction Results.

Method	Number of Features
LBP	59
HOG	36
Haralick	14
Total	**109**

a. Local Binary Patterns Features

By using LBP technique, features may be taken out of a picture. Each pixel in a picture is compared to its surrounding pixels, with the outcome being encoded as a binary integer. LBP features may be applied to applications including texture categorization, object identification, and face recognition.

Table 5.9: Features Sample Extracted by LBP.

1	2	3	4	5	6
0.42009	0.06385	0.01756	0.07130	0.01775	0.06663
7	8	9	10	11	12
0.01788	0.07449	0.01852	0.02105	0.01926	0.02060
13	14	15	16	17	18
0.02076	0.02061	0.01983	0.01967	0.02083	0.02626
19	20	21	22	23	24
0.01785	0.02432	0.01883	0.02562	0.01752	0.02422
25	26	27	28	29	30
0.01602	0.02264	0.02500	0.02794	0.02219	0.02137
31	32	33	34	35	36
0.02116	0.01921	0.02062	0.02108	0.02410	0.02112
37	38	39	40	41	42
0.02533	0.02016	0.02313	0.01802	0.02397	0.02386
43	44	45	46	47	48
0.02331	0.02142	0.02413	0.02324	0.02186	0.02282
49	50	51	52	53	54
0.02434	0.02111	0.06596	0.02117	0.06896	0.01772
55	56	57	58	59	
0.06783	0.02019	0.06721	0.49325	0.72149	

b. Histogram of oriented gradients Features

A feature descriptor that is often utilized in computer vision and image processing is HOG. To do this, a picture is divided into tiny cells, the gradient magnitude and direction for each pixel in each cell are determined, and a histogram of gradient orientations for

each cell is then created. Using the generated HOG characteristics, applications like item identification and pedestrian tracking are possible.

Table 5.10: Features Sample Extracted by HOG.

1	2	3	4	5	6
0.10281	0.09462	0.12609	0.10716	0.12796	0.18860
7	8	9	10	11	12
0.15549	0.13962	0.13690	0.13394	0.12349	0.07036
13	14	15	16	17	18
0.11914	0.19968	0.19979	0.15876	0.14776	0.12815
19	20	21	22	23	24
0.21564	0.21564	0.17132	0.10852	0.09996	0.21564
25	26	27	28	29	30
0.21564	0.21564	0.13902	0.21564	0.21564	0.16022
31	32	33	34	35	36
0.11962	0.18358	0.21564	0.21442	0.21564	0.18590

c. Haralick texture features

Haralick texture characteristics are a set of statistical features based on the grey-level co-occurrence matrix (GLCM) of an image that are used to characterize the texture of an image. The frequency of pixel pairings with certain grey-level values and spatial correlations is described by the GLCM, which is a matrix. The GLCM is used to compute the haralick texture characteristics, which may be applied to image segmentation, classification, and retrieval applications. Haralick texture characteristics include things like contrast, entropy, homogeneity, and correlation.

Table 5.11: Features Sample Extracted by GLCM.

1	2	3	4	5	6	7
0.07776	0.79992	0.84853	24.18436	0.75103	9.31046	57.97160
8	9	10	11	12	13	14
2.36711	4.22844	0.67907	0.93959	-0.36781	0.92272	0.89998

5.4.3 Feature Normalization

Feature normalization is the process of scaling and shifting the values of features (variables) in a dataset to a common scale, typically with zero mean and unit variance. This is done to ensure that each feature contributes equally to a model's predictions and to prevent one feature from dominating the others. Common normalization techniques include z-score normalization, min-max scaling, and log transformation.

Table 5.12: Normalization Sample Results.

1	2	3	4	5	6	7	8
0.77736	-1.07263	0.89697	-0.02697	0.90783	-0.94856	0.99767	0.20010
9	10	11	12	13	14	15	16
1.17590	-0.73193	-1.11864	-0.80810	-0.78686	-0.78052	-0.97239	-1.0131
17	18	19	20	21	22	23	24
-0.75316	-0.72521	-0.84026	-0.79448	-0.36044	-0.74150	-0.88442	-0.7833
25	26	27	28	29	30	31	32
-0.71243	-0.73548	-0.66329	-0.38877	-0.59992	-0.76669	-0.75588	-0.7027
33	34	35	36	37	38	39	40
-0.69180	-0.81751	-0.72350	-0.67663	-0.62469	-0.86253	-0.74094	-0.7979
41	42	43	44	45	46	47	48
-0.73685	-0.80282	-0.78706	-0.93181	-0.75265	-0.85650	-0.88156	-0.8137
49	50	51	52	53	54	55	56

Table 5.12: Normalization Sample Results "Tables Continued".

-0.73708	-0.10357	-0.96075	-0.03764	0.10658	-0.96195	-0.83906	-0.3185
57	58	59	60	61	62	63	64
-0.06058	0.10141	0.52751	-1.00346	-1.10792	-0.64867	-1.23738	-0.9548
65	66	67	68	69	70	71	72
0.52381	0.04083	-0.15230	-0.26328	-0.42447	-0.56672	-1.82318	-0.9617
73	74	75	76	77	78	79	80
0.49999	0.71525	0.04024	-0.03314	-0.48967	1.23916	1.18854	0.19486
81	82	83	84	85	86	87	88
-1.22667	-1.57817	1.07980	1.23823	1.32017	-0.26255	1.16550	1.17544
89	90	91	92	93	94	95	96
-0.01599	-0.99722	0.14593	0.99282	1.11691	1.24822	0.59267	-0.7613
97	98	99	100	101	102	103	104
0.47570	-0.14189	0.52251	-0.62191	0.52200	0.32271	0.85151	0.77280
105	106	107	108				
0.45850	0.60076	0.47748	0.00335				

5.4.4 Features Selection Using GWO

The process of choosing relevant features (predictors, variables) from a larger set of characteristics to incorporate into a ML model is known as features selection. Enhancing the model's interpretability, accuracy, and generalization is the aim of feature selection. By combining the results of LBP, HOG, and Haralick texture features, the Grey Wolf Optimizer (GWO) was used to select the most efficient features among these three types of features, and for three types of plant leaves, pepper, potato, and tomato, and for the two classes, three classes, and nine classes, respectively, results were extracted for a number Instants from (1000, 1200, 1400, 1600, 1800, and 1978) for the pepper , from (1200, 1400, 1600, 1800, 2000 and 2202) for the potato and from (7000, 8000, 9000,10000, 11000 and 12498) for

the tomato, taking into account that the number of iterations was 50. And the population was 100, with a 70%:30% training-to-testing ratio. Tables 5.13 ,5.14, 5.15 show the number of selected features for five Rounds.

Table 5.13: Feature Selection Results for Different Instants in Pepper**.**

	no. of features				
No. of Instants	**round1**	**round 2**	**round3**	**round 4**	**round 5**
1000	29	39	22	24	45
1200	28	72	42	22	37
1400	58	23	36	21	35
1600	34	86	27	28	23
1800	69	33	53	58	24
1978	80	21	22	47	23

Table 5.14: Feature Selection Results for Different Instants in Potato.

	no. of features				
No. of Instants	**round1**	**round 2**	**round3**	**round 4**	**round 5**
1200	29	21	41	66	40
1400	22	30	21	34	39
1600	26	49	57	21	26
1800	28	71	26	23	21
2000	78	26	68	30	45
2202	21	22	41	43	37

Table 5.15: Feature Selection Results for Different Instants in Tomato.

	no. of features				
No. of Instants	**round1**	**round 2**	**round3**	**round 4**	**round 5**
7000	38	80	29	21	27
8000	58	23	22	23	27
9000	34	21	24	34	39
10000	27	26	69	31	21
11000	31	21	21	28	57
12498	77	21	32	23	21

Result sample were extracted by taking different samples from the threshold (0.2, 0.25, 0.3, 0.35, 0.4, 0.45 and 5) with 1900 instants for the pepper ,2200 instants for the potato and 12000 instants for the tomato ,100 population, 0.3 threshold and the number of iterations being 50 iterations and with a training-to-testing ratio of 70:30%. Tables 5.16 ,5.17, 5.18 show the number of selected features for five Rounds.

Table 5.16: Feature Selection Results for Different Threshold in Pepper.

	no. of features				
threshold	**round1**	**round 2**	**round3**	**round 4**	**round 5**
0.2	73	27	26	24	23
0.25	33	28	21	46	32
0.3	37	22	26	28	61
0.35	23	22	47	24	34
0.4	27	44	58	21	67
0.45	44	21	25	65	61
0.5	35	34	52	23	49

Table 5.17: Feature Selection Results for Different Threshold in Potato.

	no. of features				
threshold	**round1**	**round 2**	**round3**	**round 4**	**round 5**
0.2	57	40	30	25	23
0.25	50	25	51	30	21
0.3	21	22	41	43	37
0.35	60	33	22	25	63
0.4	66	71	28	38	64
0.45	42	28	40	47	62
0.5	49	54	50	23	42

Table 5.18: Feature Selection Results for Different Threshold in Tomato.

	no. of features				
threshold	**round1**	**round 2**	**round3**	**round 4**	**round 5**
0.2	72	23	22	39	22
0.25	29	24	21	22	32
0.3	77	21	32	23	21
0.35	24	28	22	21	33
0.4	26	40	39	45	22
0.45	24	25	24	34	27
0.5	55	29	26	24	22

Other results were obtained by using different population samples (50, 75, 100, 125, 150, 175 and 200), 1900 instants for the pepper ,2200 instants for the potato and 12000 instants for the tomato, 0.3 threshold, 50 iterations, and a 70:30% training-to-testing ratio for the three plant types: pepper, potato, and tomato. Tables 5.19 ,5.20, 5.21 show the number of selected features for five Rounds.

Table 5.19: Feature Selection Results for Different Population in Pepper.

	no. of features				
pop size	**round1**	**round 2**	**round3**	**round 4**	**round 5**
50	48	44	35	58	67
75	35	33	27	52	38
100	68	80	25	38	27
125	27	23	21	43	28
150	48	29	23	35	25
175	37	30	29	40	21
200	35	87	21	47	35

Table 5.20: Feature Selection Results for Different Population in Potato.

	no. of features				
pop size	**round1**	**round 2**	**round3**	**round 4**	**round 5**
50	56	75	24	31	71
75	21	24	41	36	61
100	21	22	41	43	37
125	26	28	32	53	43
150	21	25	37	25	38
175	53	22	29	33	36
200	26	34	22	23	37

Table 5.21: Feature Selection Results for Different Population in Tomato.

	no. of features				
pop size	**round1**	**round 2**	**round3**	**round 4**	**round 5**
50	69	65	22	21	27
75	21	23	43	45	36
100	77	21	32	23	21
125	21	76	42	59	31
150	58	74	21	47	22
175	21	27	30	23	22
200	24	21	47	21	22

samples results were extracted by taking malty iterations (10, 20,30, 40, 50 and 60) with 1900 instants for the pepper ,2200 instants for the potato and 12000 instants for the tomato,100 population and with a training-to-testing ratio of 70:30%. Tables 5.22 ,5.23, 5.24 show the number of selected features for five Rounds.

Table 5.22: Feature Selection Results for Different Iteration in Pepper.

	no. of features				
Max Iteration	**round1**	**round 2**	**round3**	**round 4**	**round 5**
10	39	36	32	51	31
20	44	67	57	21	29
30	36	22	23	65	27
40	66	27	22	24	22
50	68	80	25	38	27
60	78	51	26	36	75

Table 5.23: Feature Selection Results for Different Iteration in Potato.

	no. of features				
Max Iteration	**round1**	**round 2**	**round3**	**round 4**	**round 5**
10	29	33	72	46	32
20	47	39	76	77	37
30	85	29	30	27	63
40	26	83	27	22	42
50	21	22	41	43	37
60	31	29	25	30	76

Table 5.24: Feature Selection Results for Different Iteration in Tomato.

	no. of features				
Max Iteration	**round1**	**round 2**	**round3**	**round 4**	**round 5**
10	25	37	23	71	39
20	79	24	27	23	26
30	21	22	24	21	29
40	21	25	30	22	78
50	21	23	22	21	24
60	21	52	25	68	32

samples results were extracted by taking different training-to testing ratio (90:10%, 80:20%, 70:30%, and 60:40%) with 1900 instants for the pepper ,2200 instants for the potato and 12000 instants for the tomato, 100 population and 50 Iterations. %. Tables 5.25 ,5.26, 5.27 show the number of selected features for five Rounds.

Table 5.25: Results of Feature Selection for Various Training-to-Testing Ratios in Pepper.

		no. of features				
Training	**testing**	**round1**	**round 2**	**round3**	**round 4**	**round 5**
90%	**10%**	54	29	24	80	56
80%	**20%**	25	23	21	23	37
70%	**30%**	26	45	24	38	25
60%	**40%**	23	21	25	26	35

Table 5.26: Results of Feature Selection for Various Training-to-Testing Ratios in Potato.

		no. of features				
Training	**testing**	**round1**	**round 2**	**round3**	**round 4**	**round 5**
90%	**10%**	33	38	44	59	32
80%	**20%**	71	42	39	78	22
70%	**30%**	21	22	41	43	37
60%	**40%**	31	35	65	78	32

Table 5.27: Results of Feature Selection for Various Training-to-Testing Ratios in Tomato.

		no. of features				
Training	**testing**	**round1**	**round 2**	**round3**	**round 4**	**round 5**
90%	**10%**	46	21	22	78	23
80%	**20%**	24	47	37	23	21
70%	**30%**	39	23	26	21	29
60%	**40%**	22	34	22	23	67

5.5 TRAINING RESULTS

At this level, the features data and training data are ready to be used in training phase, three models SVM, CNN, and KNN are trained to detect 14 classes of plaint diseases.

5.5.1 Support Vector Machine (SVM) Results

The SVM have been trained to classify Pepper Plant diseases, where we take 1000 samples, training 90% with several K-fold (5,10,15 and 20), accuracy was calculated for the minimum, maximum and the mean. Tables 5.28, 5.29, 5.30, 5.31 show the details of SVM results with average training.

Table 5.28: Accuracy of Support Vector Machine (SVM) Findings.

K-Fold=5			
Round#	**Min**	**Mean**	**MAX**
1	87.00%	88.10%	90.50%
2	86.50%	88.80%	92.00%
3	88.00%	89.40%	91.00%
4	85.50%	87.00%	89.00%
5	83.50%	87.80%	93.00%

Table 5.29: Accuracy of Support Vector Machine (SVM) Findings.

K-Fold=10			
Round#	**Min**	**Mean**	**MAX**
1	86.00%	88.65%	92.00%
2	87.00%	88.95%	91.50%
3	85.00%	88.55%	93.00%

Table 5.29: Accuracy of Support Vector Machine (SVM) Findings “Tables Continued”.

4	85.00%	88.55%	93.00%
5	85.00%	88.10%	90.00%

Table 5.30: Accuracy of Support Vector Machine (SVM) Findings.

K-Fold=15			
Round#	**Min**	**Mean**	**MAX**
1	86.00%	89.03%	93.00%
2	87.00%	89.30%	92.00%
3	86.00%	88.80%	92.50%
4	84.00%	89.07%	93.50%
5	87.00%	89.80%	92.50%

Table 5.31: Accuracy of Support Vector Machine (SVM) Findings.

K-Fold=20			
Round#	**Min**	**Mean**	**MAX**
1	84.50%	89.88%	94.00%
2	85.50%	89.03%	92.50%
3	82.50%	89.15%	93.00%
4	85.00%	88.68%	91.00%
5	85.00%	88.68%	91.00%

Other results were extracted when the k-fold was at 10, the training was 80 % and the number of samples (1000, 1400, 1800, 1978). Tables 5.32, 5.33, 5.34, 4.35 show the details of SVM results with

Table 5.32: Accuracy of Support Vector Machine (SVM) Findings.

Sample size 1000			
Round#	**Min**	**Mean**	**MAX**
1	86.00%	88.65%	92.00%
2	87.00%	88.95%	91.50%
3	85.00%	88.55%	93.00%
4	85.00%	88.55%	93.00%
5	85.00%	88.10%	90.00%

Table 5.33: Accuracy of Support Vector Machine (SVM) Findings.

Sample size 1400			
Round#	**Min**	**Mean**	**MAX**
1	84.00%	90.04%	94.29%
2	84.00%	90.08%	93.00%
3	84.00%	89.99%	93.57%
4	84.00%	89.97%	93.93%
5	84.00%	90.16%	93.93%

Table 4.34: Accuracy of Support Vector Machine (SVM) Findings.

Sample size 1800			
Round#	**Min**	**Mean**	**MAX**
1	84.00%	90.47%	95.31%
2	84.00%	90.61%	94.06%
3	90.28%	92.53%	95.00%
4	91.39%	92.86%	95.00%
5	89.72%	93.14%	95.28%

Table 5.35: Accuracy of Support Vector Machine (SVM) Findings.

Sample size 1978			
Round#	**Min**	**Mean**	**MAX**
1	84.00%	91.17%	94.19%
2	84.00%	91.12%	95.71%
3	84.00%	91.17%	95.96%
4	84.00%	91.09%	94.95%
5	84.00%	90.99%	94.19%

Other results were extracted when the k-fold was at 10 and the number of samples was 1000 and the training was (90%, 80% 70% and 60%). Tables 5.36, 5.37, 5.38, 4.39 show the details of SVM results with

Table 5.36: Accuracy of Support Vector Machine (SVM) Findings.

Training 90%			
Round#	**Min**	**Mean**	**MAX**
1	86.00%	89.40%	94.00%
2	85.00%	89.20%	93.00%
3	81.00%	89.60%	97.00%
4	85.00%	91.50%	98.00%
5	86.00%	90.50%	94.00%

Table 5.37: Accuracy of Support Vector Machine (SVM) Findings.

Training 80%			
Round#	**Min**	**Mean**	**MAX**
1	82.50%	89.25%	93.00%
2	82.50%	88.40%	92.00%
3	82.50%	88.40%	92.00%
4	85.50%	88.55%	92.00%
5	86.00%	89.40%	92.50%

Table 5.38: Accuracy of Support Vector Machine (SVM) Findings.

Training 70%			
Round#	**Min**	**Mean**	**MAX**
1	85.67%	88.47%	91.67%
2	85.00%	88.23%	90.33%
3	85.00%	88.60%	91.00%
4	86.00%	88.33%	91.00%
5	83.67%	88.50%	92.00%

Table 5.39: Accuracy of Support Vector Machine (SVM) Findings.

Training 60%			
Round#	**Min**	**Mean**	**MAX**
1	82.00%	87.30%	89.75%
2	83.75%	86.50%	88.00%
3	85.25%	88.08%	90.25%
4	83.25%	86.53%	89.00%
5	85.50%	86.83%	88.50%

5.5.2 K-Nearest Neighbors (KNN) Results

The K-Nearest Neighbours (KNN) was trained to classify Potato and Tomato Plants diseases, where we take (1700 potato sample and 9000 tomato samples), training rete was 80% with several K-fold (5,10,15 and 20), accuracy was calculated for the minimum, maximum and the mean. Tables 5.40, 5.41, 5.42 and 5.43 show the details of KNN-model results with average training accuracy.

Table 5.40: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

K-Fold=5				**K-Fold=5**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	83.33%	86.53%	89.33%	**1**	76.90%	77.52%	78.50%
2	82.33%	85.07%	89.00%	**2**	75.00%	77.20%	78.90%
3	83.33%	84.73%	86.33%	**3**	76.60%	77.40%	79.70%
4	80.33%	84.27%	86.67%	**4**	76.00%	78.54%	82.20%
5	82.67%	84.80%	87.00%	**5**	76.30%	77.80%	80.10%

Table 5.41: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

K-Fold=10				**K-Fold=10**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	81.00%	85.53%	89.33%	**1**	73.00%	77.37%	79.40%
2	77.67%	85.23%	89.00%	**2**	73.80%	77.50%	80.10%
3	82.67%	84.87%	87.67%	**3**	74.80%	77.61%	79.70%
4	83.67%	86.23%	88.67%	**4**	74.30%	77.71%	81.00%
5	81.00%	85.47%	88.67%	**5**	76.00%	78.18%	81.30%

Table 5.42: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

K-Fold=15				K-Fold=15			
Potato				Tomato			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	81.33%	84.38%	89.00%	**1**	74.90%	77.49%	80.90%
2	82.00%	85.78%	89.33%	**2**	73.40%	77.86%	80.10%
3	80.67%	85.00%	87.67%	**3**	74.10%	77.85%	81.00%
4	81.00%	85.38%	88.00%	**4**	74.10%	76.93%	80.30%
5	81.33%	84.82%	87.33%	**5**	75.60%	77.38%	78.40%

Table 5.43: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

K-Fold=20				K-Fold=20			
Potato				Tomato			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	79.67%	83.97%	87.00%	**1**	75.50%	77.56%	79.60%
2	81.33%	85.45%	89.33%	**2**	73.80%	77.42%	80.60%
3	79.00%	84.32%	87.67%	**3**	74.40%	77.62%	79.90%
4	82.00%	85.87%	90.67%	**4**	75.70%	77.88%	79.70%
5	80.67%	85.12%	90.00%	**5**	76.30%	78.22%	80.70%

Other results samples were taken by making the k-fold 10, training rate 80% and several Potato samples (1500, 1700, 1900 and 2202) and Tomato samples (5000, 7000, 9000 and 12000) accuracy was calculated for the minimum, maximum and the mean. Tables 5.40, 5.41, 5.42, 5.43 show the details of KNN-model results with average training accuracy.

Table 5.44: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

no. of samples =1500				**no. of samples =5000**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	82.00%	85.40%	88.67%	**1**	74.10%	77.20%	79.50%
2	81.00%	84.27%	87.00%	**2**	77.00%	79.24%	81.80%
3	83.00%	85.70%	89.00%	**3**	77.70%	79.03%	80.70%
4	79.67%	85.37%	88.67%	**4**	75.00%	77.36%	81.00%
5	80.67%	84.87%	88.33%	**5**	74.10%	77.20%	79.50%

Table 5.45: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

no. of samples =1700				**no. of samples =7000**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	82.00%	85.40%	88.67%	**1**	75.80%	77.26%	78.10%
2	81.00%	84.27%	87.00%	**2**	74.10%	77.20%	79.50%
3	83.00%	85.70%	89.00%	**3**	77.00%	79.24%	81.80%
4	79.67%	85.37%	88.67%	**4**	77.70%	79.03%	80.70%
5	80.67%	84.87%	88.33%	**5**	77.36%	81.00%	80.67%

Table 5.46: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

no. of samples =1900				**no. of samples =9000**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	84.21%	87.74%	92.63%	**1**	87.78%	89.61%	91.11%
2	85.00%	87.03%	91.32%	**2**	88.11%	89.21%	90.28%
3	84.21%	87.47%	88.95%	**3**	88.56%	89.84%	90.94%
4	83.95%	86.47%	88.95%	**4**	88.44%	90.11%	91.33%
5	85.00%	87.45%	89.47%	**5**	87.78%	89.61%	91.11%

Table 5.47: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

no. of samples =2202				**no. of samples =12000**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	85.23%	89.02%	92.05%	**1**	94.04%	94.87%	95.38%
2	84.77%	88.32%	91.14%	**2**	94.33%	94.92%	95.58%
3	85.23%	87.98%	90.45%	**3**	93.96%	94.64%	95.38%
4	85.45%	88.09%	90.68%	**4**	93.75%	94.95%	96.33%
5	86.59%	89.30%	91.14%	**5**	94.67%	95.18%	95.83%

Other results samples were taken by making the k-fold 10, with 1700 potato sample, 9000 tomato sample and several training rate (90%, 80% ,70% and 60%) accuracy was calculated for the minimum, maximum and the mean. Tables 5.48, 5.49, 5.50 and 5.51 show the details of KNN-model results with average training accuracy.

Table 5.48: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

training size 90%				**training size 90%**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	77.65%	85.82%	91.18%	**1**	91.22%	92.00%	93.00%
2	81.76%	87.12%	90.00%	**2**	91.11%	92.96%	94.33%
3	80.00%	86.12%	91.18%	**3**	90.67%	92.10%	93.89%
4	80.59%	86.41%	91.18%	**4**	90.67%	91.76%	93.44%
5	82.94%	86.29%	89.41%	**5**	90.56%	91.87%	92.89%

Table 5.49: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

training size 80%				**training size 80%**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	85.29%	86.44%	88.82%	**1**	88.33%	89.63%	91.33%
2	83.82%	86.85%	89.71%	**2**	87.72%	89.67%	91.94%
3	83.53%	86.03%	88.53%	**3**	87.83%	89.54%	90.89%
4	84.71%	86.71%	88.82%	**4**	88.56%	89.59%	90.39%
5	83.24%	86.29%	88.82%	**5**	88.11%	89.64%	91.22%

Table 5.50: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

training size 70%				**training size 70%**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	81.96%	85.41%	88.24%	**1**	85.96%	86.76%	87.96%
2	83.33%	85.84%	89.61%	**2**	86.26%	87.16%	88.63%
3	82.94%	85.57%	88.43%	**3**	86.15%	87.03%	88.41%
4	81.96%	84.75%	87.25%	**4**	85.63%	86.61%	87.30%
5	83.33%	85.41%	86.47%	**5**	85.07%	86.34%	87.85%

Table 5.51: K-Nearest Neighbors (KNN) Results in Potato and Tomato.

training size 60%				**training size 60%**			
Potato				**Tomato**			
Round#	**Min**	**Mean**	**MAX**	**Round#**	**Min**	**Mean**	**MAX**
1	81.96%	85.41%	88.24%	**1**	85.96%	86.76%	87.96%
2	83.33%	85.84%	89.61%	**2**	86.26%	87.16%	88.63%
3	82.94%	85.57%	88.43%	**3**	86.15%	87.03%	88.41%
4	81.96%	84.75%	87.25%	**4**	85.63%	86.61%	87.30%
5	83.33%	85.41%	86.47%	**5**	85.07%	86.34%	87.85%

5.5.3 Convolution Neural Network (CNN) Results

Plant disease classification was taught to the Convolution Neural Network (CNN) in two types of pepper, three types of potato, and nine types of tomato, and those were classified at

several epochs (500, 600, 700, 800, 900, and 1000). Tables 5.52, 5.53, and 5.54 show the details of CNN-model results with training accuracy for the three plants types.

Table 5.52: Convolution Neural Network (CNN) Results in Pepper.

Epoch	Max Iteration	Iteration per epoch	Elapsed time	Accuracy
500	7500	15	4min, 27sec	99.70%
600	9000	15	5min, 16sec	99.80%
700	10500	15	6min, 1sec	99.80%
800	12000	15	7min, 3sec	98.60%
900	13500	15	7min, 32sec	96.70%
1000	15000	15	8min, 27sec	97.40%

For the pepper plants, as shown in the above table, when taking into account the minimum implementation time, the best accuracy results (99.80%) we obtained were in this case 600 epoch, 9000 max-iteration in 5min, 16sec execution time. Knowing that CNN, in this case, classified 2 types of leaves, one is healthy and the other infected disease.

Table 4.53: Convolution Neural Network (CNN) Results in Potato.

Epoch	Max Iteration	Iteration per epoch	Elapsed time	Accuracy
500	7500	15	4min, 30sec	99.80%
600	9000	15	5min, 13sec	99.90%
700	10500	15	5min, 58sec	99.90%
800	12000	15	6min, 21sec	99.80%
900	13500	15	6min, 34sec	99.80%
1000	15000	15	8min, 29sec	99.90%

For the potato plants, as shown in the above table, when taking into account the minimum implementation time, the best accuracy results (99.90%) we obtained were in this case 600 epochs, 9000 max-iteration in 5min, 13sec execution time. Knowing that CNN, in this case,

classified 3 types of leaves, one of which is healthy and the rest infected with 2 different types of diseases.

Table 5.54: Convolution Neural Network (CNN) Results in Tomato.

Epoch	Max Iteration	Iteration per epoch	Elapsed time	Accuracy
500	7500	15	3min, 43sec	88.60%
600	9000	15	5min, 1sec	89.40%
700	10500	15	5min, 31sec	94.60%
800	12000	15	6min, 39sec	89.60%
900	13500	15	7min, 34sec	92.90%
1000	15000	15	8min,57sec	95.70%

For tomato plants, as shown in the table above, when considering the minimum execution time, the best accuracy results (95.70%) we obtained in this case were 1000 epochs, 15000 max-iteration in 8 min, 57 sec execution time. Knowing that CNN, in this case, classified 9 types of leaves, one of which is healthy and the rest infected with 8 different types of diseases.

According to CNN model the training reached a maximum of 7500 iterations by 15 iterations per epoch with a maximum epoch of 500 in the first case, and a maximum of 12000 iterations by 15 iterations per epoch with a maximum epoch of 800 in the second case .Figure 5.4 show the graphical results for the two cases of the CNN model training in pepper plants indicating that total accuracy has reached the highest level and losses have reached their lowest level.

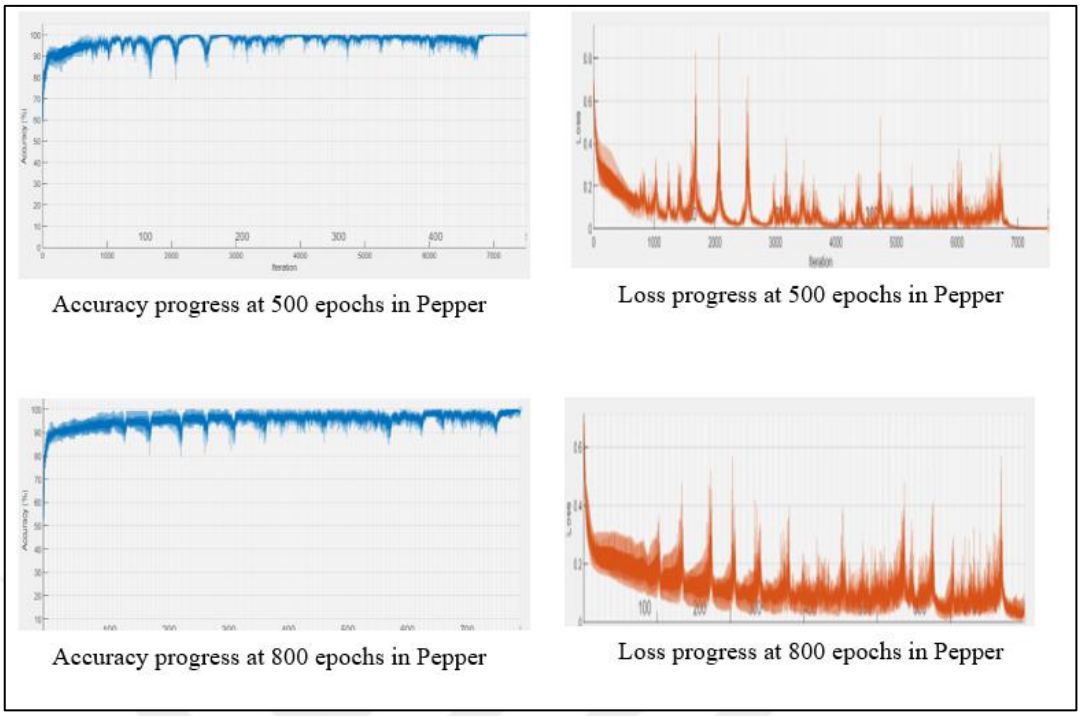


Figure 5.4: Accuracy and Loss Progress in CNN of Pepper Plant.

In potato plant and according to CNN model the training reached a maximum of 7500 iterations by 15 iterations per epoch with a maximum epochs of 500 in the first case, and a maximum of 12000 iterations by 15 iterations per epoch with a maximum epochs of 800 in the second case Figure 5.5 show the graphical results for the two cases of the CNN model training in potato plants indicating that total accuracy has reached the highest level and losses have reached their lowest level.

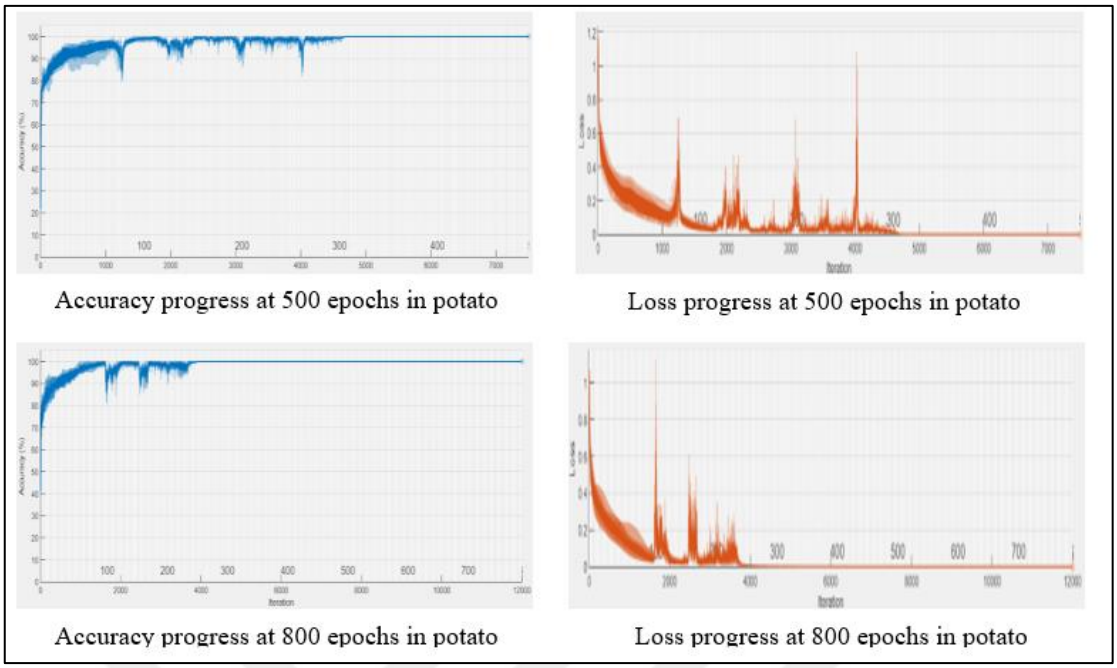


Figure 5.5: Accuracy and Loss Progress in CNN of Potato Plant.

In tomato plant and according to CNN model the training reached a maximum of 7500 iterations by 15 iterations per epoch with a maximum epochs of 500 in the first case, and a maximum of 12000 iterations by 15 iterations per epoch with a maximum epochs of 800 in the second case Figure 9show the graphical results for the two cases of the CNN model training in tomato plants indicating that total accuracy has reached the highest level and losses have reached their lowest level.

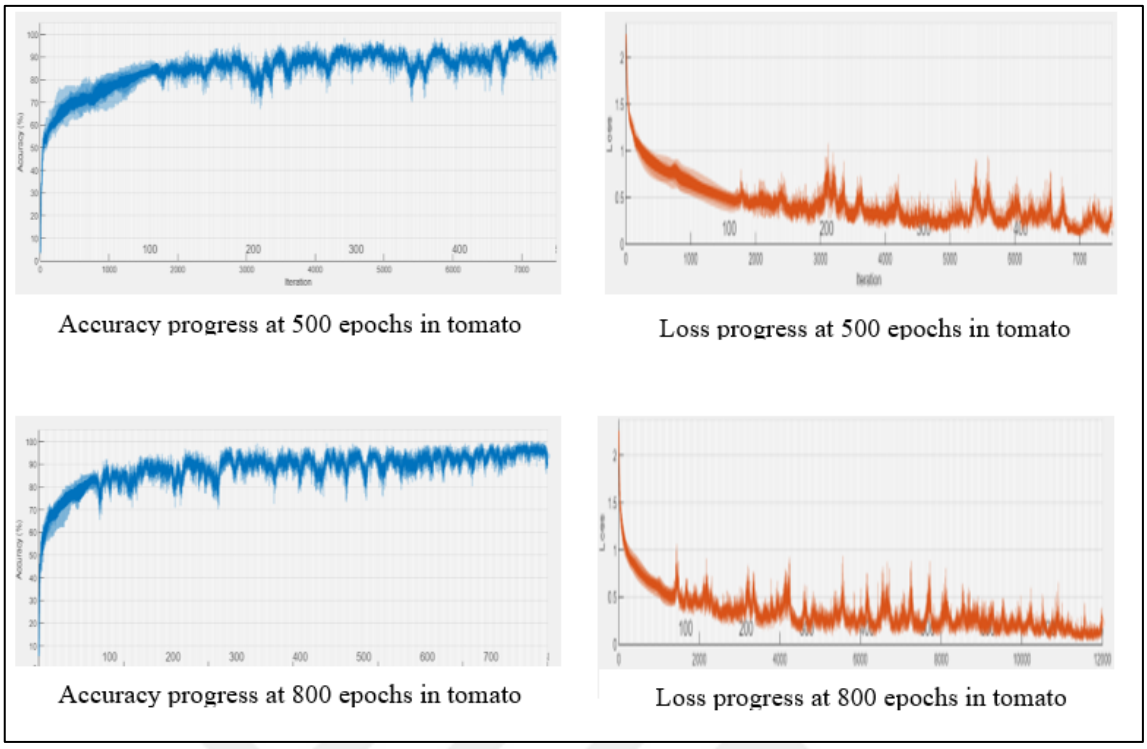


Figure 5.6: Accuracy and Loss Progress in CNN of Tomato Plant.

6. CONCLUSIONS AND FUTURE WORKS

6.1 OVERVIEW

The proposed system for automatic plant disease detection and diagnosis uses two main subsystems: a ML subsystem and a DL subsystem, with the latter being the main contribution of this work. The system aims to address the challenge of detecting and diagnosing plant diseases, which can be a difficult task for farmers.

6.2 CONCLUSIONS

The gathering of test findings led to the following deductions. The following list includes some of such conclusions:

i. It indicates that the suggested strategy for identifying and classifying plant leaf diseases is a reliable and precise methodology. It may offer significant benefits to farmers by helping them to detect and diagnose plant diseases early.

ii. Two of the most efficient and effective ML classification methods, SVM and KNN and the second subsystem uses a CNN and achieved the highest results among the two systems. By comparing the results obtained by the two systems, the most efficient method for identifying and categorizing plant leaf diseases can be determined.

iii. The ML and DL subsystems of the proposed system were tested and evaluated using the same dataset of photos of plant diseases. The dataset was divided into two sets: a training set with an 80% size and a testing set with a 20% size. This made guaranteed that both subsystems were assessed under the identical circumstances and that the outcomes could be adequately compared.

iv. The SVM algorithm for pepper plant achieved an accuracy of 93.14% at 1800 sample size and 10 k-fold, while the KNN algorithm achieved an accuracy of 89.30% for potato plant at 2202 sample size, 10 k-fold and accuracy of 95.18%% for tomato plant at 12000 sample size ,10 k-fold. The CNN algorithm achieved an accuracy of 99.80%for pepper plant, 99.90%for potato plant and 95.70% for tomato plant. However, it is important to note that the specific details of the implementation and the dataset used may influence the results and should be taken into consideration when drawing conclusions.

6.3 SUGGESTION FOR FUTURE WORKS

It is advised to take the following factors into account for upcoming tasks:

a. If implemented on a mobile platform, the suggested approach for diagnosing plant diseases may provide more convenience and flexibility. Farmers and users may quickly access the system and use it to identify and treat plant diseases on the road by putting the system on a mobile device. This may help to address the challenge of plant disease detection and diagnosis by providing farmers with a portable and accessible tool to help them identify and treat plant diseases early.

b. Testing the method with additional plant diseases that were not included in the original paper would be beneficial to confirm its efficacy and generalizability. By expanding the types of plant diseases included in the dataset and evaluating the performance of the system on these diseases, it may be possible to identify the strengths and limitations of the system and improve its accuracy and reliability for a wider range of diseases. This could lead to a more comprehensive and effective tool for plant disease detection and diagnosis.

c. A field research on plant diseases would be beneficial for addressing the issues with plant diseases and enhancing the nation's agricultural output. By conducting a survey of the types of plant diseases affecting crops in Iraq, as well as their prevalence, causes, and impact on crop yield and quality, it may be possible to identify key areas for intervention and improvement. This could include developing strategies for prevention and control of plant diseases, improving crop management practices, and implementing training and education programs for farmers and agricultural workers to help them recognize and respond to plant diseases effectively. By addressing the challenges of plant diseases, it may be possible to improve agricultural productivity and contribute to the overall economic development and well-being of the country.

d. Classification of plant diseases based on plant genes and biological changes using modern computer applications is an interesting and promising area of research. By analysing gene expression and other molecular markers associated with plant diseases, it may be possible to develop more accurate and reliable methods for detecting and classifying plant diseases. Insights into the underlying processes of illness genesis and progression may also be gained by this method, which might result in the creation of more potent medicines and management techniques. The use of modern computer

applications, such as ML and DL algorithms, can help to analyse large and complex datasets of gene expression data and identify patterns and associations that may be difficult to discern through traditional methods. The overall goal of this strategy is to enhance our knowledge of plant diseases and our capacity to control and avoid them.

REFERENCES

[1] Sindhu, P., and G. Indirani. "IoT based Wheat Leaf Disease Classification using Hybridization of Optimized Deep Neural Network and Grey Wolf Optimization Algorithm." Annals of the Romanian Society for Cell Biology (2020): 35-53.

[2] Anjana., and K. Keshav. " Plant Leaf disease classification and detection with CNN and optimization." International Journal of Research in Electronics and Computer Engineering (IJRECE) (2018): ISSN: 2393-9028.

[3] Nagaraju, Mamillapally, and Priyanka Chawla. "Systematic review of deep learning techniques in plant disease detection." International journal of system assurance engineering and management 11 (2020): 547-560.

[4] Hemavathi, S., et al. "Plant Uses And Disease Identification Using Svm Technology." Ilkogretim Online 19.3 (2020): 4626-4633.

[5] Prakash, R. Meena, et al. "Detection of leaf diseases and classification using digital image processing." 2017 international conference on innovations in information, embedded and communication systems (ICIIECS). IEEE, 2017.

[6] Ma, Juncheng, et al. "A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network." Computers and electronics in agriculture 154 (2018): 18-24.

[7] Ngugi, Lawrence C., Moataz Abelwahab, and Mohammed Abo-Zahhad. "Recent advances in image processing techniques for automated leaf pest and disease recognition–A review." Information processing in agriculture 8.1 (2021): 27-51.

[8] Sun, Guiling, Xinglong Jia, and Tianyu Geng. "Plant diseases recognition based on image processing technology." Journal of Electrical and Computer Engineering 2018 (2018).

[9] Reeta Janet Jessy I, Mrs. N. Bindu M.E, Mr. N. Rajkumar M.E. " COLOR CO-OCCURRENCE AND BIT PATTERN FEATURE BASED CBIR." International Journal of Scientific & Engineering Research (2016) ISSN 2229-5518

[10] Orchi, Houda, Mohamed Sadik, and Mohammed Khaldoun. "On using artificial intelligence and the internet of things for crop disease detection: A contemporary survey." Agriculture 12.1 (2022): 9.

[11] Yuvaraj, N., T. Karthikeyan, and K. Praghash. "An improved task allocation scheme in serverless computing using gray wolf Optimization (GWO) based reinforcement learning (RIL) approach." Wireless Personal Communications 117 (2021): 2403-2421.

[12] Harakannanavar, Sunil S., et al. "Plant leaf disease detection using computer vision and machine learning algorithms." Global Transitions Proceedings 3.1 (2022): 305-310.

[13] Verma, Shradha, Anuradha Chug, and Amit Prakash Singh. "Prediction models for identification and diagnosis of tomato plant diseases." 2018 International Conference on advances in computing, communications and informatics (ICACCI). IEEE, 2018.

[14] Kaur, Navneet, and V. Devendran. "WITHDRAWN: Novel plant leaf disease detection based on optimize segmentation and law mask feature extraction with SVM classifier." (2020).

[15] Sullca, Cecilia, et al. "Diseases detection in blueberry leaves using computer vision and machine learning techniques." International Journal of Machine Learning and Computing 9.5 (2019): 656-661.

[16] Hussein, Mohammed A., and Amel H. Abbas. "Plant leaf disease detection using support vector machine." Al-Mustansiriyah Journal of Science 30.1 (2019): 105-110.

[17] Fujita, Erika, et al. "Basic investigation on a robust and practical plant diagnostic system." 2016 15th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2016..

[18] Anagnostis, Athanasios, et al. "A convolutional neural networks based method for anthracnose infected walnut tree leaves identification." Applied Sciences 10.2 (2020): 469..

[19] Geetharamani, G., and Arun Pandian. "Identification of plant leaf diseases using a nine-layer deep convolutional neural network." Computers & Electrical Engineering 76 (2019): 323-338.

[20] Zhang, Shanwen, Wenzhun Huang, and Chuanlei Zhang. "Three-channel convolutional neural networks for vegetable leaf disease recognition." Cognitive Systems Research 53 (2019): 31-41.

[21] Balakrishna, K., and Mahesh Rao. "Tomato plant leaves disease classification using KNN and PNN." International Journal of Computer Vision and Image Processing (IJCVIP) 9.1 (2019): 51-63.

[22] Lu, Yang, et al. "Identification of rice diseases using deep convolutional neural networks." Neurocomputing 267 (2017): 378-384..

[23] Available:https://www.scantips.com/lights/resize.html

[24] Mumuni, Alhassan, and Fuseini Mumuni. "Data augmentation: A comprehensive survey of modern approaches." Array (2022): 100258..

[25] Available: https://www.projectpro.io/article/data-augmentation/777

[26] Available: https://iq.opengenus.org/data-augmentation/

[27] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of big data 6.1 (2019): 1-48.

[28] Thorpe, Simon, Denis Fize, and Catherine Marlot. "Speed of processing in the human visual system." nature 381.6582 (1996): 520-522..

[29] Treisman, Anne M., and Garry Gelade. "A feature-integration theory of attention." Cognitive psychology 12.1 (1980): 97-136..

[30] Ahmed, Ali Saadoon, et al. "Detection of COVID-19 Using Classification of an X-Ray Image Using a Local Binary Pattern and K-Nearest Neighbors." 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE, 2022.

[31] Khalid, Samina, Tehmina Khalil, and Shamila Nasreen. "A survey of feature selection and feature extraction techniques in machine learning." 2014 science and information conference. IEEE, 2014.

[32] Ojala, Timo, Matti Pietikäinen, and David Harwood. "A comparative study of texture measures with classification based on featured distributions." Pattern recognition 29.1 (1996): 51-59.

[33] Kaplan, Kaplan, et al. "Brain tumor classification using modified local binary patterns (LBP) feature extraction methods." Medical hypotheses 139 (2020): 109696..

[34] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Vol. 1. Ieee, 2005..

[35] Zhou, Wei, et al. "Histogram of oriented gradients feature extraction from raw bayer pattern images." IEEE Transactions on Circuits and Systems II: Express Briefs 67.5 (2020): 946-950..

[36] Brynolfsson, Patrik, et al. "Haralick texture features from apparent diffusion coefficient (ADC) MRI images depend on imaging and pre-processing parameters." Scientific reports 7.1 (2017): 4041.

[37] Hall-Beyer, Mryka. "GLCM texture: a tutorial v. 3.0 March 2017." (2017).

[38] Available:https://www.mathworks.com/help/images/texture-analysis-using-the-gray-level-co-occurrence-matrix-glcm.html.

[39] Löfstedt, Tommy, et al. "Gray-level invariant Haralick texture features." PloS one 14.2 (2019): e0212110..

[40] Available: https://www.javatpoint.com/normalization-in-machine-learning.

[41] Available:https://www.sciencedirect.com/topics/computer-science/feature-normalization.

[42] Available:https://developers.google.com/machine-learning/data-prep/transform/normalization.

[43] Available: https://deepchecks.com/glossary/normalization-in-machine-learning/.

[44] Xie, Qiyue, et al. "Optimization of heliostat field distribution based on improved Gray Wolf optimization algorithm." Renewable Energy 176 (2021): 447-458.

[45] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.

[46] Zareie, Ahmad, Amir Sheikhahmadi, and Mahdi Jalili. "Identification of influential users in social network using gray wolf optimization algorithm." Expert Systems with Applications 142 (2020): 112971..

[47] Sun, Xiaodong, et al. "State feedback control for a PM hub motor based on gray wolf optimization algorithm." IEEE Transactions on Power Electronics 35.1 (2019): 1136-1146.

[48] Available: https://en.wikiversity.org/wiki/Algorithm_models/Grey_Wolf_Optimizer.

[49] Kramer, Oliver. Dimensionality reduction with unsupervised nearest neighbors. Vol. 51. Berlin: Springer, 2013.

[50] PETERSON, Leif E. K-nearest neighbor. Scholarpedia, 2009, 4.2: 1883.

[51] STEINBACH, Michael; TAN, Pang-Ning. kNN: k-nearest neighbors. In: The top ten algorithms in data mining. Chapman and Hall/CRC, 2009. p. 165-176.

[52] Singh, Archana, Avantika Yadav, and Ajay Rana. "K-means with Three different Distance Metrics." International Journal of Computer Applications 67.10 (2013).

[53] Danielsson, Per-Erik. "Euclidean distance mapping." Computer Graphics and image processing 14.3 (1980): 227-248.

[54] Sinwar, Deepak, and Rahul Kaushik. "Study of Euclidean and Manhattan distance metrics using simple k-means clustering." Int. J. Res. Appl. Sci. Eng. Technol 2.5 (2014): 270-274.

[55] Xu, Hang, et al. "An evolutionary algorithm based on Minkowski distance for many-objective optimization." IEEE transactions on cybernetics 49.11 (2018): 3968-3979.

[56] Norouzi, Mohammad, David J. Fleet, and Russ R. Salakhutdinov. "Hamming distance metric learning." Advances in neural information processing systems 25 (2012)..

[57] Yamashita, Rikiya, et al. "Convolutional neural networks: an overview and application in radiology." Insights into imaging 9 (2018): 611-629..

[58] Available:https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[59] Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[60] Alzubaidi, Laith, et al. "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions." Journal of big Data 8 (2021): 1-74.

[61] Available:https://www.techtarget.com/searchenterpriseai/definition/backpropagation-algorithm

[62] Available:https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[63] Available:https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02

[64] Available: https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

[65] Vujović, Ž. "Classification model evaluation metrics." International Journal of Advanced Computer Science and Applications 12.6 (2021): 599-606.

[66] Available: https://www.ml-science.com/confusion-matrix