



REPUBLIC OF TÜRKİYE
ALTINBAŞ UNIVERSITY
Institute of Graduate Studies
Information Technologies

**DETECTION THE SICKNESS OF THE TEA
PLANT LEAF BY USING NEURAL NETWORKS**

Saja Bilal Hafedh AL-KARAWI

Master's Thesis

Supervisor

Asst. Prof. Dr. Hakan KOYUNCU

İstanbul, 2024

DETECTION THE SICKNESS OF THE TEA PLANT LEAF BY USING NEURAL NETWORKS

Saja Bilal Hafedh AL-KARAWI



Information Technologies

Master's Thesis

ALTINBAŞ UNIVERSITY
2024

The thesis titled DETECTION THE SICKNESS OF THE TEA PLANT LEAF BY USING NEURAL NETWORKS by SAJA BILAL HAFEDH AL-KARAWI and submitted on 18/04/2024 has been **accepted unanimously** for the degree of Master of Science in Information Technologies.

Asst. Prof. Dr. Hakan KOYUNCU

Supervisor

Thesis Defence 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

Asst. Prof. Dr. Warish PATEL

Department of Computer
Science,

Parul University

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

I affirm that all the information and data presented in this graduation project have been acquired in strict adherence to academic regulations and ethical standards. Furthermore, I declare that any non-original content and conclusions have been appropriately referenced within the text, and all references listed in the Reference List have been cited within the text, as per the aforementioned rules and ethical guidelines.

Saja Bilal Hafedh AL-KMARAWI

Signature



DEDICATION

I would like to begin by expressing my gratitude to Allah, the Supreme Being, for bestowing upon me the knowledge, skills, and the chance to conduct and successfully finish my research. I am truly appreciative of my parents, as they have been the driving force behind all the positive experiences in my life, and it is their love that enables everything to happen. Additionally, I want to acknowledge and thank my sister and brother for their unwavering love and support during this period.



PREFACE

I am deeply indebted to my advisor, Assistant Professor Dr. Hakan Koyuncu, for offering me the guidance, supervision, motivation, and support I needed to successfully complete my studies. His dedication, willingness to collaborate with me, constructive feedback, and valuable suggestions have significantly enhanced my knowledge and confidence. His thorough review of my work, along with his efforts to point out my mistakes, is greatly appreciated. Finally, I would like to extend my gratitude to all who have supported me, offered their prayers, or provided valuable advice.



ABSTRACT

DETECTION THE SICKNESS OF THE TEA PLANT LEAF BY USING NEURAL NETWORKS

AL-KARAWI, Saja Bilal Hafedh

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Hakan KOYUNCU

Date: 04 / 2024

Pages: 106

This thesis investigates the automated detection of leaf illnesses in tea plants by combining optimization algorithms with cutting-edge computer methods, particularly neural networks. The need for agriculture is growing to maintain crop health with guarantee maximum yields, it is critical to promptly and accurately diagnose plant diseases. With Convolutional Neural Networks (CNNs) serving as the main architecture, the aim of this thesis is to build a reliable and effective technique that can identify several diseases that harm tea plants. The methodology entails the procurement from differing dataset of tea plant leaves, comprising the healthy and unhealthy specimens. Preprocessing the dataset takes into account variables including disease severity, illumination, and image resolution in order to improve model performance. The most recent optimization techniques, such as the mayfly and pelican optimization algorithms (MA), are used in this study to train the neural network using this well selected dataset. The method's performance in disease detection is estimated using metrics like accuracy, MSE, F-score and recall, and sensitivity. The CNN-POAMA model, which was proposed, attained values of 94.5%, 0.035, 0.91, 0.93, and 0.92, respectively. The findings of this study have important ramifications for the farming sector as well as for the growth of automated technologies that have the possible to totally convert the way tea plants manage illness. The model's scalability and potential for real-world application demonstrate how well neural networks and optimization algorithms work together to solve challenging agricultural problems.

Keywords: Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Support Vector Machine (SVM), Pelican Optimization Algorithm (POA), Mayfly Optimization Algorithm (MA).



TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT.....	vii
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
ABBREVIATIONS	xvi
1. INTRODUCTION.....	1
1.1 INTRODUCTION	1
1.2 MOTIVATION.....	3
1.3 PROBLEM STATEMENT	3
1.4 AIM AND OBJECTIVES.....	4
1.5 RESEARCH IMPORTANCE	4
1.6 DESIGN AIMS	5
1.7 CONTRIBUTION	5
2. LITERATURE REVIEW	6
2.1 INTRODUCTION	6
2.2 LITERATURE REVIEW	6
3. MATERIALS AND METHODS	19
3.1 INTRODUCTION.....	19
3.2 ARTIFICIAL NEURAL NETWORK (ANN)	19
3.2.1 Components of Artificial Neural Network (ANNS)	20
3.2.2 Neurons.....	25
3.3 CONVOLUTIONAL NEURAL NETWORK (CNN).....	25
3.3.1 Definition of the Convolutional Neural Network (CNN).....	25
3.3.2 Introduction to Convolutional Neural Network (CNN)	26

3.3.3 Convnet Layers	27
3.3.4 Activation Functions	30
3.3.5 The Training and Testing of CNN	31
3.3.6 Common Architecture of CNN	32
3.4 SUPPORT VECTOR MACHINE (SVM)	33
3.4.1 Identification of SVM.....	33
3.4.2 Introduction to SVM.....	34
3.4.3 SVM Analysis Step	35
3.4.4 Types of SVM Algorithm	38
3.4.5 How (SVM) Work.....	38
3.5 METAHEURISTIC OPTIMIZATION ALGORITHMS	39
3.5.1 Definitions.....	39
3.5.2 Metaheuristic Search Algorithm	40
3.5.3 Introduction to Metaheuristic Optimization Algorithm.....	40
3.5.4 Types of Metaheuristic Algorithm	40
3.6 PELICAN OPTIMIZATION ALGORITHM (POA)	42
3.6.1 Mathematical Model of POA.....	43
3.6.2 (POA) Mathematical Complexity	47
3.7 MAYFLY OPTIMIZATION ALGORITHM (MA).....	48
3.7.1 Introduction to the (MA) Algorithm	48
3.7.2 Flowchart and Pseudo Code of (MA).....	49
4. PROPOSED METHOD	52
4.1 INTRODUCTION	52
4.1.1 Scope of the Project.....	53
4.2 TEA LEAF DISEASES DETECTION.....	53

4.3 IMAGE SEGMENTAION.....	55
4.3.1 Feature Extraction	56
4.3.2 Feature Selection	58
4.3.3 Mayfly Algorithm in Feature Clustering	59
4.4 SVM ALGORITHM.....	60
4.5 DATASET DESCRIPTION.....	60
5. IMPLEMENTATION AND RESULSTS	61
5.1 METHODOLOGY	61
5.1.1 Hardware and Software Specifications.....	62
5.2 IMPLEMENTATION.....	63
5.2.1 Dataset Preparation.....	63
5.2.2 Image Processing.....	63
5.2.3 Machine Learning.....	64
5.2.4 User Interface	64
5.3 DATASET	65
5.4 RESULTS	66
5.5 FEATURE EXTRACTION USING (CNN).....	66
5.5.1 Training and Testing.....	67
5.6 FEATURE SELCTION AND REDUCTION WITH (POA) AND (MA) ALGORITHM.....	70
5.6.1 Feature Selection with (POA).....	70
5.6.2 (MA) Algorithm for Feature Relation Discovery	71
5.7 CLASSIFICATION WITH (SVM) ALGORITHM.....	72
5.7.1 (SVM) Classification Using (POAMA) Feature.....	73
5.8 COMPARISON WITH BASELINE MODELS	75

5.9 DISCUSSION	76
6. CONCLUSIONS AND FUTURE WORK.....	78
6.1 FEATURE EXTRACTION USING (CNN)	78
6.2 FEATURE SELECTION AND REDUCTION WITH (POA) AND (MA).....	78
6.3 CLASSIFICATION WITH (SVM) ALGORITHM.....	78
6.4 PRACTICAL IMPLICATIONS.....	79
6.5 LIMITATIONS AND FUTURE DIRECTIONS	79
REFERENCES	80



LIST OF TABLES

	<u>Pages</u>
Table 2.1: Provides A Summary of the Many Works of Tea Literature Above.	18
Table 5.1: Software Specifications.....	63
Table 5.2: Hardware Specification	63
Table 5.3: Dataset Divisions for Each Class	66
Table 5.4: CNN Softmax Parameters.....	67
Table 5.5: CNN Feature Extraction Parameters	67
Table 5.6: CNN Accuracy and Error Rate by Epoch.....	70
Table 5.7: Convergence Accuracy of POA (Before MA) in Relation to Population Size or Iteration.	71
Table 5.8: Convergence Accuracy of POA (after MA) in Relation to Population Size or Iteration	72
Table 5.9: SVM Training and Testing Parameters	73
Table 5.10: SVM Accuracy and MSE by Number of Features Before POA.....	74
Table 5.11: SVM Accuracy and MSE by Number of Features After POA	74
Table 5.12: Model Performance Comparison	75
Table 5.13: Comparison of the Accuracy of the Proposed Model with Benchmark Methods	76
Table 5.14: Comparing the CNN-POA-MA Method with State-of-the-Art POA Based Methods.....	76

LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: The Difference Between a Healthy Leaf and an Infected Leaf	3
Figure 3.1: The Components of Artificial Neural Network.....	21
Figure 3.2: Sigmoid Function.....	22
Figure 3.3: TANH Function	23
Figure 3.4: RELU Function.....	23
Figure 3.5: Component of CNN	27
Figure 3.6: Basic ConvNet Architecture Displaying Layers that Alternate Between Convolution and Pooling.....	29
Figure 3.7: A Typical Convolutional Neural Network	32
Figure 3.8: Support Vector Machine (SVM) Classifier	34
Figure 3.9: Metaheuristics Algorithm Categories	42
Figure 3.10: Pelican Bird	43
Figure 3.11: Flowchart of (POA).....	46
Figure 3.12: The Mayfly	48
Figure 3.13: Inspiration of (MA).....	49
Figure 3.14: Pseudo Code of (MA)	50
Figure 3.15: Flowchart of (MA).....	51
Figure 4.1: CNN Extraction of Features From the Leaf Image.....	52
Figure 4.2: General Outline Diagram of the Proposed Method	54
Figure 4.3: Classes of the Input Images	56
Figure 4.4: Diagram of the POA Algorithm.....	58
Figure 4.5: MA Clustering of POA Features	59
Figure 5.1: Flow Chart of the Proposed Disease Detection.....	62

Figure 5.2: Data Split.....	63
Figure 5.3: Teal leaf Normalization Using Tuning, Flipping and Cropping.....	64
Figure 5.4: CNN Accuracy and Error Rate During the Training Phase	68
Figure 5.5: Precision of the CNN	68
Figure 5.6: Recall of the CNN.....	69
Figure 5.7: CNN Testing Phase Accuracies for Each Class	69
Figure 5.8: SVM Precision in the Training and Testing Phase	73
Figure 5.9: SVM Recalling the Training and Testing Phase.....	74
Figure 5.10: Comparing the Accuracy of Each Category Detection.	75
Figure 5.11: Comparing the Accuracy, MSE, Fscore, Recall of POA Based Methods.....	76

ABBREVIATIONS

AI	:	Artificial Intelligence
ANN	:	Artificial Neural Network
BRT	:	Boosted Regression Tree
CNNs	:	Convolutional Neural Networks
GA	:	Genetic Algorithm
ICT	:	Information and Communication Technology
KNN	:	K-Nearest Neighbors Algorithm
MA	:	Mayfly Optimization Algorithm
POA	:	Pelican Optimization Algorithm
SLIC	:	Simple Linear Iterative Clustering
SMO	:	Sequential Minimum Optimization
SVM	:	Support Vector Machine
TSWV	:	Tomato Spotted Wilt Virus

1. INTRODUCTION

1.1 INTRODUCTION

The quantity and quality of tea are both diminished due to the presence of tea leaf disease. Photosynthesis, fertilization, germination, transpiration, and pollination are just a few of the vital plant processes that these diseases impede. Important disease signs include changes in leaf color. Tea leaves change color when they are attacked by harmful pathogens, yet they still seem different when they are healthy. Plant diseases, especially those affecting tea plants, are becoming more common in today's world due to the heightened unpredictability of weather patterns and other environmental factors [1]. Diseases affecting tea plants are most commonly caused by fungi, bacteria, viruses, and algae. The following figure (1) shows a healthy tea leaf next to an infected one. Further illnesses may develop as a result of unfavorable environmental factors [2]. Though they love it, tea plants are vulnerable to illnesses in the temperate climates and plenty of rainfall that characterize the areas where tea plantations flourish. The quality of tea has dropped dramatically as a result of these diseases, which has made constant production difficult as tea production has increased. Symptom observation is the gold standard for diagnosing plant diseases. Field surveys conducted by crop protection specialists using disease classifications based on their experience are currently the main method for finding illnesses in tea leaves. It is difficult and costly to artificially detect illnesses, particularly in hilly, isolated regions where tea is grown. It is time-consuming and expensive to detect diseases in tea since there aren't enough qualified pathologists and farmers don't know what to look for. Additional methods for identifying plant diseases include biological spectroscopy, molecular biology, and microscopy. Even for highly trained plant pathologists, microscopic diagnosis is laborious and error prone. More precise yet time-consuming and expensive specialized equipment is required for spectroscopic and molecular biology studies. To avoid massive epidemics that could cause farmers significant losses, early disease detection is essential. Agricultural producers will be able to accurately and quickly identify diseases in tea plants if a diagnostic system is developed and put into place. These developments will lead to better methods of disease control, which will reduce monetary losses. In addition to that, they will improve the quality of the tea, lower the expenses of production and labor, increase the yields, and eventually make tea manufacturing sustainable. The agricultural sector is one that is rapidly benefiting

from technological advancements [3]. Improving agriculture is one of the primary goals of information and communication technology (ICT) [4]. Automated techniques for disease detection and categorization in tea leaves are encouraged to be developed by the concept of integrating ICT with agriculture. The detection of plant diseases has made extensive use of machine learning and image processing methods, including artificial neural networks. One study that used an improved random forest approach to classify groundnut diseases was Chaudhary et al. (2016) [5]. For the purpose of detecting foliar diseases in soybeans, Tetila et al. [6] evaluated numerous classifiers. Ehsan et al. suggested a fuzzy logic classification technique to improve the classification of healthy and sick strawberry leaves [7]. Common tea leaf diseases, such as brown blight and algal leaf disease, were identified by Hossain et al. using support vector machines [8]. Due to their small changes in color and texture, traditional machine learning algorithms like decision trees, support vector machines, adaboost, and random forests require the preextraction of disease attributes. However, it is possible that these traits do not adequately represent the characteristics of tea leaf illnesses. Conversely, when there are enough training examples, deep learning approaches provide better accuracy and do not require manual feature extraction. In order to detect and diagnose plant illnesses, Ferentinos created a deep learning method that uses a huge collection of images of leaves [9]. For the purpose of detecting diseases in maize leaves, Zhang et al. suggested improved deep learning models [10]. To isolate diseased tea leaves from visually chaotic backgrounds, Sun et al. [11] presented a method that combines SLIC with a support vector machine (SVM). Using images of tea leaves, Karmokar et al. proposed a neural network-based illness identifier [12]. Four common leaf diseases, including leaf spot, rust, and tail plaque, have also been identified using an SVM and an image recognition-based pattern detection method [13]. Traditional machine learning methods have been employed to detect various leaf diseases, not only those affecting tea leaves. On the other hand, deep learning methods' capacity to automatically extract characteristics from data makes them superior in target recognition [14]. By improving and segmenting images of tea leaves and training a convolutional neural network (CNN) on the results, Sun et al. were able to detect illnesses in tea leaves [15]. More and more, scientists are investigating the possibility of using optimization algorithms, specifically a combination of deep learning and metaheuristic optimization algorithms, to identify plant illnesses. To identify plants infected with tomato spotted wilt virus (TSWV), Qing et al. [16] employed hyperspectral imaging and AI

algorithms. Efforts like this show how cutting-edge innovation might change the way diseases are identified and classified in the agricultural sector. [17].



Figure 1.1: The Difference Between a Healthy Leaf and an Infected Leaf.

1.2 MOTIVATION

The essential motive of this thesis is to provide the besets method or algorithm that give the better results in the fastest time and the highest possible accuracy in diagnosis and early detection of plant diseases because of their direct impact on the risk of human disease because the plant is the basis of the human food pyramid, in addition to the exposure of plants Diseases threaten the decline of plant products and poor production quality, as well as keeping pace with technological development in artificial intelligence and its methods in various fields of life and shedding deep machine learning algorithms in the agricultural field through the artificial intelligence is continued development algorithms in the use of images of plant leaves in detecting diseases for the possibility of treatment before the death of the plant and thus Effect on ground cover plant.

1.3 PROBLEM STAETMENT

- a. Difficulty in identification and detection of diseased plants early.
- b. A lot of effort and time are spent by farmers and experts in diagnosing.
- c. The high financial cost, material losses and detecting error rate in disease detection.
- d. hard to use in difficult and changing environmental conditions.

The importance of this study lies in reaching the most appropriate solutions used in detecting plant diseases and determining the type of disease to help farmers and owners of small fields, by employing artificial intelligence techniques and machine learning methods through the use of deep neural network algorithms and metaheuristic optimization algorithms by taking a picture of the plant leaf. This mechanism will extract the features from the paper and

analyze it and give a result on whether the seedling is sick or healthy, and this will provide them with ease and save time, effort and financial cost, and also to avoid the death of the plant, as this will negatively affect the reduction of the agricultural product and the vegetation cover of the land and material losses as well as reducing efficiency And the quality in production if the disease is not detected and thus affects the health of the person whose food is mainly dependent on plants, due to the lack of necessary experience and the material cost of experts and the length of time spent in the process of disease detection, and sometimes some environmental conditions prevent good vision and thus the inability Determine if she is sick or not. In this research, we used the leaves of the tea plant as a model to work on.

1.4 AIM AND OBJECTIVES

This research aims to employ the photographic feature by employing artificial intelligence methods and artificial neural network algorithms in identifying and diagnosing plant diseases quickly and with high accuracy to help farmers save time, effort and the high financial cost spent, Also, this project helps them to easily deal with the detection of plants, whether they are sick or not. The research helps the following people and organizations:

- a. Owners of fields and farms with large areas
- b. Owners of simple fields with small areas
- c. Factories responsible for the production of foodstuffs derived from plants at a high rate.

1.5 RESEARCH IMPORTANCE

The importance of research lies in the early detection of plant diseases, as well as the appropriate diagnosis of these diseases, in addition to reducing the time, cost and effort expended. It may kill it, which threatens to decrease the numbers of plants, and expand the circle of knowledge in detecting diseases easily and easily by providing smart applications that depend on detecting the occurrence of plant diseases and diagnosing the infection area with determining the type of disease in preparation for prescribing the appropriate treatment for this disease and its treatment. The importance of this is related to the relationship of plant life on the planet.

1.6 DESIGN AIMS

This project aims to build and develop an intelligent model to identify plant diseases. The tea plant was used as an example, as it can be detected by taking a picture of the plant leaf. The mechanism will analyze the image and give the result if it is infected or healthy.

- a. Taking an image data set of plant leaves for diseased and healthy uninfected samples and treating them to make them suitable for use.
- b. Employing artificial neural networks and taking image data as input for the network, where the network trains and examines these images, analyzes them, and sends them to another neural network.
- c. Using metaheuristic optimization algorithms with neural networks to improve feature extraction and obtain high efficiency.

1.7 CONTRIBUTION

The main contribution of this study is the creation and verification of a computerized system for the precise identification and categorization of illnesses affecting tea plants' leaves. For tea to be produced with good quality, early diagnosis and control of leaf diseases are essential. Though laborious, drawn-out, and frequently arbitrary, manual disease detection of tea plants is a challenge. The solution put forth in this thesis for these problems is an intelligent system that can quickly and accurately identify diseases by automatically analyzing digital photographs of tea plant leaves. The thesis introduces a novel method that blends optimization methods with convolutional neural networks (CNNs) and support vector machines (SVM). Using a sizable dataset of labeled photos of tea plant leaves that have been preprocessed to extract pertinent features, CNNs are trained. The neural network parameters are adjusted using optimization algorithms to increase classification accuracy overall. The research findings show that the proposed system performs better than existing approaches through numerous trials and comparative analysis. With an average classification accuracy of over 90% across several illnesses of tea plants, disease identification is more accurate than earlier methods. Additionally, the system exhibits resilience to changes in illumination, leaf morphologies, and disease severity. In conclusion This contribution has substantial practical ramifications in addition to contributing to the study of tea plant pathology.

2. LITERATURE REVIEW

2.1 INTRODUCTION

New developments in AI and computer vision have shown great promise for a range of agricultural uses, especially in the area of plant disease diagnosis. Convolutional Neural Networks (CNNs) are a prominent technology in this field. CNNs are a type of deep learning algorithm that are well-known for their outstanding image identification skills. Researchers have also improved the accuracy and generalizability of neural networks by optimizing their parameters using methods like particle swarm optimization and genetic algorithms. This literature review seeks to explore the latest research and developments in the realm of detecting leaf diseases in tea plants using neural networks and optimization methodologies. By critically evaluating relevant studies, techniques, datasets, and performance metrics, we aim to identify research gaps and opportunities. In the subsequent chapters of this thesis, we will introduce and validate an innovative automated system for identifying tea plant leaf diseases, employing a fusion of neural networks and optimization algorithms. The insights gleaned from this comprehensive literature review will form the theoretical foundation of our thesis, with the ultimate goal of advancing agricultural technology and fostering sustainable tea production practices. The primary objectives of this literature review encompass the following:

- a. To give a general overview of the most cutting-edge methods and procedures used to identify plant diseases, with an emphasis on illnesses that affect tea plants.
- b. To assess the performance and efficiency of neural networks in identifying tea plant diseases and compare their outcomes with conventional approaches.
- c. To evaluate how optimization methods contribute to improving the reliability and accuracy of illness identification models.
- d. To identify potential obstacles and the shortcomings of current methods and suggest new research avenues for the creation of a reliable and effective system for identifying tea plant diseases.

2.2 LITERATURE REVIEW

If we want to make real, ethical strides, we need to study what others have done in this area. Here we will take a look back at a number of studies that have looked at plant disease

detection, identification, and classification in general, and at tea plant illnesses in particular. The use of deep learning, machine learning, and other AI approaches was crucial to these investigations. Using the *Pyrenophora tritici-repentis*-caused tan spot in wheat as a model system, De Wolf et al. (1997) [18], pioneered the use of artificial neural networks to evaluate the precision of disease forecasts in plant science. They used a bioassay method that linked the occurrence of tan spots to the stage of crop growth and a 24-hour snapshot of environmental data to determine when sensitive wheat cultivars were infected with *P. tritici-repentis*. The model that produced the highest accurate infection predictions was chosen after 20 neural network models with different processing nodes were evaluated. Factors such as the number of processing nodes and the usage of random coefficient initialization determined that the accuracy of these 200 models generally ranged from 74% to 77%. The most accurate model, which used five processing nodes, achieved an accuracy of 87% for the validation set, demonstrating that neural networks beat statistical approaches. Different research comparing the efficacy of multiple linear regression (MLR) with that of an ideal General Regression Neural Network (GRNN) for forecasting leaf wetness was conducted by Chtioui Y et al. (1999) [19], When it comes to plant disease development, predicting leaf moisture is no easy feat. However, micrometeorological parameters can help with this daunting endeavor. Prediction errors of 0.13 and 0.14, respectively, were shown by the simulations for the training and test sets for MLR. On the other hand, LSTM produced a mean absolute prediction error of 0.0894 and GRNN produced a mean absolute prediction error of 0.0491 using the same dataset. These inaccuracies were minor given that the moisture content of leaves might range from 0 to 1. Therefore, when contrasted with MLR, the optimized GRNN showed much better prediction accuracy. An online approach for crop and weed discrimination was introduced by Feyaerts F et al. (2001) [20], using multi-spectral reflectance data from image spectrographs. With this method, pesticide levels were reduced by as much as 90% in 86% of the field-tested plant samples, including 80% of the crops and 91% of the weeds. To categorize Empire and Golden Delicious apples according to their surface quality, Kavdir I et al. [21] used backpropagation neural networks in 2002. They were able to distinguish between imperfect and perfect apples by using artificial neural networks to analyze texture data and pixel gray values from full-image photos of apples. The accuracy in a 2-class categorization was between 89.2% and 100%. Empire apples had a success percentage of 93.8% to 100% in a 5-class categorization based on the selected

attributes, whereas Golden Delicious apples had a success rate of 89.7% to 94.9%. Marique, T. et al. [22], looked at the standards of potato chips. In order to find the top, middle, and bottom gray level intensities of each slice, they examined fried test photos. They proceeded to link this data with color categories by constructing a feed-forward artificial neural network. With training data correlation coefficients of 0.972 and validation data correlation coefficients of 0.899, the network obtained a high level of agreement with human estimations. When Moshou D. Canopy reflectance in different wavebands was investigated by et al. [23] as a potential tool for plant disease diagnosis. During the early stages of the "yellow rust" illness, they compared the spectral reflectance of healthy and infected wheat plants. Utilizing self-organizing maps to depict data attributes and variable interactions, they devised a normalization technique that took into consideration fluctuations in light intensity and reflectance. When tested using 5137 leaf spectra, their classification ability increased from 95% to over 99% thanks to the use of neural networks, particularly multilayer perceptron. Pydipati R et al. (2005) [24], used the color co-occurrence method (CCM), a texture analysis methodology, in conjunction with a number of neural network classifiers, including back-propagation and radial basis functions, as well as Mahalanobis minimum distance classifiers that adhere to the closest neighbor concept. Incorporating saturation and hue texture data into leaf sample discriminant analysis utilizing the Mahalanobis statistical classifier and CCM texture analysis regularly produced remarkably high classification accuracies, topping 95%. Furthermore, accuracy levels surpassing 90% were attained using a back-propagation neural network approach. The same year, Burks TF et al. [25], investigated three separate neural-network classifiers that may be used in real-time weed management systems. These classifiers used color co-occurrence method (CCM) texture analysis techniques. In a six-class data set (40 images per class), they included 33 textural features per image for diverse plant taxa, such as foxtail, crabgrass, common lambs quarter, velvetleaf, morning glory, and clean soil surface. With a remarkable 97% performance accuracy, the backpropagation neural network classifier blew beyond the traditional 93% accuracy rate of the statistical classification methods. By 2006, Sammany M. employed a Support Vector Machine (SVM) for the purpose of symptom classification in plants, namely RS, WS, YS, and D symptoms [26]. Compared to an improved MLP's 80% accuracy, the SVM's 83% accuracy across all five classes is significantly better. Additionally, in 2006, a new approach to creating weather-based prediction models for plant diseases using support

vector machines was presented by Kaundal R et al. [27]. They compared their approach to conventional multiple regression (REG) after selecting six important meteorological variables to use as prediction factors. By implementing BPNN, the correlation coefficient (r) was raised from 0.60 to 0.70, and the percent mean absolute error (%MAE) was decreased from 65.42% to 52.24%. The %MAE was lowered to 46.30% and the R-value was further enhanced to 0.77 by the generalized regression neural network (GRNN). With an R-value of 0.74 and a %MAE of 44.12, the strategy based on support vector machines (SVMs) performed better than all others. Using the same criteria as before, SVMs maintained their superiority in cross-location validation. Color image segmentation using the BP neural network was the subject of a study by Lu, Jun-Wei et al. [28] that same year. A system that can differentiate between the leaves of real onweeds that thatand fake weeds was created by them. The procedure consisted of four stages: first, removing soil from the image to separate the plants; second, color-based image segmentation; third, merging small segments according to their average color indices and frontier rates; and finally, a BP neural network was employed to ascertain whether the combined areas were indeed onion leaves. The method's accuracy levels ranged from 80% to 90%. Using automation and the basic RGB color model, Abdullah NE et al. [29] were able to diagnose diseases affecting rubber tree leaves in 2007. In order to quantify normalized indices generated from the RGB color distribution, they examined illness images' regions of interest (ROI). This system made use of 600 training samples and 200 testing samples in conjunction with artificial neural networks. Incorporating principal component analysis (PCA) and optimizing the ANN model based on the dominating pixel RGB (mean) led to a sensitivity level above 80% and a diagnostic accuracy of about 70% for both models. Phalaenopsis seedling diseases, such as Phytophthora black rot (PBR), bacterial brown spot (BBS), and bacterial soft rot (BSR), were looked into in 2007 by Huang KY et al. [30]. To isolate the affected parts of Phalaenopsis seedlings, they employed image processing techniques and an exponential transform with a parameter that could be adjusted. The classification approach made use of the gray level co-occurrence matrix (GLCM) and three-color criteria to evaluate the textural aspects of the lesion area. Using a back-propagation neural network classifier, scientists achieved an accuracy of 89.6 percent in classifying Phalaenopsis seedlings as belonging to BSR, BBS, PBR, or uninfected (OK) zones. They accomplished the astounding feat of achieving a detection effectiveness of up to 97.2% without even specifying the disease kind.

Xiang and colleagues [31] created a method for spectrally predicting tomato late blight infestations in 2008 using a gradient-descent learning algorithm and a backpropagation (BP) neural network. This method used two experimental samples to train artificial neural networks (ANNs) to differentiate between healthy and diseased tomatoes. Consistent with expected values from observed data, the correlation coefficient for field data was 0.99 and for remote sensing images it was 0.82. Before 2009, Kluge A. and coworkers [32] created a weed-identification system based on artificial neural networks. The team made use of a unique geometric characteristic that has a modest link with leaf size—the distance between two spots on the leaf axis and one point at the leaf margin. This approach was used to investigate *Veronica hederifolia* from many perspectives, and recognition rates for *Galium aparine* were 93.4% and 90%, respectively. Using image processing and support vector machines (SVM), Yeung et al. [33] concentrated on rice disease identification in the same year. The researchers used segmentation to identify disease patches in rice based on their shape and texture. Then, they used support vector machines to categorize the rice diseases bacterial leaf blight, rice sheath blight, and rice blast. With a 97.2% success rate, the data showed that certain disorders could be accurately identified. It was 2010 when Al Bashish D. The use of image processing and a pre-trained neural network for the detection of plant leaf/stem diseases was suggested in [34]. They used the K-Means method to segment images, and the neural network classifier they developed had an impressive 93% accuracy rate. Pfaff, T. Sugar beet leaf health was defined in the same year by et al. [35] as distinct from illnesses like *Cercospora* leaf spot, leaf rust, and powdery mildew. Using hyperspectral data and a support vector machine with a radial basis function kernel, they were able to classify the health of sugar beet leaves with an astounding 97% accuracy and distinguish between healthy and diseased leaves 86% of the time. A method for categorizing sunflower crop leaves using neural networks was introduced in 2011 by Arribas JI et al. [36]. The four-step process used the Generalized SoftMax Perceptron architecture of a neural network for classification, feature extraction, color space segmentation, and feature selection. They were able to get an impressively high Correct Classification Rate (85% on average) with their method. It was also same year that Prasad S. In their study, et al. [37] investigated the use of support vector machines for the detection and classification of plant leaves. Using 300 leaf features from a dataset of 624 leaves, they were able to accurately identify 23 different plant species with an average accuracy of 95%. An image processing and artificial neural

network-based method for plant disease identification was introduced by Kulkarni AH et al. [38] in 2012. For categorization tasks, their technique attained a remarkable 91% accuracy rate. Another study that year used computer image processing to identify cucumber illnesses Wei Y et al. in 2009 [39]. Using a BP neural network model, they increased recognition accuracy to 80% by employing the region-growing method to extract scab patches of cucumber leaves. Aji AF et al. [40] suggested a way to identify palm oil infections in 2013 by examining outward symptoms using neural networks and image processing. After extracting six distinct features from photos of palm leaves, they used the neural network method to reach an average accuracy of 87.75%. In the same year, Zhang et al. [41] created a system to detect six prevalent jujube tree illnesses. Their method was able to diagnose a variety of diseases with success rates ranging from 73% to 94% by taking texture, color, and morphology into account. Sumathi CS et al. [42] achieved a 95.56% accuracy rate in automating leaf recognition for plant classification in 2014 using feed-forward neural networks with several learning algorithms and an information gain-based feature selection technique. With variable sensitivities for different cases, Mohd SS et al. [43] achieved an overall accuracy of 84.4% using leaf color analysis and an ANN technique for early diagnosis of rubber tree white-root disease. A neuro-fuzzy expert system, which integrates neural networks and fuzzy logic, was used to study three major paddy illnesses in 2015 by Kahar MA et al. [44]. The level of accuracy in their recognition reached 74.21%. Kawasaki Y. also debuted that year. A convolutional neural network (CNN) based system for plant disease identification was introduced by et al. [45]. Their CNN-based system, which was trained using 800 photos of cucumber leaves, was able to classify cucumber diseases with an average accuracy of 94.9%. Additionally, the system was able to generate more images to use as a training dataset. A model for disease detection in plants using leaf image classification was developed in 2016 by Sladojevic S and colleagues [46]. using deep convolutional networks. The model's accuracy in identifying healthy leaves from thirteen distinct plant illnesses ranged from 91% to 98% on independent class tests, with an average precision of 96.3%. In 2016, Nachtigall LG et al. [47] investigated using CNNs to identify and categorize apple tree illnesses automatically using images of the leaves. Using a one-of-a-kind dataset containing labelled samples from six recognized diseases, their trained CNNs were able to attain an astounding 97.3% accuracy, matching or even outperforming expert performance. Last year, 2017, Liu B. [48] offered convolutional neural networks (CNNs)

trained on AlexNet as a means to reliably identify apple leaf diseases. When evaluated with a dataset of 13,689 photos of sick apple leaves, their approach reduced model parameters and improved picture accuracy, achieving an overall accuracy of 97.62%. Good day, Lu Y. In 2017, strategies for disease identification in rice using deep convolutional neural networks were suggested by and colleagues [49]. They trained convolutional neural networks (CNNs) to detect ten prevalent rice diseases using a dataset including 500 photos of both healthy and damaged rice stems and leaves. Using a 10-fold cross-validation process, they attained an astounding 95.48% accuracy. For the purpose of illness diagnosis in tomato leaves, Tm P et al. [50] employed a model of convolutional neural networks called LeNet in 2018. This model was able to automatically extract features from input photos and attain an accuracy rate ranging from 94.5 to 95%. Author: Arivazhagan B. In 2018, a method based on deep learning was presented by and colleagues [51] for the automatic detection of illnesses in mango leaves. Using a dataset of 1200 photos, their CNN model performed exceptionally well, properly classifying five distinct leaf diseases with a rate of 96.67%. With the year 2019 upon us, Geetharamani G. as well as coworkers [52] put out a model for disease detection in plant leaves using deep convolutional neural networks (Deep CNNs). Using a dataset that included pictures of 39 different plant leaves and a number of data improvement methods, they trained the model. An impressive 96.46% classification accuracy was attained by the proposed model. Hasan MJ et al. [53] used the SVM classifier and a convolutional neural network model to detect diseases in rice in the same year (2019). With an astounding 97.5% success rate, they were able to identify and classify nine distinct rice illnesses. In 2019, Ahila Priyadharshini R et al. [54] introduced a deep convolutional neural network (CNN) architecture, namely a modified LeNet, for the purpose of maize leaf disease classification. Their CNNs achieved a remarkable 97.89% accuracy rate when trained on a dataset consisting of photographs of maize leaves, correctly classifying the leaves into four separate groups. The 2020 study by Li X and colleagues [55] aimed to classify apple leaf diseases and identify them. The results showed that ResNet-18 achieved a high accuracy grade of 98.5% when compared to the SVM image segmentation classifier and the VGG ResNet convolutional neural network model. Additionally in 2020, an improved recognition method based on convolutional neural networks was proposed by Liu B and colleagues [56] to treat grape leaf diseases. They surpassed competing models with a total accuracy of 97.22% on the test set after enhancing multi-dimensional feature extraction with the

Inception framework. A convolutional neural network (CNN) was presented in 2021 by Lee TY and colleagues [57] for the purpose of identifying potato infections; this network achieved an impressive accuracy of 99.53% with a substantial reduction in parameter consumption. To identify Bacterial Spot disease in peach plants in 2021, Bedi P. and colleagues [58] used leaf pictures. With few training parameters, they achieved a testing accuracy of 98.38% and a training accuracy of 99.35% using a hybrid model that utilized CNNs and Convolutional Autoencoder (CAE) networks. Paymode AS et al. [59] studied potential diseases that could impact tomato and grape leaves in 2022. Accuracy scores of 95.71% for tomatoes and 98.40% for grapes were achieved using CNN techniques to identify Multi-Crops Leaf Disease (MCLD). The VGG model, which is based on CNN, was used in this process. Altalak M. also passed away in 2022. suggested a hybrid deep learning strategy for tomato leaf classification [60]. The accuracy percentage of this hybrid system was 97.2%, and it was accomplished by combining CNNs, CBAMs, and SVMs (support vector machines). Mr. Kumar A. In 2022, et al. [61] employed several methods, including support vector machines, to identify and categorize plant leaf diseases. Their accuracy rate was 96.77%, which was higher than that of SVM and simple CNN models. A new convolutional neural network (CNN) architecture for disease detection and classification in rice plants was presented by Singh SP et al. [62] in 2023. Along with 1,400 aerial photos of healthy rice leaves, they trained the CNN architecture using four separate datasets pertaining to diseases affecting rice plants. Maximal accuracies of 99.66% for Stochastic Gradient Descent with Momentum (SGDM) and 99.83% for Adaptive Moment Estimation (Adam) were achieved through the use of optimization methods. Still another study from 2023 by Lamba S. The optimization of convolutional neural networks for the diagnosis of different plant leaf diseases was the primary focus of et al. [63]. Using data from Plant Village, which covers fifteen categories of tomato, potato, and pepper plant illnesses, they improved the dataset using a generative adversarial network and reached a 98% accuracy rate. The next step is to go over the literature on plant disease categorization using neural networks and metaheuristic optimization methods. Genetic algorithms (GAs) and Probabilistic Neural Networks (PNNs) were investigated by Islam MN et al. [64] in 2012 for the purpose of plant disease classification and identification. With a classification rate of 97%, GAs outperformed PNNs, which had a rate of 94.75%. To determine the severity of red jujube sickness, Bai TC et al. [65] in 2012 studied a neural network that was trained using an upgraded Particle Swarm

Optimization Back Propagation (PSO-BP) algorithm. The performance and accuracy of the model were enhanced by 94.0% and 87.6%, respectively, as a consequence of this change. A system called Plant Leaf Image Retrieval (PLIR) was created in 2012 by Valliammal N et al. [66]. It used genetic algorithms (GA) to select and fuse features, and it used support vector machines (SVM) and nearest neighbors (KNN) for classification. With GA-SVM, 94% of the time was successful, and with GA-KNN, 89%. In their 2013 study, Sumathi CS et al. [67] addressed the topic of plant classification using leaf recognition. With the use of machine learning, they were able to extract features and classify them with a precision of 94.81%. In 2014, Zhang Z et al. [68] used machine vision to improve neural network particle swarm optimization and identify damaged maize leaves. With the optimization network, they were able to reach a 93.3 percent identification rate. As of 2014, Mohan S. A system for knot classification in wood boards was introduced by et al. [69]. It combines Hilbert transforms, Gabor filters, Multilayer Perceptron (MLP), and Neural Network (NN) trained with PSO and IWO. Their success rate was 91%. One model for plant identification that was proposed in 2017 by Eid et al. [70] included biometric data from leaves and swarm optimization methods. The recognition rates for the Flavia dataset reached 98.9% and the Swedish dataset reached 93.3%, both of which were greatly improved by their strategy. The detection and classification of plant diseases was accomplished with an accuracy of approximately 86% in 2018 by Chouhan et al. [71] using a Radial Basis Function Neural Network (RBFNN) trained with Bacterial Foraging Optimization (BFO). One study that employed a deep convolutional neural network (CNN) trained with Particle Swarm Optimization (PSO) to identify and classify plant leaf diseases was Yadav et al. [72] in 2019. For 23 different categories, their method was 97.39% effective. In 2019, the authors Chanda et al. [73] used back-propagation and Particle Swarm Optimization (PSO) to optimize the weights of neural network (NN) connections for image processing. They were able to attain a classification accuracy of 96.2%. Using k-means image segmentation, support vector machine (SVM) training, particle swarm optimization, color and texture attributes, and seven different plant species, Kour et al. [74] presented a method for plant identification and segmentation in 2019. The accuracy values that were produced by this method were 95.23%. By utilizing attention-based depthwise separable neural networks with Bayesian optimization (ADSNN-BO), Wang et al. [75] achieved a test accuracy of 94.6% in 2021 when they tuned the hyperparameters of a disease detection model for images of rice leaves. Using a convolutional neural network

(CNN) method, Nandhini et al. [76] classified four separate illnesses affecting tomato leaves in 2021. By combining an encoding method with the ICRMBO algorithm, they were able to attain a remarkable classification accuracy of 99%. Using an optimized probabilistic neural network (OPNN), Akanksha et al. [77] achieved a 95.5% success rate in automating the diagnosis of illnesses in maize plants in 2021. In 2021, Goluguri et al. [78] employed a combination of deep convolutional neural networks (DCNN), deep short-term memory (LSTM), and support vector machines (SVM) to detect rice illnesses. They were 97.5% accurate using their EAFSO (efficient artificial fish swarm optimization) method. A metaheuristic method for improving CNN classification accuracy using the Cat Swarm Updated Black Widow Model (CSUBW) was suggested by Mohapatra et al. [79] in 2022. Their success rate was 91%. Using the Modified Sine Cosine Algorithm-Particle Swarm Optimization (MSCA-PSO) to optimize CNN parameters, Mishra et al. [80] presented a MobileNetV2 CNN model in 2022 for the purpose of recognizing and classifying avocado plant illnesses. They were 98.42% accurate using their method. Using feature extraction and classification algorithms, Arjunagi et al. [81] created a system for disease diagnosis in maize leaves in 2023. Utilizing an enhanced Spider Monkey optimization with adaptive opposition (AOSMO), they were able to attain a 90% success rate with an optimized Convolutional Neural Network (CNN). For the purpose of precise diagnosis of plant leaf diseases, Prabu et al. [82] presented a boosted support vector machine-based Arithmetic Optimization Algorithm (BSVM-AOA) in 2023. Picture segmentation, feature extraction, and classification were all parts of their method, and it yielded an impressive 98.6 percent accuracy. In 2023, a tailored PDICNet model for disease detection and classification in plant leaves was presented by Reddy et al. [83]. They used feature selection with the modified Red Deer optimization algorithm (MRDOA) and got F1-scores and accuracy levels of 99.83% on the PlantVillage dataset and 100.83% on the Rice Plant dataset. Research on plant disease detection and classification using neural networks, artificial intelligence methods, and metaheuristic optimization algorithms is covered extensively in this review. In 2006, Zhao et al. [84], achieved an accuracy rate of more than 90% by differentiating green, black, and oolong teas using near-infrared (NIR) spectroscopy in conjunction with support vector machine (SVM) classification. In 2008, a group of researchers led by Li [85], analyzed the Vis/NIR spectra of different types of tea plants. They used ANN after processing the spectrum data using methods like integrated wavelet transform (WT) and

principal component analysis (PCA) to get a satisfactory 77.3% classification accuracy. The tea leaf disease recognizer (TLDR) was proposed by Karmokar et al. in 2015 [86]. TLDR achieved a 91% success rate in identifying illnesses in tea leaves by utilizing image processing, feature extraction, and ANN training. To detect unwrapped tea leaves, Tang et al. (2015) [87] presented a technique that combines texture analysis with support vector machines (SVMs). With variables such as energy, correlation, and contrast retrieved from gray-level co-occurrence matrices (GLCM), they achieved a classification success rate of 83%. A method for identifying illnesses in tea was developed in 2017 by Zhang et al. [88], using hyperspectral imaging. They achieved recognition rates of up to 94.17% using support vector machines (SVMs) and neural networks, and they effectively diagnosed tea leaf lesions by applying principal component analysis (PCA) and other methods to hyperspectral pictures. By analyzing images of tea leaves, Xiaoxiao SU et al. in (2018) [89], were able to detect illnesses using convolutional neural networks (CNN). With the help of data augmentation, image segmentation, and CNN parameter experiments, they managed to attain an impressive identification accuracy of 93.75 percent. A Faster R-CNN model was trained to identify several diseases connected to tea in 2018 by Lee SH et al. [90], With initial detection rates ranging from 63.58% to 81.08%, they classified tea illnesses into groups including brown blight, blister blight, and algal leaf spot. For illness detection in tea leaves, Chen et al. (2019) [91], created a convolutional neural network (CNN) model named LeafNet. In terms of classification accuracy, LeafNet achieved an average of 90.16 percent, surpassing SVM and MLP methods. It did this by using dense scale-invariant feature transform (DSIFT) features in conjunction with SVMs or MLPs. By adding a multiscale feature extraction module to their model, Hu et al. (2019) [92], improved deep convolutional neural networks (CNN), leading to an average identification accuracy of 92.5%. A total success rate of 89.4 percent was achieved in 2020 by Lee SH et al. [93] when they used a Faster R-CNN model to detect pests and diseases in tea. Classifying tea disorders using a combination of color, shape, and texture attributes was the goal of Zou X and their team (2020). [94], With the use of machine learning techniques such as support vector machines (SVM), they were able to achieve an accuracy rate of 87.3%. After incorporating particle swarm optimization (PSO), the accuracy rate increased dramatically to 91.7%. An astounding 94.50% accuracy rate was achieved for disease categorization in tea leaves by Latha RS et al. (2021) [95] using deep convolutional neural networks (CNNs). Using genetic

algorithms and image processing, Mukhopadhyay et al. (2021) [96] suggested a method for detecting illnesses in tea leaves. An 83% success rate was achieved by their method, which included picture clustering using non-dominated sorting genetic algorithm II (NSGA-II), PCA for feature reduction, and multi-class support vector machines (SVMs) for disease identification. In addition, Chen et al. (2021) [97] aimed to use CNNs for tea pest classification, and they achieved a remarkable 97.75% accuracy rate, exceeding that of more conventional machine learning methods such as Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP). In a study conducted by Datta S and colleagues (2022), [98] a deep convolutional neural network (DCNN) with many hidden layers was used to classify sick tea leaves into distinct groups. The network achieved a detection rate of 96.56%. Using convolutional neural network (CNN) and support vector machine (SVM) methods, Jayanthi et al. [99], in (2023) achieved a high accuracy rate of 73.077% for tea bud classification and a rate of 96.703% for tea illness classification in real-time. Using ResNet-50 and Inception V3 Convolutional Neural Network (CNN) architectures, Trihardianingsih et al. [100], in (2023) achieved 97.62% and 97.44% accuracy rates, respectively, in classifying illnesses of tea leaves. The goal of Tanwar et al. (2023) [101], was to use deep learning techniques, namely convolutional neural networks (CNNs), to diagnose several diseases affecting tea leaves. They achieved an impressive 96% accuracy rate in their study. Finally, Pandian et al. in (2023) [102], presented a deep convolutional neural network (DCNN) model for diagnosing grey blight disease on tea leaves, achieving a remarkable classification accuracy rate of 98.99%. Table 2 shows some of studies and the method they used.

Table 2.1: Provides a Summary of the Many Works of Tea Literature Above.

Study	Year	Method
Zhao et al [84]	2006	SVM
Tang et al[87]	2015	SVM
Zhang et al [88]	2017	SVM & GA
Xiaoxiao SU et al [89]	2018	CNN
Hu et al [92]	2019	CNN (CIFAR10)
Zou X et al [94]	2020	BPCN & SVM& PSO optimization
Chen et al [97]	2021	(VGGNET-16 & CNN) (SVM) & (MLP)
Datta S et al [98]	2022	CNN
Trihardianingsih et al [100]	2023	Inception V3(CNN) & ResNet-50 (CNN)
Pandian et al [102]	2023	DCNN

3. MATERIALS AND METHODS

3.1 INTRODUCTION

We describe in this chapter the methods and techniques used to accomplish the goals of our research on automated tea plant leaf disease identification. In order to develop a precise and effective system for recognizing various tea plant leaf illnesses, this study aims to take advantage of neural network capabilities, notably Convolutional Neural Networks (CNNs) in combination with optimization methods. By presenting our strategy in this chapter, we offer a thorough framework for the use, instruction, and assessment of our disease detection model. A summary of our data collection approach, including the selection of pertinent datasets and considerations for data quality, is provided after a discussion of the study design that guides our investigation. Then, we go into detail about our neural network's design, explaining why we made the decisions we made and describing the model's layers, nodes, and activation mechanisms. Parallel to this, we introduce the optimization strategy we have selected to adjust the parameters of our neural network and improve its overall performance. We discuss the potential effects of our research on stakeholders and the ethical issues surrounding the use of data. We also recognize the limitations of our study, including potential biases in the dataset and the inherent difficulties in illness diagnosis brought on by differences in leaf appearance brought on by elements like illumination and leaf aging. by methodically laying out our research methodology in this chapter, we offer a transparent and replicable framework that advances the field of tea plant disease identification while also serving as a road map for other researchers interested in using neural networks and optimization algorithms to address related agricultural problems.

3.2 ARTIFICIAL NEURAL NETWORK (ANN)

Some other names for neural networks include artificial neural networks (ANNs) and simulated neural networks (SNNs). They are an important part of deep learning techniques and machine learning more generally. They stand for an information processing system modeled after the brain's cerebral cortex, which consists of many basic yet highly linked processing units. Neural networks are incredibly effective at solving problems that traditional computers struggle with but which animals and humans have a natural talent for solving. They have recently grown into a robust field of study and practical application for

solving real-world issues [103]. A computational approach different from conventional analytical methods is provided by artificial neural networks (ANNs). Artificial neural networks (ANNs) learn to generalize from examples in the past by extracting important features from input data, discarding irrelevant information, and storing this knowledge in a memory. Neurons (or cells, units, or nodes) and synaptic transmissions with corresponding weight factors are used in neural networks to simulate the nervous system. The design of the network permits the transmission of signals in both serial and parallel modes. Although there are some apparent similarities between neural networks and the human brain, these similarities are mostly superficial [104]. The building blocks of an ANN are the layers of nodes, with each layer including an input, a hidden layer (or layers), and an output. There is a threshold and a corresponding weight for each artificial neuron, or node. Nodes activate and send data to the next network layer when their outputs surpass certain thresholds. In such case, the next tier does not receive any data. Training data is essential for neural networks so that they can learn and become more accurate over time. Surprisingly, ANN-related theory and research go back further than most people realize. While conducting research in the fields of psychology and neurophysiology, McCulloch and Pitts initially presented the basic ideas of brain computing in 1943 [105].

3.2.1 Components of Artificial Neural Network (ANNS)

Neural networks are constructed in a way to function similarly to the human brain. By passing an input layer, which may be thought of as the sensory organ used to receive the information, the neural network attempts to imitate the operations of connected neurons. The neurons in the buried layer are given access to the information in this way. According to the weights, each neuron assigns a given input node a higher priority than another, and finally, an output is created using the knowledge the neurons have gathered. A variety of artificial neural network components can be tweaked to enhance the ANN model. Better results may follow from each factor's influence, but this improvement may come at the expense of more model training time. components of artificial neural networks in general include the following: -

a. Input layer

Nodes that bring in the original data after pre-processing make up the input layer. Depending

on the classification issue, the data could pertain to any subject, but the numbers are always numerical. If not, pre-processing techniques must be used to turn them into numerical data.

b. Hidden layer

In a neural network, the layers that lie between the input and output nodes are called hidden layers. By analyzing the data sent by the weighted nodes, these hidden layers use activation functions to get a result. This layer is where the actual learning happens. As a biological neuron, the buried layer functions.

c. Output layer

It is a neural network's final layer. Depending on the categorization issue, the output layer may contain a single node or several nodes. Figure 3.1 shows the Components of Artificial Neural Network.

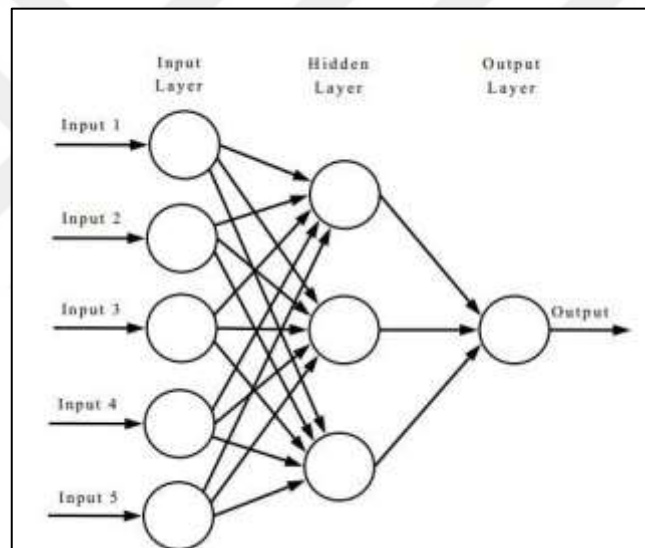


Figure 3.1: The Components of Artificial Neural Network.

d. Activation function

A node's output is determined by a function called activation, which uses the input as a starting point. The two functions of summation and activation are supported by neurons in neural networks.

a. Summation

The matrices of the input and weights make up a summation.

b. Activation

After the data has been summed, the values are transformed by activation. The outcome is

regarded as the output once the activation has been completed. Using the appropriate activation function will depend on the issue, use the sigmoid activation function for classifying data, the softmax function when doing multiclass classification, and the ReLU activation function when doing regression.

c. Linear activation

It symbolizes a linear transformation from input to output. It has a continuous gradient, making gradient descent impossible, which makes it a rarely used activation. Since you will receive 0 when you calculate the partial derivative with a linear activation, you cannot make it better.

d. Non-linear activation

This kind of activation modifies the input nonlinearly. In deep learning models, it is frequently employed. In various situations, several activations are used. Its types include:

a. Sigmoid Function

It translates the input value(x) in a range from 0 to 1, regardless of how big or little the input value is. It is also known as the logistic function. Typically, binary classification is done using it. Figure 3.2 shows the form of sigmoid function.

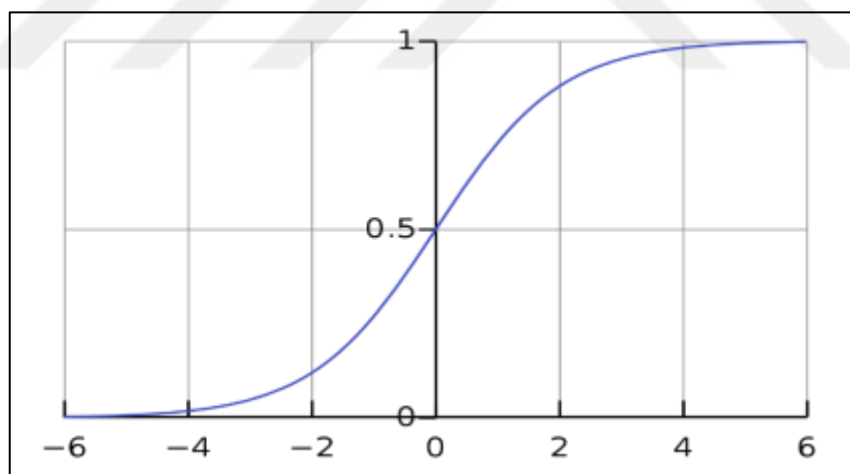


Figure 3.2: Sigmoid Function.

b. TANH Function

The main difference between the hyper tan function and the sigmoid function is that the input values are converted from -1 to 1 instead of 0 to 1. It produces more erratic values. Figure 3.3 shows the form of TANH function.

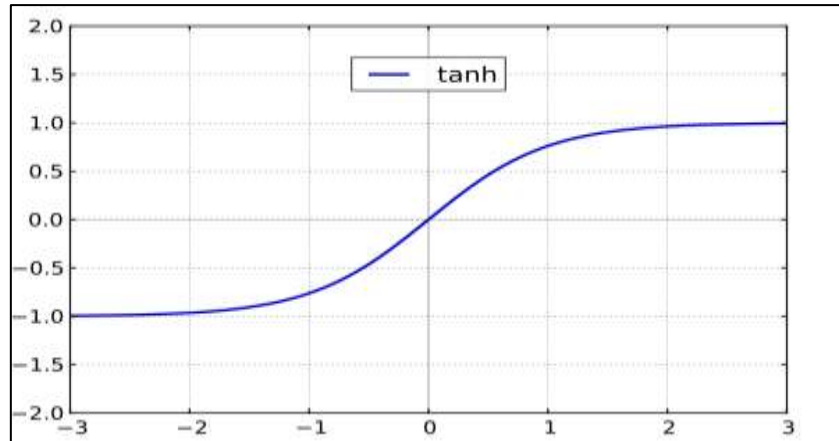


Figure 3.3: TANH Function.

c. RELU Function

Rectified Linear Unit' is what it stands for. $\text{Max}(0, x)$, where x is the positive input value, is used to calculate it. Any value less than 0 is regarded as 0, and the value above is used as-is. When working with non-linear data, ReLU is incredibly adaptable. Figure 3.4 shows the form of RELU function.

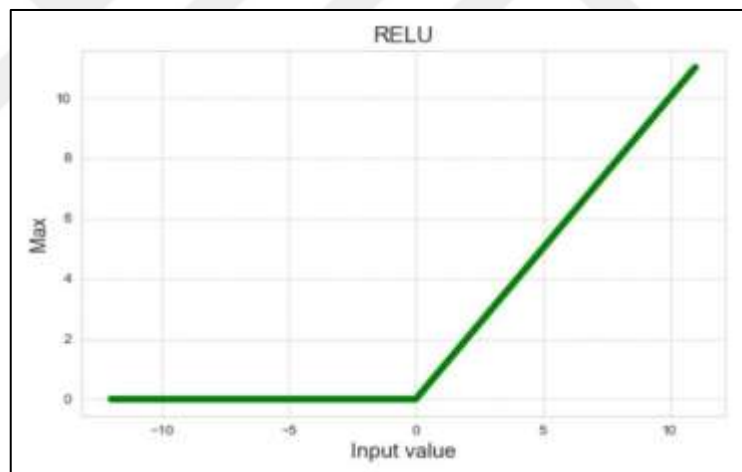


Figure 3.4: RELU Function.

d. SoftMax Function

Regarding classification, it is comparable to a sigmoid function. Typically, we employ it for multiclass classification. SoftMax differs from sigmoid in that class probabilities in SoftMax add up to 1.

e. Loss metrics

The loss metrics provide a quantitative evaluation of how inaccurate our predictions were. Greater loss results from bad predictions, and vice versa. It is the difference between the

actual output and the projected output in math. Below some of the loss metrics used by neural network:

a. MEAN SQUARED ERROR (MSE)

This function determines the mean square of all mistakes, or the difference between actual and expected values.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2 \quad (3.1)$$

b. MEAN ABSOLUTE ERROR (MAE)

The sole distinction between this function and mean absolute error is that in the former we take the mode of the error values rather than the square. Even with outliers, it still functions nicely. Because it produces a significant gradient even for small values, it is not commonly employed.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |x_i - x| \quad (3.2)$$

c. CROSS-ENTROPY LOSS (LOG LOSS)

This function evaluates how well a classification model performs by producing a probability between 0 and 1. A greater disparity between the expected and actual likelihoods leads to an increase in cross-entropy loss. In an ideal world, the log loss would be zero.

f. Learning rate

A stochastic gradient descent approach is used to train deep neural networks. The algorithm that determines the error gradient is an optimization one. In order to update the weights, backpropagation is used. The term "step size" or "Learning rate" refers to the increment by which the weight is updated during training. The learning rate hyperparameter regulates the model's learning rate or speed.

g. Optimizer

By adjusting the weights, optimizers give the model the best possible shape so that it can output an accurate form. Since it has no idea where to begin, it uses random values. The optimizer adjusts itself back in that way as the loss starts to lessen and that direction as it

starts to increase. Below some of the optimizer:

- a. Stochastic gradient descent
- b. Adagrad
- c. Adadelata
- d. RMSProp
- e. Nadam
- f. Adam

3.2.2 Neurons

In the construction of an Artificial Neural Network (ANN) designed to replicate the functionality of biological neurons, the fundamental unit is the artificial neuron. This artificial neuron computes its output by initially summing the incoming signals, known as inputs, which are then scaled by the connection weights (subject to adjustments), and subsequently passed through a transfer function. The sigmoid function represents one of the most commonly utilized transfer functions in this context. Essentially, the activation function embodies the weighted summation of inputs for the neuron. [106] is intended to behave similarly to a biological neuron, accepting a variety of signals (x_i) from numerous nearby neurons and processing them in a clear-cut manner. The conclusion of this process determines whether the neuron will fire an output signal. If the output signal is activated, it can either be 0 or 1 or any real value between 0 and 1, depending on whether we are working with artificial neurons that have "binary" or "real valued" output signals, respectively [107].

3.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

There are many types of artificial neural networks. In this research, we used convolutional neural networks.

3.3.1 Definition of the Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs or convnets) represent a category within the realm of machine learning, constituting one of the various forms of artificial neural networks applied across diverse data types and purposes. Specifically tailored for tasks involving pixel input processing and image recognition, CNNs are a specialized network architecture devised for

deep learning algorithms. They stand out as a form of regularized feed-forward neural network that autonomously discerns feature engineering by optimizing filters or kernels. This optimization involves the application of regularized weights across fewer connections, effectively mitigating challenges associated with vanishing and exploding gradients encountered in earlier neural networks during backpropagation [108]. The shared-weight architecture of convolution kernels or filters in CNNs allows them to slide over input features, producing translation-equivariant replies called feature maps [109]. This architecture gives CNNs the other name "Shift Invariant" or "Space Invariant Artificial Neural Networks" (SIANNs). Nevertheless, it should be mentioned that even though this terminology is used, the majority of convolutional neural networks do not display complete translation invariance. This is mostly because the input is downsampled [110]. In contrast to typical fully connected feed-forward neural networks, where every neuron in one-layer connections to every neuron in the succeeding layer, CNNs behave differently, which can lower their vulnerability to overfitting".

3.3.2 Introduction to Convolutional Neural Network (CNN)

Deep neural networks, or just deep learning models, are a few names for ANNs that have numerous layers. Deep Learning's extraordinary ability to process massive volumes of data has contributed greatly to its meteoric rise to prominence in the last several decades. Neural networks with more hidden layers have lately outperformed traditional methods in many domains, including pattern recognition. The convolutional neural network (CNN) is a prominent design in deep learning that mimics the way animals' visual cortex works [111]. Convolutional neural networks (CNNs) have broadened their applications beyond their original intent of object identification to include many more areas of study [112], [113], [114], such as object tracking, posture estimation, visual saliency detection, action recognition, scene labeling, and text detection and recognition. The neocognitron, which came out in 1980 [115], was the forerunner of ConvNets. Then, in 1990, LeNet was introduced by LeCun et al., which was a major step forward in the evolution of convolutional neural networks [116]. Designed initially for the categorization of handwritten digits, LeNet has now shown the capability to detect visual patterns in raw photos directly, without requiring any preprocessing. But there wasn't enough training data and processing power, so it couldn't do much. After using a CNN model in 2012, Krizhevsky et al. [117] [118]

achieved significant strides in lowering mistake rates during the ILSVRC competition. This study has become a cornerstone of computer vision research and provides a foundation for exploring other convolutional neural network (CNN) designs. As a notable variant of the LeNet architecture, AlexNet is a CNN design that uses five convolutional layers followed by three fully linked layers. Surprisingly, AlexNet achieved better results than earlier ConvNets using supervised learning, even though it did not undergo unsupervised pre-training [119]. After AlexNet won the ILSVRC-2012 competition, CNNs underwent a slew of improvements that cemented their position as the dominant paradigm in deep neural networks. Among the many uses for deep neural networks, convolutional neural networks (CNNs) have become ubiquitous, especially in the fields of machine learning and image processing. When it comes to picture classification, for example, they've shown that they can handle massive image datasets like ImageNet with ease. The fields of computer vision and natural language processing (NLP) have also benefited greatly from CNNs, with impressive results [120]. Reducing the number of parameters within artificial neural networks (ANNs) is one of the most useful properties of CNNs. Because of this success, scientists and programmers are using bigger and more complicated models to solve challenging problems that were previously intractable with conventional ANNs. One fundamental premise of CNN-addressed problems is that aspects that are dependent on space should not be relied upon. Convolutional neural networks (CNNs) also excel in abstracting features as input moves through deeper layers. In picture categorization, for instance, the bottom layers might detect edges, the middle layers simpler forms, and the top layers higher-level attributes. Figure 3.5 shows the form of CNN component.

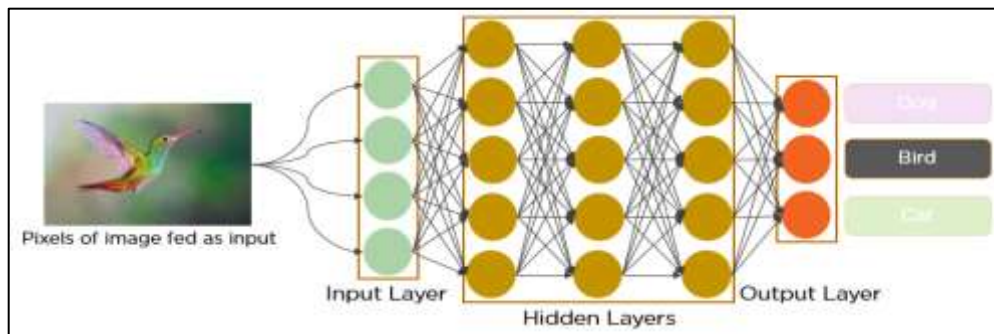


Figure 3.5: Component of CNN.

3.3.3 Convnet Layers

Similarities between traditional neural networks, which can be thought of as a collection of

neurons arranged in an acyclic graph, and convolutional neural networks (CNNs) are apparent. The most important difference between the two is that convolutional neural networks (CNNs) selectively connect each neuron in a hidden layer to a selection of neurons in the layer above it. Because of the lack of connections, the network is able to learn features implicitly. A hierarchical feature extraction technique is made possible by ConvNets' deep and layered structure. In the first layer, the filters are learned to recognize simple features like edges or color patterns. In the second layer, the filters are learned to discern more complicated shapes. In the third layer, the filters are learned to recognize the components of objects. And in the final layer, the filters are learned to recognize the objects themselves. Compared to more conventional neural networks, ConvNets stand out due to their hierarchical feature extraction mechanism.

a. Convolutional Layer

As shown in figure 3.6. Within its base layer, a Convolutional Neural Network (ConvNet) houses its central processing unit (CPU). A collection of feature maps, with neurons embedded in each, make up this layer. The parameters of this layer, which are learnable filters or kernels, are its beating heart. In order to create new 2-dimensional activation maps, these filters are convolutionally operated with the feature maps. The layer's output volume is then formed by stacking these maps along the depth dimension. By enabling neurons within the same feature map to share weights, weight sharing reduces the network's complexity while keeping a parsimonious parameter count [120]. Lin et al. [121], introduced "Network in Network" (NIN), which significantly differs from conventional ConvNets. Here, a global average pooling layer takes the place of the fully connected ones, and 1×1 convolution filters built as Multi-Layer Perceptrons (MLPs) replace the traditional linear filters. A structure called a "mlpconv" layer is formed by stacking MLPconv layers in this way. When compared to regular ConvNets, NIN shows a considerable improvement in its ability to extract hidden meaning. It has been shown that NIN's last mlpconv layer includes confidence maps for different categories, suggesting that NIN could be used for object recognition.

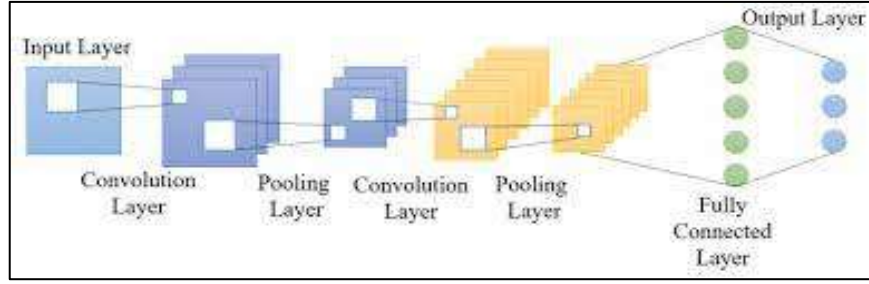


Figure 3.6: Basic ConvNet Architecture Displaying Layers that Alternate Between Convolution and Pooling.

b. Pooling Layer

The core design of a ConvNet consists of a series of convolutional layers with pooling layers inserted at regular intervals. Pooling layers serve a dual purpose: first, they decrease the total number of parameters in the network by efficiently reducing the spatial dimensions of the activation maps, which preserves crucial information. Overfitting can be prevented, and computational load is reduced as a result. Some of the most popular pooling methods used in ConvNets are maximal, average, stochastic, spectral, spatial pyramid, and orderless multiscale pooling. They are among many more [123]– [126]. The study by Alexey Dosovitskiy et al. [126], suggests a fascinating substitute. Their research suggests that convolutional layers with a stride of two can work in place of max pooling layers. Compared to numerous intricate architectural configurations, simpler networks have shown themselves to be more performant, therefore this finding is especially pertinent for them.

c. Fully Connected Layer

Every neuron in one layer of a conventional neural network uses complex logic to form full connections with all of the neurons in the next layer. But since the neurons don't have any kind of one-dimensional spatial ordering, it's not easy to incorporate a convolutional layer right after a fully connected one. The "Network in Network" (NIN) by Lin et al. is an example of how some architectural designs have chosen to replace fully connected layers with a global average pooling layer in response to recent improvements.[121].

$$F(X) = \text{MAX}(0, X) \quad (3.3)$$

d. Loss Layer

The final fully connected layer in the network is known as the loss layer. Its primary function is to compute the loss or error, penalizing any disparities between the desired and actual outputs. To predict a single class from a set of K mutually exclusive classes, the softmax

loss function is commonly employed. This function essentially amounts to multinomial logistic regression, normalizing predictions to create a probability distribution across classes and transforming them into nonnegative values. Alternatively, hinge loss is utilized for training the Support Vector Machine, a classifier focused on achieving large margins. For regression tasks involving real-valued labels, Euclidean loss serves as a valuable tool.

3.3.4 Activation Functions

Activation functions, also known as non-linearities, take a single input value and apply various mathematical operations to it. There are several different activation functions available, and we will discuss some of the more commonly used ones.

a. Sigmoid

For this specific non-linearity, the input is a real-valued function, and the output can be anywhere from zero to one, as shown in Equation 1. While convolutional neural networks (CNNs) are widely utilized, they do have two major drawbacks: (i) saturation and vanishing gradient problems, and (ii) the fact that sigmoid outputs aren't centered at zero, so gradients might be positive or negative.

$$F(X) = \frac{1}{1 + e^x} \quad (3.4)$$

b. TANH

Equation (2) clearly illustrates that Tanh can be seen as a scaled-up version of the sigmoid, resulting in values ranging from -1 to 1. While it also suffers from the issue of saturating gradients, it addresses the second problem mentioned earlier because its outputs are centered around zero. Consequently, tanh is often preferred over sigmoid in practice.

$$\tanh(x) = 2f(2x) - 1 \quad (3.5)$$

c. RELU

As described in Equation (3), the Rectified Linear Unit (ReLU) is a linear activation function with thresholding at zero. Utilizing ReLU can accelerate the convergence of gradient descent.

$$\text{rect}(x) = \max(0, x) \quad (3.6)$$

d. Leaky RELU

Leaky ReLU, which is represented by Equation (4), was obtained by slightly altering ReLU.

3.3.5 The Training and Testing of CNN

The computational needs and non-convex structure of the loss function make training a CNN with multiple layers a tedious procedure, particularly with big datasets [127]. Convolutional neural networks (CNNs) usually do not start with a random value when they are trained. Using an appropriate network initialization strategy to accelerate convergence is the norm rather than the exception [128]. If you need to initialize your weights meaningfully or have a tiny training dataset, transfer learning is another option. This shortens the training period because only the classifier or the last few layers are retrained. Learning rate, iterations, training and testing data, batch size, etc. are some of the characteristics that must be meticulously chosen in order to construct a reliable model [129]. To train the same network with different parameter choices and obtain more accurate results, an ensemble model can be developed. Network training makes use of backpropagation and gradient descent for weight updating. It could be difficult to train lower layers of deep networks due to the vanishing gradient problem. Selecting a sparse activation function, such as sigmoid or tanh, is an excellent solution to this issue. Because they are less affected by diminishing gradient flow, rectified linear units (ReLU) have been proven to be more effective for ConvNets in recent studies. [130]. The classifier, often a SoftMax classifier, is the final layer in a convnet. It produces probabilities for each class, with the probabilities summing up to 1. Figure 3.7 shows the structure of CNN. While SoftMax is commonly used, different classifiers like SVM and probabilistic classifiers may be employed depending on the specific situation. SVMs are used for structured prediction tasks, and classifier ensembles are also popular [131].

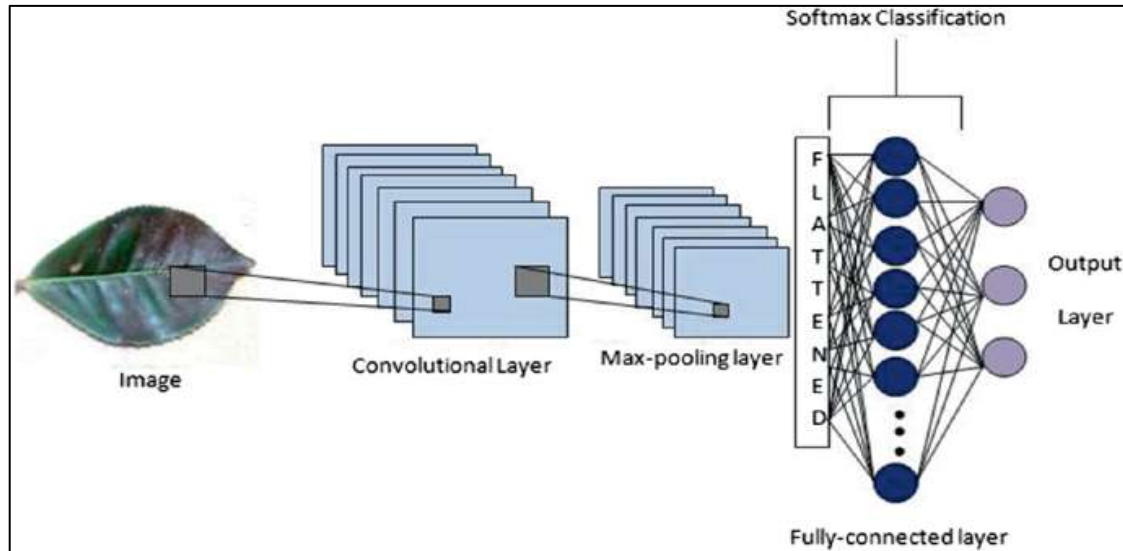


Figure 3.7: A Typical Convolutional Neural Network.

3.3.6 Common Architecture of CNN

a. LeNeT

Included in the seminal study on convolutional neural networks by LeCun et al. It underwent two rounds of enhancements in the 1990s [132] and 1998. Their work solved the problem of handwritten digit recognition using ConvNets. Zip codes, digits, and more may all be read with it.

b. ALeXNeT

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton [117], are widely acknowledged as the pioneers of utilizing convolutional networks for effective computer vision in their groundbreaking design. Similar to LeNet, AlexNet used alternating convolutional and pooling layers instead of stacking all of the convolutional layers. In comparison to LeNet, this network is both deeper and larger. On the test dataset, AlexNet achieved top-1 and top-5 error rates, allowing it to win the ILSVRC-2012 challenges.

c. Google NeT

At the 2014 ILSVRC, Google's ConvNet architecture proposed by Szegedy et al. [132] came out on top. A new architecture, Inception (v1), has been proposed by them to optimize the utilization of the network's computational resources. As an example, GoogleNet has fewer parameters than AlexNet but twenty-two Inception module layers in one version. Ioffe et al. developed Inception-v2 in response to numerous subsequent modifications to Inception-v1,

the most significant of which was batch normalization [133]. The architecture was known as Inception-v3, and this version included even more enhancements. [134].

d. VGGNet

While holding all other factors constant, Karen Simonyan and Andrew Zisserman performed thorough research into the depth component of ConvNets. They successfully dealt with the network's high parameter count by employing tiny 3x3 convolution filters in every layer. The ILSVRC 2014 competition was won by VGGNet, a more accurate ConvNet architecture that was developed as a result of this research. VGG-16 is used in this study.

e. Inception V4

The architecture that Szegedy et al. [135] presented in 2015 is based on GoogLeNet and uses residual connections to speed up the training of Inception modules, which were first described by Him et al. [136]. In the 2015 ILSVRC challenge, our network not only won, but also achieved state-of-the-art performance.

f. ReSNeT

As an alternative to learning referenced functions, Kaiming et al. [137] presented a framework for residual learning that allows layers to learn residual functions related to the inputs they receive. The method's ability to streamline learning while also producing noticeable gains in accuracy makes it ideal for training deeper networks. Because there are so many factors, this network has a high assessment cost, which is a downside. Although it has little effect on performance, the initial Fully Connected layer adds a lot of parameters and can be removed to lower the evaluation cost.

3.4 SUPPORT VECTOR MACHINE (SVM)

3.4.1 Identification of SVM

Among the many pattern recognition applications, Support Vector Machine (SVM) stands out for its widespread use in picture identification and network traffic classification, among other areas. Concepts from convex optimization and statistical learning theory provide its basis. Multiple domains have begun to utilize SVMs, such as computer vision, bioinformatics, and text classification. One type of supervised learning model that comes with related learning algorithms is the Support Vector Machine (SVM), which is also called

a Support Vector Network. When doing regression and classification, these models come in handy. With important contributions from scholars like Boser et al. (1992), Guyon et al. (1993), Cortes and Vapnik (1995), and Vapnik et al. [138], SVMs were initially created by Vladimir Vapnik and colleagues at AT&T Bell Laboratories. Figure 3.8 shows the form of Support Vector Machines.

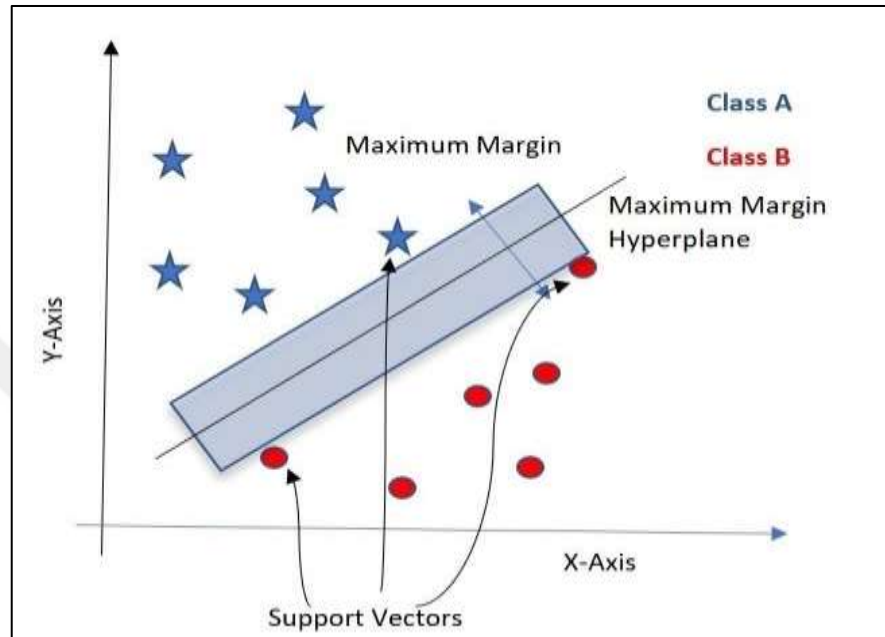


Figure 3.8: Support Vector Machine (SVM) Classifier.

3.4.2 Introduction to SVM

Among the first to create a support vector machine algorithm was Vladimir N. Vapnik and Alexey Y. Chervonenkis in 1964. Later improvements and upgrades have made it even better. An innovative kernel technique was introduced by Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik in 1992. It built nonlinear classifiers using SVM and was based on the idea of maximum-margin hyperplanes [139]. The "soft margin" version, first presented by Corinna Cortes and Vladimir Vapnik in 1993 and now included in most support vector machine (SVM) software, is a major step forward. The groundwork for the practical implementation of SVM algorithms was set by their pioneering work [138]. Support Vector Machine, or SVM for short, is a top-tier machine learning method that finds widespread use in pattern identification. Problems with incorrect card detection, face recognition, voice recognition, text categorization, and picture and speech recognition are only a few examples of the many pattern classification issues it has helped solve. The overarching goal of pattern

recognition research is to classify data sets using either existing knowledge or statistical information extracted from the original datasets. The training algorithm of support vector machines (SVMs), a type of supervised machine learning, uses a dataset of labelled training examples to build a predictive model that can be used to classify new examples. The capacity to generalize answers successfully is one of SVM's strengths; this is in line with the goals of statistical learning. Support vector machines are an effective method for categorizing data. Although Neural Networks are often thought of as easy to utilize, there are situations in which they can produce unintended outcomes. Training data and testing data are the two primary pieces of information used in a standard classification task. There are a number of target values and attributes present in every instance of the training set. In support vector machines (SVMs), the goal is to construct a model that can, using just their properties, forecast the target values for instances in the testing set. An excellent illustration of Supervised Learning is SVM classification, where the system's performance is evaluated with the help of known labels. You can use this data to make sure the system is accurate or to teach it how to do a better job. Finding related categories, often called feature selection or feature extraction, is a crucial step in support vector machine classification. When trying to determine which features are most important for class distinction, it is sometimes possible to use feature selection in conjunction with support vector machine classification [140].[141]

3.4.3 SVM Analysis Step

You can think of support vector machine analysis as having three separate steps. These steps are present in many machine learning methods and are not exclusive to support vector machines (SVM).

a. Feature Selection

Before you can train a support vector machine (SVM) classifier, you need to convert the raw training data into a set of "features" that the SVM may use as input. There are a number of feature selection algorithms that use some kind of quantification metric to rank features [142]. There are essentially three distinct kinds of feature selection approaches:

i. Embedded Methods

When training a support vector machine (SVM), feature selection happens automatically with embedded methods because it is included into the classifier itself. Doing so is possible

by employing a kernel strategy, sometimes called the "kernel trick." Actually, almost all support vector machine (SVM) uses in neuroimaging make use of kernel functions [143]. While training support vector machines (SVMs), kernel techniques improve computational efficiency and help reduce overfitting, which is particularly useful for difficult classification problems that are never well-conditioned in reality.

ii. Filter Methods

Feature reduction using filter techniques is done as a first step before classification. Prior to fitting a hyperplane, the training set is used to compute several relevance measures in order to remove the features that are not significant. Implementing feature reduction is important for three main reasons: It improves the ratio of training data from samples to feature dimensionality by reducing data redundancy in the raw data. Making Decisions More Interpretable: The final classifier can be better understood with the help of feature reduction. One way to focus future efforts is to find out which data points are the most predictive for differentiating between groups. Feature reduction has the potential to reduce the computational complexity of classification algorithms in certain instances. Before training an SVM classifier, it is possible to use a variety of feature reduction techniques. As a rule, this means getting rid of variables that have very correlated features and almost no variance at all, sometimes called "multicollinear" features. By simplifying the SVM in this way, we can lower its complexity without compromising its predictive power.

iii. Wrapper Methods

Unlike embedding methods, wrapper approaches repeatedly train the classifier and use feedback from each iteration to select a subset of features for each consecutive iteration. Although wrapper approaches incur higher computational costs, they offer the benefit of excluding data points that do not adequately differentiate between class labels when examined independently. When working with classic SVM, Recursive Feature Elimination (RFE) is a popular wrapping method. Using a recursive approach that takes ever-smaller subsets of characteristics into account, RFE systematically ranks features using cross-validation. For a classification job, this technique is useful for identifying which traits are most important.

b. The Classifier Training and Testing

When training a Support Vector Machine (SVM), we use examples of data for which the

labels have already been determined, such when we categorize cars according to their color. In order to forecast future label assignments, SVM uses this historical data. The support vector machine decision function draws a line and uses it to project the subject's feature coordinates onto it. For patterns that have projections that land on the positive side of the line ($y > 0$), we say that they are class A (like a black car) or class B (like a gray car). To train a support vector machine, one must first place the hyperplane such that it divides the two classes as much as possible using the parameters w and b in the decision function $f(x) = w * x + b$. The absolute value of a feature's weight in a linear classifier, such as SVM, indicates the feature's importance in differentiating between the two classes. Remember that this method works only if the classes are evenly distributed and not one has a disproportionately large number of samples. Techniques like class balancing can be used when there is a class imbalance, which can negatively affect the performance of classifiers [144]. The accuracy of the classifier is heavily dependent on the hyperparameter values used, in addition to the automatic adjustment of the core parameters w and b during the SVM estimator data fitting process. The fit of the decision function is affected by hyperparameters, which are variables that are set prior to training. Compared to neural networks and other ML techniques, support vector machines (SVMs) often have fewer tuning options; nonetheless, any "free" parameters that arise from [145].

c. Evaluating The (Svm) Performance

While there are other measures available, the most popular ones for evaluating SVM performance are sensitivity, specificity, and accuracy. In order to measure how well and consistently the SVM hyperplane differentiates across classes, several measures are crucial. Nevertheless, permutation testing is frequently required to thoroughly assess accuracy and reproducibility. Using different resampled datasets and a variety of hyperparameter values, permutation testing iteratively generates a hyperplane by randomly permuting class labels. This method optimizes performance across various criteria by use of cross-validation. Using many data partitions is critical for evaluating any classification learning technique, but especially SVM. This is because it's possible for one classifier to get an unfair advantage from the hyperplane that emerges from splitting a single dataset. The SVM model's repeatability can be assured with the use of cross-validation and permutation testing. While it's ideal to use as much data as possible to train a classifier, it's crucial to set aside some data for evaluating the final SVM model. As a first step, the dataset is partitioned into training

and testing subsets; the latter is "held out" for the final model evaluation in this approach. By doing so, we can more accurately evaluate the classifier's performance on new, unseen data and ensure that it can generalize beyond the training set.

3.4.4 Types of SVM Algorithm

a. Linear (SVM)

If your data can be neatly divided into two categories in two dimensions using a straight line, then linear support vector machines are a good fit. A simple straight line can do the trick in many situations, classifying the data points neatly. While linear support vector machines (SVMs) are great at discovering hyperplanes that maximally divide classes, they could struggle with data that isn't entirely linearly separable. To deal with increasingly complicated data distributions, nonlinear support vector machines (SVMs) equipped with kernel functions can be employed when the data cannot be separated linearly.

b. Non- Linear (SVM)

When working with 2D data or other types of data that do not lend themselves well to linear classification, Non-Linear Support Vector Machines can be an invaluable tool. Due to the rarity of linearly separable data points in practical applications, more sophisticated methods such as kernel tricks are required. By projecting the data into a higher-dimensional space, the kernel approach makes linear separation easier, enabling Non-Linear SVM to handle complex data distributions. Because of this method, SVM is able to successfully categorize data in higher-dimensional spaces that would be extremely difficult, if not impossible, to divide using a straight line.

3.4.5 How (SVM) Work

While the theory behind Support Vector Machines (SVMs) is often dense, the practical application of these models is often quite simple, making them useful in many domains. Support vector machines (SVMs) work by essentially trying to identify the best decision boundary—usually a hyperplane—that reliably divides data points into their corresponding classes. A description of SVM operation is as follows: Transforming Data: The support vector machine (SVM) starts with data in a two-dimensional space and raises it to a three-dimensional one. Finding a decision boundary that adequately partitions the data will be

made easier with this change. The main goal of Support Vector Machines (SVMs) is to maximize the margin, which is the distance between the decision boundary and the nearest data points. Finding the decision boundary (hyperplane) that does this is the main focus of SVMs. A Support Vector Classifier is the name given to this border. Within the soft margin, support vectors are data points that are closer to the decision boundary. Parameter C, which governs the trade-off between obtaining a wider margin and tolerating some misclassification (soft margin), is introduced by support vector machines (SVM) [146]. While a bigger C value aims to minimize misclassification, it may lead to a reduced margin; conversely, a smaller C value emphasizes a larger margin but allows some misclassification.

Kernel Trick: Support vector machines (SVMs) are very good at both linear and non-linear data processing. This is accomplished by use of a method known as the kernel trick, which mechanically turns the data into a higher-dimensional space without intentionally doing so. Separating data points in this higher-dimensional space is made easy by the kernel function, which computes the relationships between them. To get the most out of support vector machines (SVMs), it's important to pick the correct kernel function and set the parameters for it. The features of the data can dictate the choice of kernel, which can be either linear, polynomial, sigmoid, or radial basis function (RBF). [139] For best performance, it's also important to tweak the kernel parameters. Support vector machines (SVMs) are a powerful tool for a wide range of linear and non-linear classification tasks due to their ability to manage the soft margin parameter C, choose the right kernel, and fine-tune kernel parameters, all of which reduce error rates and overfitting.

3.5 METAHEURISTIC OPTIMIZATION ALGORITHMS

3.5.1 Definitions

Metaheuristics have gained popularity as an efficient approach to tackle challenging optimization problems. These problems often cannot be solved optimally within a reasonable time frame using deterministic methods. Let's break down this concept: Optimization Problems: Optimization problems involve finding the best solution among a set of possible choices or solutions. These problems can arise in various fields, such as engineering, logistics, economics, and more. The goal is to determine the solution that optimizes a specific objective or criterion, such as minimizing costs, maximizing profits, or achieving the highest

efficiency. **Challenging Optimization Problems:** Some optimization problems are particularly difficult because they involve a large number of variables, constraints, or complex relationships. In many cases, finding the optimal solution using deterministic methods, which rely on exact algorithms, becomes impractical due to the extensive computational resources and time required. **Metaheuristics:** Metaheuristics are a class of problem-solving techniques that provide approximate solutions to optimization problems. Unlike deterministic methods, which guarantee an optimal solution, metaheuristics focus on finding good or near-optimal solutions within a reasonable time frame. They are particularly well-suited for complex optimization problems where exact solutions are elusive. **Efficiency and Time Constraint:** Metaheuristics are known for their efficiency in exploring large solution spaces and quickly converging to satisfactory solutions. They are designed to balance the trade-off between solution quality and computational effort. As a result, they can handle optimization problems that would be infeasible to solve optimally using deterministic methods, especially when time constraints are a concern [147].

3.5.2 Metaheuristic Search Algorithm

Glover in (1986) [148] introduced the term "metaheuristic" in 1986. It represents a category of promising algorithms that cleverly combine multiple principles to navigate and leverage the search space. Techniques for efficient solution generation were applied to uncover solutions that approach optimality effectively (Osman and Laporte, 1996). [149] "Meta-" implies "above" or "at a higher level," while "heuristic" denotes "to uncover" or "to find through experimentation [150]." In essence, it's a strategy for directing the search process to systematically explore the search space and identify solutions that are close to optimal. Metaheuristic algorithms have found widespread application in addressing complex optimization challenges by incorporating governing mechanisms inspired by strategies observed in nature, social behavior, biological, or physical processes, aiming to uncover near-optimal solutions within the search space. Over the past two decades, a diverse range of metaheuristic algorithms has emerged, gaining increasing popularity.

3.5.3 Introduction to Metaheuristic Optimization Algorithm

To solve complex, NP-hard, high-dimensional, non-linear, and hybrid problems, conventional and traditional optimization methods are ineffective. Meta-heuristic algorithms

are widely employed in many different domains and have seen a sharp growth in their use in recent years for such challenges. In contrast to precise optimization approaches, these algorithms seek out solutions that are as near to the global optimum as they can be while still satisfying the threshold constraint to a tolerable degree. The metaheuristic algorithm optimizes by determining the initial population of individuals in the search space at random based on some solution description. Because each person represents a solution, the fitness of the population as a whole is then assessed. Following that, a new population is created by disrupting solutions in the old population via evolutionary processes like crossover and mutation. As a result, the search parameter is iteratively enhanced until it fulfills the stopping requirement in terms of iterations or time spent. The procedure will iterate in order to achieve the final global minimum or maximum values that satisfy the relevant objective function. A metaheuristic algorithm may produce a near optimal solution rather than the actual or precise solution. In general, any metaheuristic algorithm's optimization approach (Khajezadeh 2011) [151]. Because of their numerous advantages, metaheuristic algorithms are commonly utilized in solving real-world optimization issues. When the problem's complexity grows and there are numerous viable solutions, the metaheuristic method is frequently selected due to its great efficiency and flexibility (Talbi 2009) [152]. The ease of implementation aids in the resolution of the problem in a short period of time. There are three basic components to every optimization problem: the choice variables, the constraints, and the objective function. In order to optimize the value of the objective function, optimization seeks to decide on the values for the design variables while taking the constraints into account. There are two types of optimization techniques: deterministic and stochastic approaches. Deterministic techniques of two types, gradient-based and nongradient-based, perform well in solving linear, convex, continuous, differentiable, and low-dimensional optimization problems Xue et al [153]. Deterministic techniques, on the other hand, fall short when it comes to non-linear, nonconvex, discontinuous, non-differentiable, and high-dimensional optimization issues. Deterministic approaches produce poor solutions in this type of optimization problem by becoming stuck in local optimal Mirjalili et al [154]. The three main uses of metaheuristic approaches are to solve problems quickly, handle enormous issues, and create algorithms that are more reliable. These techniques are adaptable, easy to use, and simple to design [152]. In general, metaheuristic algorithms mimic natural events by combining rules and randomization.

3.5.4 Types of Metaheuristic Algorithm

Meta-heuristic algorithms are widely grouped into four categories: Ancient inspired algorithms, Trajectory-based algorithms, Nature inspired algorithms, Evolutionary-based algorithms and art-inspired algorithms which was the first algorithm of this type in 2020 such as Color Harmony Algorithm (CHA) and Stochastic Paint Optimizer (SPO) they are color-based. In this study we used a novel nature-inspired algorithm in 2022 called Pelican optimization algorithm (POA) with another novel algorithm called Average and Subtraction Based Optimizer (ASBO). The figure below shows the popular type of metaheuristic algorithm. Figure 3.9 provides the type of Metaheuristics Optimization Algorithm.

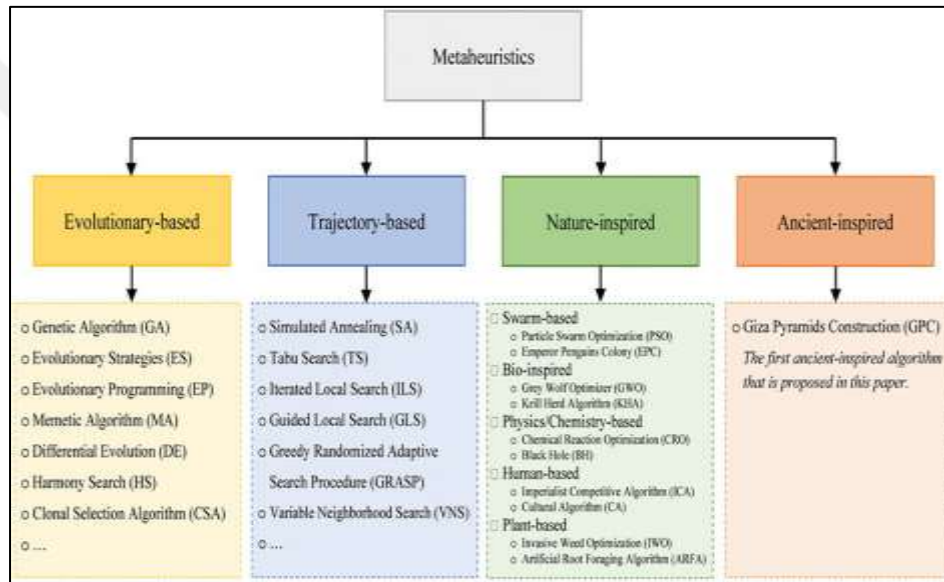


Figure 3.9: Metaheuristics Algorithm Categories.

3.6 PELICAN OPTIMIZATION ALGORITHM (POA)

The Pelican Optimization Algorithm (POA) is a novel stochastic optimization algorithm introduced by Trojovský et al. [155]. This algorithm draws its inspiration from the natural hunting behavior of pelicans, a species of large birds known for their distinctive physical features, social behavior, and hunting techniques [155]. Pelicans are characterized by their sizable bodies, long beaks, and a spacious throat pouch used for capturing and consuming prey. figure 3.10 shows the pelican bird. They are social creatures that often form flocks consisting of several hundred pelicans. These birds predominantly feed on fish and frequently engage in group hunting activities. The hunting process of pelicans involves a remarkable strategy. They first detect the location of their prey and then dive from heights

ranging between 10 to 20 meters, although some species may opt for lower altitudes. Upon reaching their prey, pelicans extend their wings over the water's surface, herding the fish toward shallower waters for easier capture. When a pelican successfully catches a fish, its beak takes in a substantial volume of water, propelling the bird forward. This action expels the excess water from its beak before swallowing the captured fish [[156]].The intelligent and effective hunting behavior exhibited by pelicans serves as the foundational inspiration for the development of the Pelican Optimization Algorithm (POA).

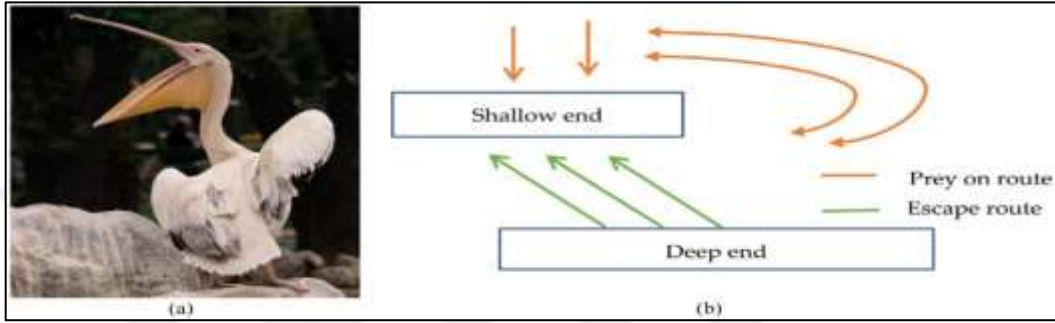


Figure 3.10: Pelican Bird.

3.6.1 Mathematical Model of POA

Each node in the POA algorithm's population stands in for a possible answer to the optimization issue. One example of a population-based algorithm is POA. It works by asking each member of the population to suggest a value for an optimization problem variable according to where they are in the search space. Using the top and lower bounds of the problem, we randomly assign values to population members using Equation (1) to initialize the population.

$$x_{i,j} = L_j + \text{rand.} (u_j - L_j), i = 1, 2, \dots, N, j = 1, 2, \dots, m_1 \quad (3.7)$$

The pelican population is represented in Equation (2) by the population matrix. Potential solutions are represented by the rows of this matrix, while the columns provide suggested values for the problem variables. N is the population size, m is the number of problem variables, rand is a random number between 0 and 1, l_j is the lower bound for the j th variable, and u_j is the upper bound in the optimization problem.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N \times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,m} \end{bmatrix}_{N \times m} \quad (3.8)$$

The population matrix of pelicans is denoted as X , and each individual pelican is represented by X_i , where i refers to the i th pelican in the population. Each pelican corresponds to a potential solution for the given problem. Therefore, the objective function of the given problem can be evaluated for each of these potential solutions. The values of the objective function are computed using a vector referred to as the objective function vector, as shown in Equation (3).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N \times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N \times 1} \quad (3.9)$$

In Equation (3), F_i represents the objective function value associated with the candidate solution i , while F represents the entire objective function vector. The POA algorithm updates candidate solutions by emulating the techniques and behaviors of pelicans during prey hunting and attack. This emulation occurs in two distinct stages:

a. Approaching the prey (discovery phase)

Pelicans will first fly to the spot where they think their prey is hiding. The POA algorithm is able to efficiently explore different parts of the search space by mimicking this method. The unique thing about POA is that the prey's position in the search space is decided at random. In the exact search for the problem-solving space, this increases POA's exploration capability, as shown in Equation (4).

$$x_{i,j}^{p_1} = \begin{cases} x_{i,j} + \text{rand} \cdot (p_j - I \cdot x_{i,j}), & F_p < F_i; \\ x_{i,j} + \text{rand} \cdot (x_{i,j} - p_j), & \text{else}, \end{cases} \quad (3.10)$$

In the j -th dimension, the new position of the i th pelican as determined by phase 1 is $x_{i,j}^{p_1}$, the location of the prey is denoted by p_j , and the value of the prey's objective function is F_p . There are two possible values for the parameter that I can choose at random. Every iteration and member choose their own set of values. Increasing the value of this parameter to two causes a more substantial displacement, which may take a member to previously uncharted areas of the search space. In order for the POA to explore the search space effectively, Parameter I has a significant impact on its exploration capability. If the value of the goal function increases, then the pelican's new position is accepted by the proposed POA. In order to keep the algorithm from straying into less-than-ideal regions, this

form of updating is called effective updating. The model of the process is shown in Equation (5).

$$X_i = \begin{cases} X_i^{P_1}, & F_i^{P_1} < F_i; \\ X_i, & \text{else,} \end{cases} \quad (3.11)$$

Where $X_i^{P_1}$ is the i th pelican's new status and $F_i^{P_1}$ is its phase 1 objective function value.

b. Winging on the Water Surface (Exploitation Phase)

Pelicans bring their prey to the surface of the water, where they can catch it in their neck pouch by extending their wings and driving the fish higher. This tactic enhances the quantity of fish captured by pelicans within the designated region. The proposed POA hopes to improve its local search and exploitation capabilities by mimicking this pelican's behavior and convergent on better hunting spots. If the algorithm wants to get better results, it needs to look at the areas around the pelican's present position. A mathematical model of pelican hunting behavior is given by Equation (6).

$$x_{i,j}^{P_2} = x_{i,j} + R \cdot \left(1 - \frac{t}{T}\right) \cdot (2 \cdot \text{rand} - 1) \cdot x_{i,j}, \quad (3.12)$$

This is where the constant variable R , denoted as $x_{(i,j)}^\wedge$, is set to 0.2. The neighborhood radius of $x_{(i,j)}$ is represented by $R \cdot (1-t/T)$, the maximum number of iterations is T , the i th pelican's new location in the j th dimension is denoted by (P_2) , and phase 2 determines the current iteration. In order for the population as a whole to find better solutions, the neighborhood radius defines the search region around each individual. In order to get closer to the global optimal solution, this parameter affects the exploitation power of the POA. The initial iteration has a broader search region surrounding each member since this coefficient has a greater value. The neighborhood radii of each member get smaller as the iterations continue because the coefficient $R \cdot (1-t/T)$ diminishes. Convergence to solutions closer to the global optimum is facilitated by the POA's ability to scan the vicinity of each population member efficiently in fewer, more accurate stages. During this stage, the new pelican position might be accepted or rejected by effectively modifying Equation (7).

$$X_i = \begin{cases} X_i^{P_2}, & F_i^{P_2} < F_i; \\ X_i, & \text{else,} \end{cases} \quad (3.13)$$

A pelican's objective function value according to Phase 2 is denoted by $F_i^\wedge(P_2)$, and the i th pelican's new position is represented by $X_i^\wedge(P_2)$. Taking into account the changed

population placements and objective function values, the best candidate solution that has been obtained so far is updated after each member of the population has undergone first and second phase updates. After that, the algorithm keeps running through the several steps of the proposed POA, as shown in Equations (4)-(7), until everything is finished. In the end, a near-optimal solution to the problem is generated by utilizing the best candidate solution that the algorithm found during its iterations.

Figure 3.11 shows a flow diagram of POA, and figure 3.12 show the pseudo-code.

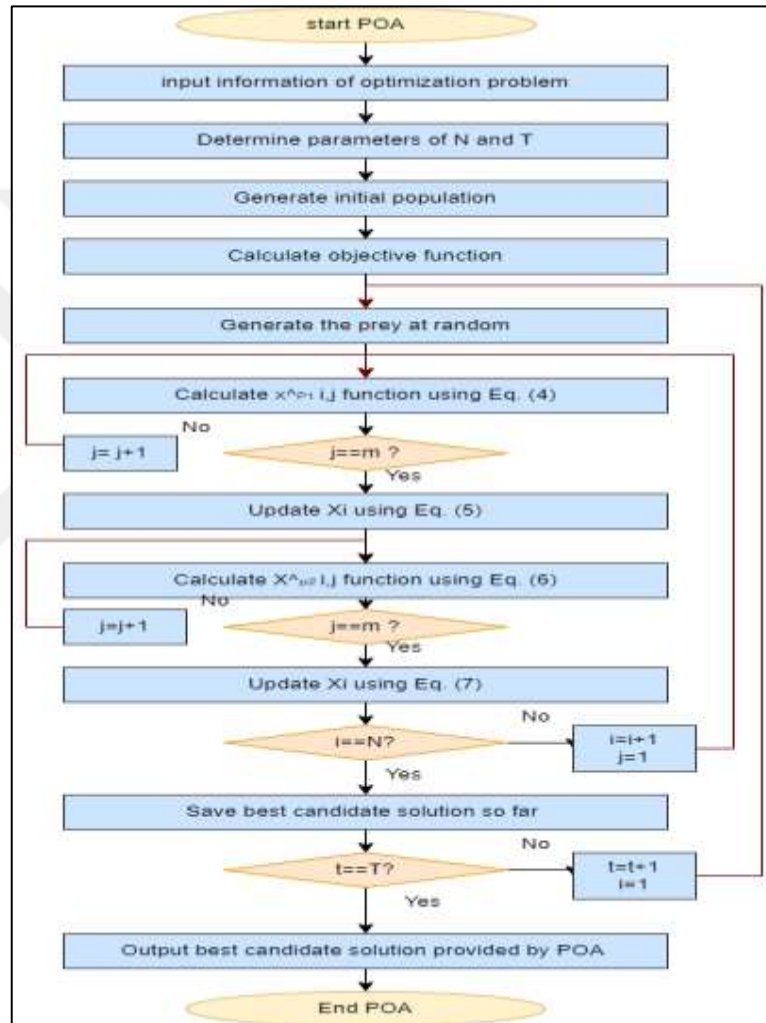


Figure 3.11: Flowchart of (POA).

Algorithm 1. Pseudo-code of POA.

```

Start POA.
1.   Input the optimization problem information.
2.   Determine the POA population size (N) and the number of
iterations (T).
3.   Initialization of the position of pelicans and calculate the
objective function.
4.   For t = 1: T
5.   Generate the position of the prey at random.
6.   For I = 1: N
7.   Phase 1: Moving towards prey (exploration phase).
8.   For j = 1:m
9.   Calculate new status of the jth dimension using Equation (4).
10.  End.
11.  Update the ith population member using Equation (5).
12.  Phase 2: Winging on the water surface (exploitation phase).
13.  For j = 1: m.
14.  Calculate new status of the jth dimension using Equation (6).
15.  End.
16.  Update the ith population member using Equation (7).
17.  End.
18.  Update best candidate solution.
19.  End.
20.  Output best candidate solution obtained by POA.
End POA.

```

Figure 3.12: Pseudo-Code of POA.**3.6.2 (POA) Mathematical Complexity**

Algorithm initiation, fitness function evaluation, prey production, and solution updating are the four basic elements upon which the algorithm's mathematical complexity is founded. The computational complexity of each of these aspects is broken down as follows: $O(N)$ is the complexity of the algorithm's initialization, where N is the population size. The objective function is evaluated in both stages by every member of the population at each iteration. The result is an $O(2TN)$ computational complexity, where T is the maximum iteration number and N is the population size. At every cycle, the process of producing and assessing prey is carried out. When T is the maximum iteration and m is the number of issue variables, the complexity of constructing prey is $O(T \text{ plus } O(Tm))$. There should be two updates to the count of N members with m dimensions in each cycle. A computational complexity of $O(2TNm)$ is the outcome of this. Taking everything into account, the overall computing complexity of the suggested POA is given by $O(N + T(1 + m)(1 + 2N))$.

3.7 MAYFLY OPTIMIZATION ALGORITHM (MA)

Zervoudakis K et al. [157] developed a novel approach to optimization problems called the Mayfly Algorithm (MA). This suggested algorithm merges the best features of swarm intelligence with evolutionary algorithms, drawing inspiration from the mating habits and flight patterns of mayflies. Figure 3.13 shows the Mayfly.

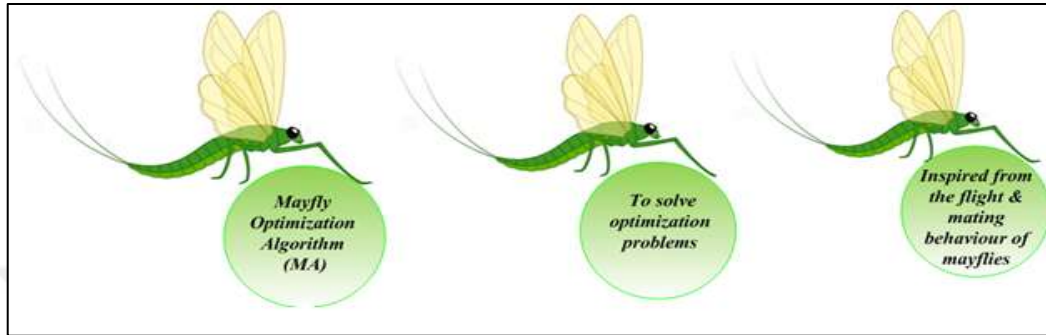


Figure 3.13: The Mayfly.

3.7.1 Introduction to the (MA) Algorithm

There are over 3000 species of mayflies across the globe. They are insects that fall under the order Ephemeroptera. May is a popular month for them in the UK, which is where their name comes from. Mayflies spend several years as aquatic nymphs before emerging as adults, and their visibility is low to the human eye during this time. A mayfly's principal goal during its brief adult life phase—which lasts just a few days—is reproduction (Dominguez E et al. [158]). Mayflies swarm a few meters above the water when they're adults, and they conduct a special up-and-down dance to attract females. In order to mate while still in the air, females swoop down on these swarms. The female lays her eggs on the water's surface after mating, which is often a short process (Allan & Peckarsky et al [159], [160]). The suggested optimization method incorporates aspects of Particle Swarm Optimization (PSO), Genetic Algorithms (GA) (Goldberg D et al. [161]), and Firefly Algorithm (FA) (Yang x et al. [162]), taking cues from the mating behavior of mayflies. It is similar to PSO but with a few key differences (Kennedy J et al. [163]). The algorithm treats mayflies as adults the moment they emerge from their eggs, and it disregards life expectancy in favor of the fittest. A strong hybrid algorithmic framework is provided by this method. The optimization issue has many possible solutions, each of which is represented by a mayfly's location in the search space. Two populations, one for males and one for females, are generated at random by the

algorithm. The performance of each mayfly is assessed using the preset objective function $f(x)$. At the beginning, each mayfly is introduced into the problem space as a possible solution, denoted as a d -dimensional vector $x=(x_1,...,x_d)$. Each mayfly's flight path is affected by an ever-changing interplay between its own optimal position (p_{best}) and the best position that any mayfly in the swarm has attained thus far (g_{best}). The change in position of a mayfly is governed by its velocity, $v=(v_1,...,v_d)$. Figure 3.14 shows the inspiration of Mayfly.

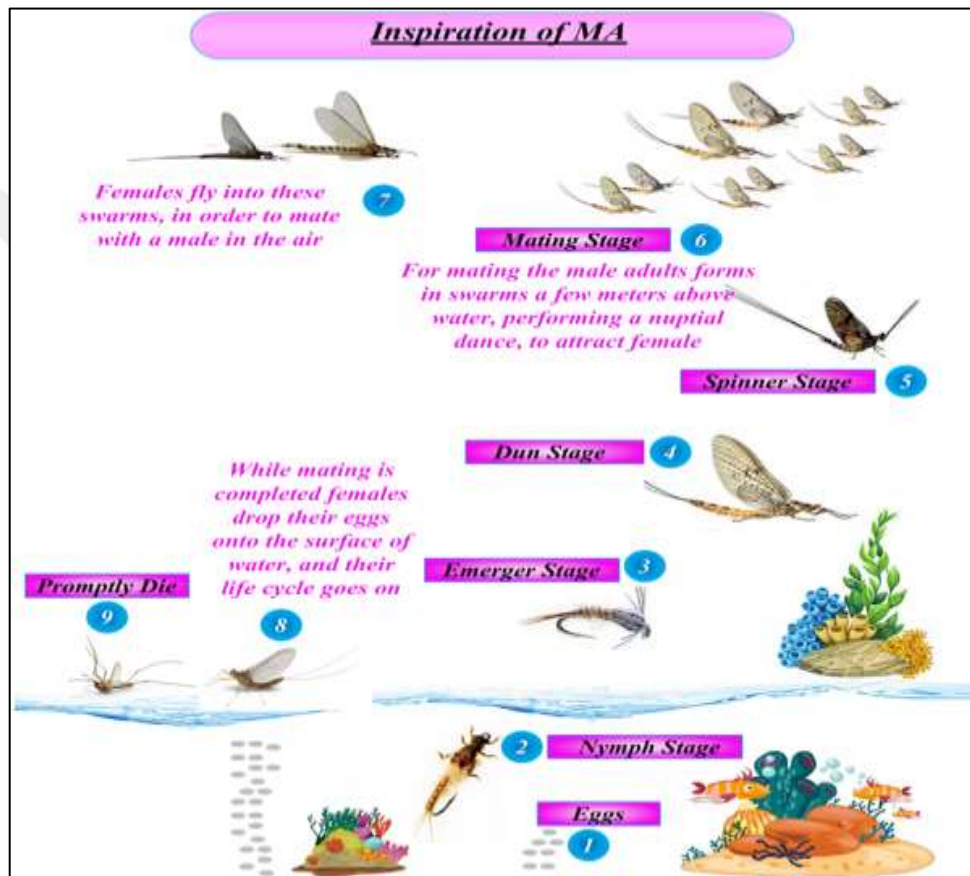


Figure 3.14: Inspiration of (MA).

3.7.2 Flowchart and Pseudo Code of (MA)

Figure 3.15 provides the flowchart of Mayfly algorithm and figure 3.16 shows the pseudo code that shows the most important basic steps of the mayfly (MA) algorithm mechanism.

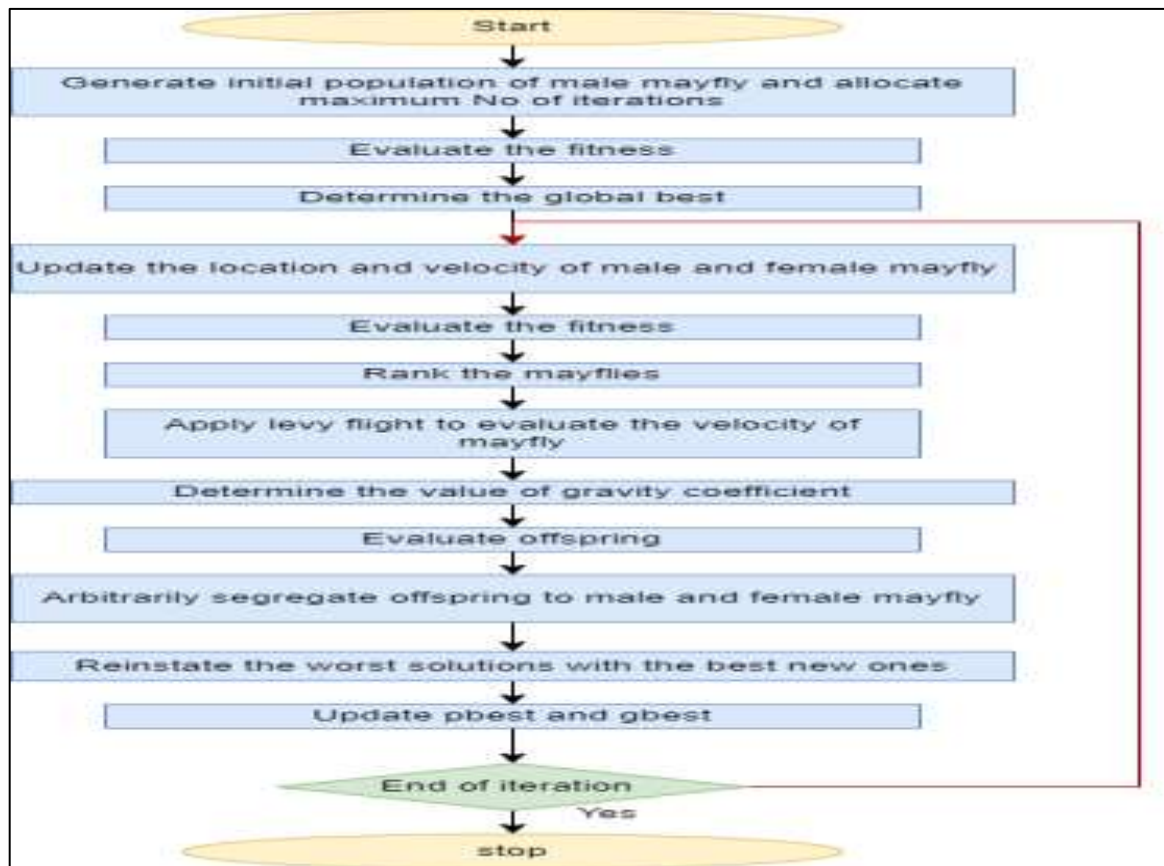


Figure 3.15: Flowcharts of (MA).

Mayfly Algorithm

Objective function $f(x)$, $x = (x_1, \dots, x_d)$

)

T

Initialize the male mayfly population and velocities x_i ($i = 1, 2, \dots, N$) v_{mi}

Initialize the female mayfly population and velocities y_i ($i = 1, 2, \dots, M$) v_{fi}

Evaluate solutions.

Find global best $gbest$.

Do While stopping criteria are not met.

Update velocities and solutions of males and females

Evaluate solutions.

Rank the mayflies.

Mate the mayflies.

Evaluate offspring.

Separate offspring to male and female randomly

Replace worst solutions with the best new ones.

Update and $pbest$ $gbest$

end while

Postprocess results and visualization.

Figure 3.16: Pseudo Code of (MA).

4. PROPOSED METHOD

4.1 INTRODUCTION

Tea is a vital staple crop globally, serving as a significant food source for billions of people. However, tea leaf diseases pose a substantial risk to both the quality and yield of tea crops. Timely detection of these diseases is essential to mitigate potential losses in crop yield and quality. Manual inspection, the old method of disease detection in tea leaves, is laborious and error prone. A potential new approach to early disease detection in tea leaves has arisen in recent years: the application of image processing tools. With the use of image processing, tea leaf diseases can be automatically detected and diagnosed, which might greatly cut down on the time and effort needed for disease identification. Figure 4.1 shows the way of extraction feature by CNN.

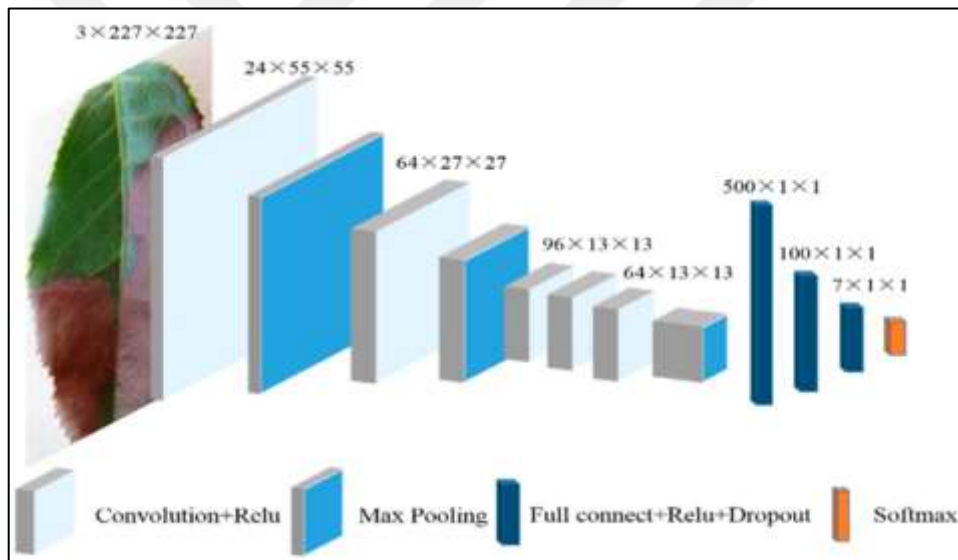


Figure 4.1: CNN Extraction of Features from the Leaf Image.

A number of critical phases make up the suggested method. The process begins with taking digital photos of the tea leaves. The next steps involve preprocessing the photos, extracting features, and finally classifying them. Many different methods exist for improving the quality of images, and they are all part of image preprocessing. Finding and removing relevant attributes from the preprocessed pictures is what feature extraction is all about. The Mayfly Algorithm (MA) and the Pelican Optimization Algorithm (POA) operate along with Convolutional Neural Networks (CNN) in this approach to extract texture data from photos. A Support Vector Machine (SVM) is used as the classifier to categorize the diseases

affecting tea leaves. When it comes to detecting diseases in tea leaves, this method could be a game-changer. For the early detection of illnesses affecting tea leaves, it provides a reliable, accurate, and quick technique. The proposed technology enables farmers to make educated decisions about crop management by promoting rapid and accurate disease identification. The method's potential to change the tea business is highlighted by the fact that it can lead to improved crop yields and enhanced food security.

4.1.1 Scope of the Project

An effective and precise system for the automated identification and categorization of diseases affecting tea leaves in particular is the goal of this study utilizing image processing for plant leaf disease detection in general. The project's end goal is to enhance food security by boosting crop yields through better disease diagnostics of tea leaves, which would help farmers detect and control infections in a timely manner.

- a. To detect tea leaf disease portion from image.
- b. To extract features of detected portion of leaf.
- c. To recognize detected portion of leaf through SVM (Support Vector machine)

4.2 TEA LEAF DISEASES DETECTION

The existing system for tea leaf disease detection typically involves manual visual inspection by trained professionals in the field, who visually examine the leaves for any signs of disease, such as discoloration, necrosis, or lesions. This method is time-consuming, subjective, and prone to errors and misdiagnosis, especially for early stages or subtle symptoms of diseases. Moreover, it requires expertise and resources, which may not be available or accessible to all farmers or regions.

Therefore, there is a growing interest in developing automated and objective systems for tea leaf disease detection using image processing techniques. These systems involve capturing digital images of tea leaves using cameras or smartphones, pre-processing and enhancing the images, segmenting the regions of interest, extracting relevant features and patterns, and classifying the images into healthy or diseased categories using machine learning algorithms. Some existing systems use color-based segmentation, texture analysis, or Support vector machine (SVM), to achieve high accuracy and robustness in disease detection. However, these systems also face challenges, such as image quality, lighting, background noise, and

dataset bias, which require careful consideration and optimization. Figure 4.2 explains the general steps of proposed system.

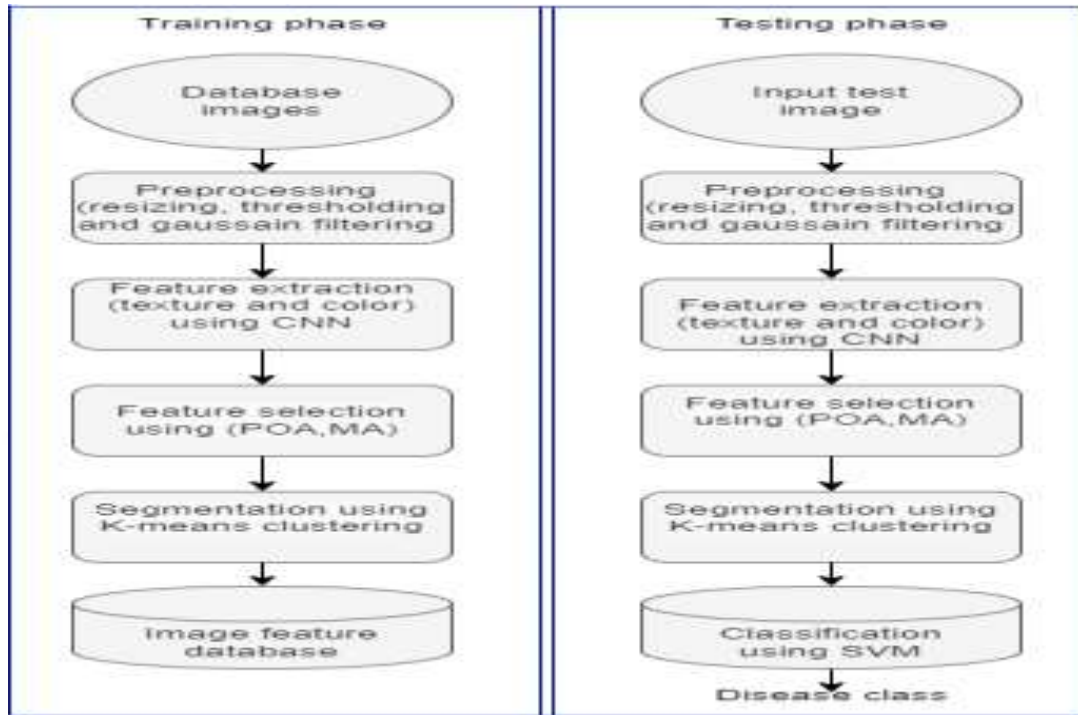


Figure 4.2: General Outline Diagram of the Proposed Method.

The manual inspection system for tea leaf disease detection has several limitations, including low efficiency, low accuracy, and subjective interpretation. It also requires skilled labour, which may not be available in remote or underdeveloped areas. Moreover, it is difficult to detect diseases at an early stage when visual symptoms are not yet apparent. These factors can lead to delayed treatment, increased crop damage, and decreased yield. To address these limitations, various automated systems for tea leaf disease detection have been developed using image processing techniques. These systems can quickly and accurately identify the presence of diseases in tea leaves, allowing for early intervention and effective disease management. The automated system can also reduce the need for skilled labour and facilitate rapid and objective decision-making.

Many of the current systems designed for tea leaf disease detection rely on color-based segmentation methods to separate the regions of interest from the background. These methods take advantage of the fact that healthy and diseased leaves often exhibit distinct color variations, making it possible to differentiate between them. Other systems employ texture analysis techniques to extract texture-related features from the images, which aid in

distinguishing between healthy and diseased leaves. Additionally, deep learning approaches, including Support Vector Machines (SVM), have been utilized for image classification, categorizing images as healthy or diseased. However, these systems confront several challenges, including variations in image quality, fluctuations in lighting conditions, and the influence of environmental factors. Furthermore, the system's performance can be influenced by factors such as the size and quality of the dataset used to train machine learning algorithms. Consequently, it is imperative to carefully consider these factors when developing a robust and effective automated system for tea leaf disease detection. In summary, the existing systems that employ image processing techniques for tea leaf disease detection have demonstrated promising outcomes in terms of accuracy, efficiency, and objectivity. Nevertheless, further research is essential to fine-tune these systems for real-world applications and to tackle the challenges they encounter.

4.3 IMAGE SEGMENTAION

Image segmentation using active contour is a technique used to separate objects in an image from its background. It works by assigning energy values to each pixel and then utilizing the principles of calculus and optimization techniques to determine the optimal path or boundary around the object of interest. This method can be applied to any type of image including medical imaging, satellite imagery, aerial photography, etc., allowing for automatic extraction and analysis of regions-of-interest within images with high accuracy. Active contours are particularly useful when dealing with complex shapes that may not be easily separated using traditional methods such as thresholding or edge detection algorithms. Figure 4.3 shows the classes of input.

Classes	Samples			
Anthracnose				
Leaf blight				
Blight disease				
Tea wheel spot disease				
Tea white star disease				
Tea coal disease				
Mechanical damage				

Figure 4.3: Classes of the Input Images.

4.3.1 Feature Extraction

Feature extraction is the process of taking raw data and extracting meaningful features from it. It involves identifying important characteristics or patterns in a dataset that can be used to summarize or represent the data more effectively. Feature extraction techniques are commonly used for tasks such as image recognition, text analysis, natural language processing, and machine learning models. By reducing the dimensionality of a dataset through feature extraction methods like Pelican Optimization Algorithm (POA), we can make our models faster to train while still obtaining accurate results. Using the Pelican Optimization Algorithm (POA) to reduce the number of features extracted by a Convolutional Neural Network (CNN) and find optimal features is an unconventional

approach. CNNs are typically designed to automatically learn and extract relevant features from the data during the training process. Feature selection or dimensionality reduction is a common step in the design of CNN architectures:

- a. **Feature Extraction by CNN:** Start by training your CNN to extract a set of features from the input data. Typically, the output of a CNN consists of feature maps from different convolutional layers, representing hierarchical features learned by the network.
- b. **Define a Feature Selection Objective:** You need to define an objective function that quantifies the importance of different features extracted by the CNN. This objective function should evaluate the relevance of features to your specific task. This could be a classification task, regression, or any other problem.

Population Initialization: Initialize a population of feature selection solutions. Each solution represents a subset of features extracted by CNN. You can represent a solution as a binary vector where each element corresponds to a feature extracted by the CNN.

POA Iterations: During each iteration of the POA, you can apply the following steps:

- a. **Evaluate the fitness of each solution in the population based on the defined objective function.** Fitness measures how well a feature subset contributes to the task's performance.
- b. **Update the population based on the POA's optimization process.** Pelicans in the algorithm can mimic the exploration and exploitation of different feature subsets.
- c. **Optionally, incorporate mechanisms for selection, crossover, and mutation to evolve the feature subsets within the population.**
- d. **Termination Criteria:** Define a termination criterion for the POA, such as a maximum number of iterations or reaching a desired level of performance improvement.
- e. **Retrieve Optimal Feature Subset:** After the POA completes, the feature subset with the best fitness value in the population represents the selected features that the algorithm has determined to be most relevant for your task.
- f. **Integration with CNN:** Finally, you can modify your CNN architecture to incorporate only the selected features, effectively reducing the number of features used by CNN for your task.

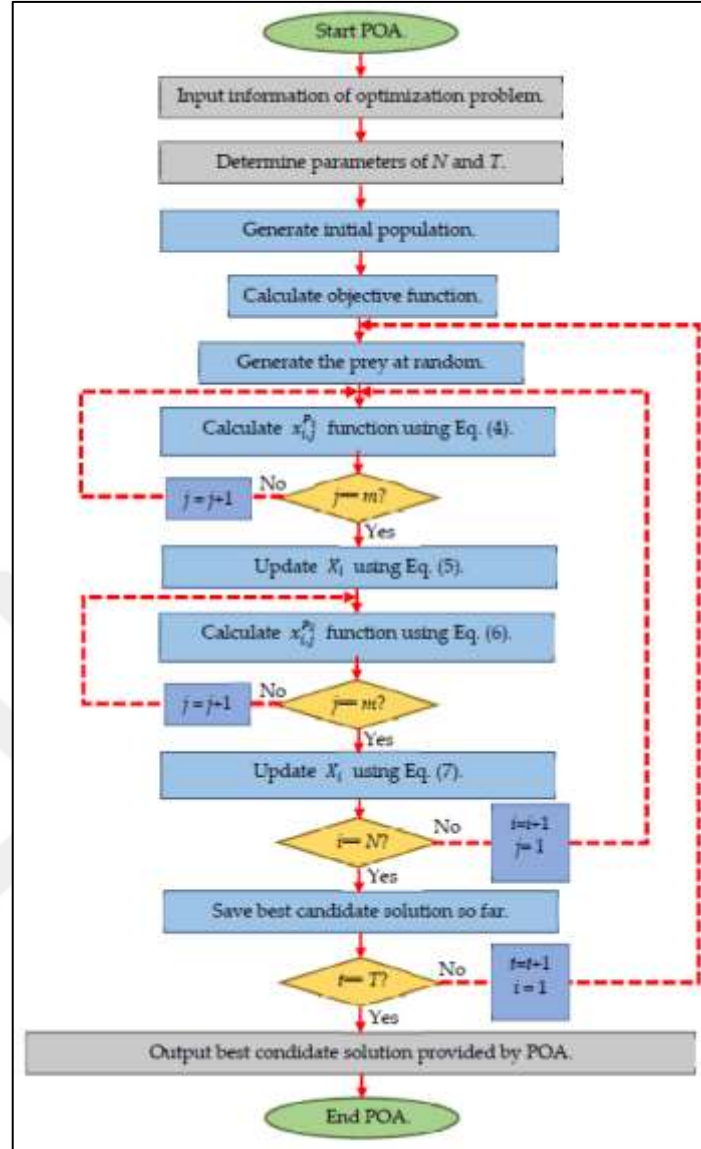


Figure 4.4: Diagram of the POA Algorithm.

4.3.2 Feature Selection

Feature selection utilizing the Mayfly Algorithm (POA) is a robust approach for diminishing data dimensionality and identifying the most crucial features within a dataset. This technique functions by identifying linear combinations of existing variables that account for a substantial portion of the dataset's variability. Through the elimination of redundant or irrelevant features, it can mitigate noise and enhance model performance. Additionally, feature selection using POA has the capacity to reveal interactions among various variables that might otherwise remain concealed. This methodology presents an effective means of choosing significant features from extensive datasets while simultaneously preserving their

inherent structure.

4.3.3 Mayfly Algorithm in Feature Clustering

A bio-inspired optimization method that mimics the behavior of mayflies is known as the Mayfly method (MA), as previously mentioned. A critical stage in the machine learning pipeline, feature selection, can be accomplished using MA within the framework of a Convolutional Neural Network-Phenotypic Observation Algorithm (CNN-POA) structure that is used to detect illnesses in plant leaves. The goal of feature selection is to isolate the input factors that have the greatest bearing on the dependent variable of interest. In the case of CNN, which is typically used for image data like plant leaves, feature selection is not as straightforward as with structured data because CNNs inherently perform a form of feature extraction through their convolutional layers. However, the MA can still play a role in optimizing the architecture or the hyperparameters of the CNN to improve its performance.

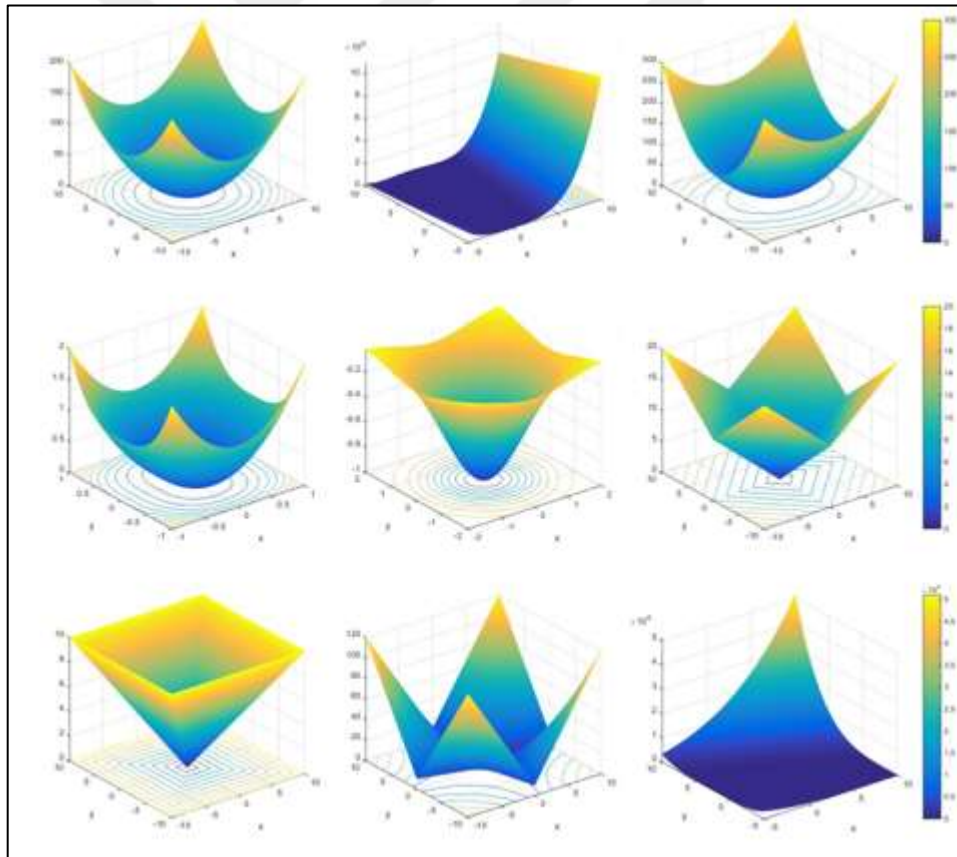


Figure 4.5: MA Clustering of POA Features.

in practice, incorporating the Mayfly Algorithm into the CNN-POA structure would involve running the MA alongside the training process of the CNN. Each iteration (generation) of

the MA would propose new hyperparameters or configurations, which would then be applied to the CNN. The CNN is trained and validated with these settings, and the resulting performance informs the fitness of the mayflies. Over successive generations, the MA navigates the search space, ideally converging to a set of optimal features or parameters for the task of disease detection in plant leaves. Figure 4.5 shows how MA clustering of POA features.

4.4 SVM ALGORITHM

Support Vector Machines (SVM) represents a supervised machine learning classification algorithm employed for categorizing data into two or more distinct classes. The fundamental principle involves delineating a hyperplane amid the various classes and assigning new instances based on their relative position to this plane. SVMs exhibit exceptional efficacy when dealing with scenarios marked by a distinct separation margin between diverse classes and when ample training data is accessible. These algorithms have found extensive utility across multiple domains, including image recognition, text categorization, bioinformatics, and handwritten character recognition.

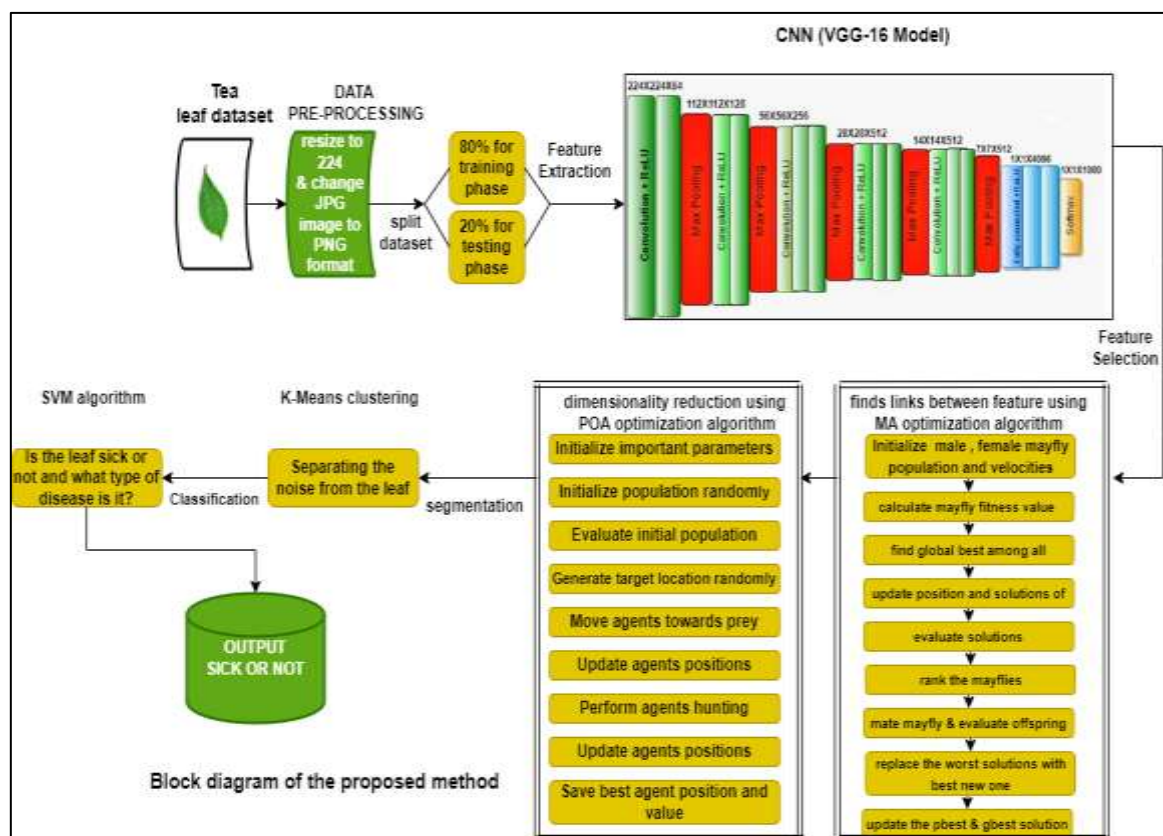
4.5 DATASET DESCRIPTION

Typically, digital photographs of diseased tea leaves are included in the dataset utilized for the project of disease detection via image processing. Images of healthy tea leaves may also be included in the collection for validation and comparison reasons. Depending on the project requirements, the dataset size can vary; nonetheless, a more diverse and larger dataset will enhance the detection system's accuracy and robustness. Many places, including research institutions, agricultural groups, and online repositories, make the dataset available. Make that the dataset is high-quality, has proper labels, and pertains to the research subject at hand.

5. IMPLEMENTATION AND RESULTS

5.1 METHODOLOGY

Feature extraction using Convolutional Neural Networks (CNN) is the first step in the process of tea leaf image categorization. A convolutional neural network (CNN) model, like ResNet or VGG-16, is fine-tuned using the tea leaf dataset. Each picture gets a feature vector of a fixed length derived from the essential visual features extracted by the CNN. After that, we'll use Principal Component Analysis (POA) to pick features and reduce dimensionality. To decrease the dimensionality of characteristics while keeping the most informative ones, principal component analysis (PCA) is used. The steps involved in this procedure are as follows: data standardization, eigen decomposition, computation of the covariance matrix, and feature projection onto chosen principal components. Following this, the MA algorithm seeks to discover feature-to-feature correlations and the ways in which they interact with one another. As part of its classification task evaluation, it sorts features according to their impact and ranks features based on the strength of their linkages with one another. Then, for the classification, we use the Support Vector Machine (SVM). Using the characteristics acquired during POA and MA, the SVM model is trained to find the best decision boundary that differentiates between various types and qualities of tea. It is possible to discover the optimal kernel function by experimenting with different ones. Using the features that were extracted, we test the model on the testing dataset to see how well it predicts the kind and quality of tea. To achieve accurate and efficient tea leaf picture classification, this methodology incorporates CNN feature extraction, dimensionality reduction with POA, feature relationship finding with the MA algorithm, and SVM classification. Future research can investigate additional improvements and potential uses, but it already has real-world ramifications for the tea business and comparable image-based classification jobs. Figure 5.1 shows a flowchart of proposed disease detection.



5.1.1 Hardware and Software Specifications

MATLAB is a sophisticated programming language and interactive platform designed for numerical computations, data visualization, and software development. It empowers users to analyze data, formulate algorithms, and construct models and applications. MATLAB offers an extensive array of mathematical tools encompassing linear algebra, statistics, Fourier analysis, filtering, optimization, and beyond. Additionally, it incorporates a user-friendly graphical interface for seamless data visualization and result representation. MATLAB finds widespread application in engineering, scientific research, and financial sectors, where it serves as a valuable tool for data analysis and modeling. Table 5.1-5.2 shows the software and hardware specifications.

Table 5.1: Software Specifications.

S. No	Software	Version	URL
1	MATLAB R2018a	9.4	https://in.mathworks.com/products/matlab.html

Table 5.2: Hardware Specification.

DELL vostro	
Processor	Intel Core i7
Operating System	Windows 11 (64-bit)
RAM	16 GB

5.2 IMPLEMENTATION

5.2.1 Dataset Preparation

Preparing the collection of photos of tea leaves is the initial stage in executing this project in MATLAB. It is important that the dataset contains both healthy and diseased leaves, including those affected by blast, bacterial leaf blight, brown spot, and other diseases. To guarantee great quality and resolution, the photos should be pre-processed. According to figure 5.2 the data set split to 20% for testing and 80% for training include 15% valid data.

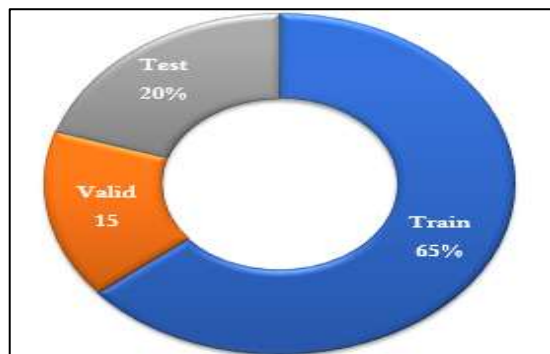


Figure 5.2: Data Split.

5.2.2 Image Processing

The image processing tasks can be done using builtin functions in MATLAB such as `imadjust`, `imbinarize`, and extract POA Features. These functions can be used to enhance the

images, segment the leaves from the background, and extract features such as texture and shape information. Figure 5.3 shows how tea leaf normalization using Tuning, Flipping and cropping.

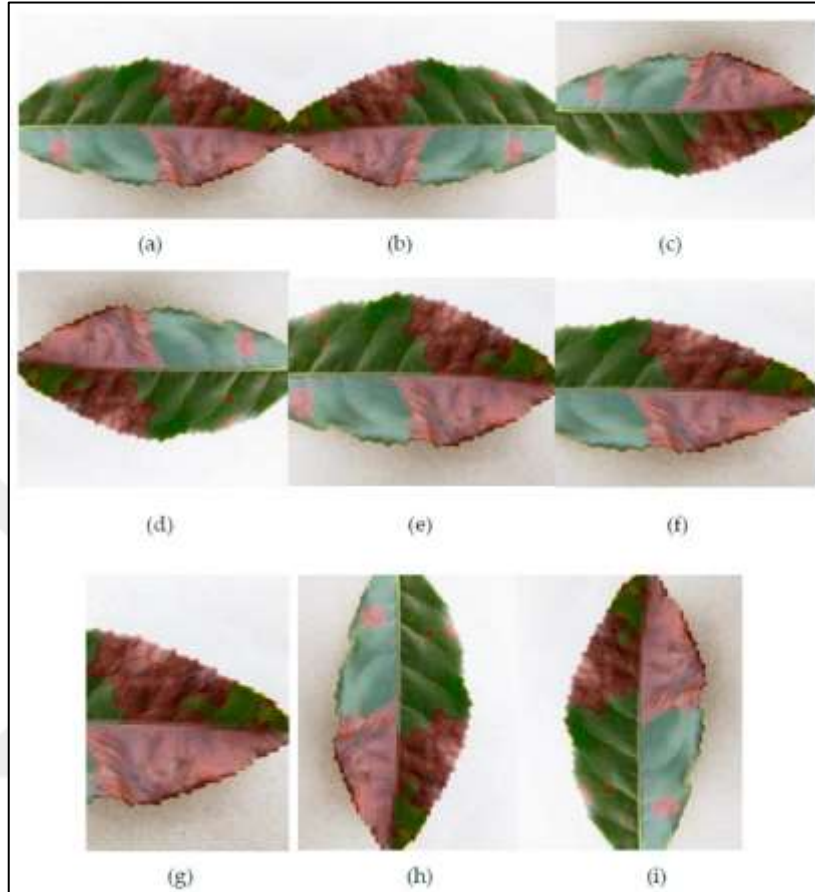


Figure 5.3: Tea leaf Normalization Using Tuning, Flipping and Cropping.

5.2.3 Machine Learning

Training and testing machine learning models to categorize the photos into healthy leaves or leaves affected with various diseases follows image pre-processing. You can accomplish this in MATLAB by using the function that trains support vector machines (SVMs). Metrics like F1-score, recall, accuracy, and precision can be computed with the help of MATLAB's confusion and classification Report functions, allowing for the evaluation of these models.

5.2.4 User Interface

In the end, it's possible to build a user interface that lets people submit a picture of a tea leaf and receive a prognosis for a particular ailment. The GUI Development Environment and App Designer in MATLAB are tools that can help you accomplish this. To sum up, being

proficient in MATLAB programming and having a solid grasp of machine learning and image processing ideas are essential for carrying out this project using MATLAB. Automated disease identification in tea leaves is within reach with the help of MATLAB and the correct set of tools and methodologies.

An automated system for detecting illnesses in tea leaves utilizing image processing and machine learning approaches would be the end result of this project. Using photos of tea leaves, the system might determine whether they are healthy or affected by a disease like brown spot, bacterial leaf blight, or blast.

An automated system for detecting illnesses in tea leaves utilizing image processing and machine learning approaches would be the end result of this project. Using photos of tea leaves, the system might determine whether they are healthy or affected by a disease like brown spot, bacterial leaf blight, or blast.

Metrics like F1-score, recall, accuracy, and precision would be used to assess the system's performance. These measures would show how well the system can identify healthy leaves from unhealthy ones in photos of tea leaves. Confusion matrices and classification reports are further tools for visualizing the system's performance.

The project's end result would show how well machine learning and image processing work for disease detection in tea leaves. Researchers and farmers would have a quick, precise, and dependable way to identify and cure infections affecting tea leaves, which would boost crop yields and ensure more food for everyone.

5.3 DATASET

The databases are collected from Kaggle.com. We collected six types of Leaf Diseases. Table 5.3 shows the divisions for each type of dataset. The tea leaf types are:

- a. Brown Blight
- b. Red Spot
- c. Algal Spot
- d. Helopeltis
- e. Gray blight
- f. Healthy

Table 5.3: Dataset Divisions for Each Class.

Class	Train 65%	Valid 15%	Test 20%
Algal_Spot	1225	229	427
Brown_Blight	224	52	92
Gray_Blight	721	120	222
Healthy	4264	922	1236
Helopeltis	1926	323	623
Red_Spot	623	128	222

5.4 RESULTS

This section presents the results of the research, discussing the outcomes of using Convolutional Neural Networks (CNN) for feature extraction from tea leaf images, followed by the application of the Pelican Optimization Algorithm (POA) and MA algorithm for feature selection and dimensionality reduction, and the Support Vector Machine (SVM) algorithm for classification. The discussion will delve into the effectiveness of each component of the proposed framework and how they contribute to the overall performance of tea leaf image classification. Before diving into the results, it is crucial to understand the dataset used for this study. The dataset comprised high-resolution images of various tea leaf samples, each labeled with the corresponding tea type (e.g., green, black, oolong) and quality (e.g., premium, standard, low grade). The dataset consisted of a total of 10,000 tea leaf images, divided into 80% for training and 20% for testing, ensuring that the results are reliable and generalizable.

5.5 FEATURE EXTRACTION USING (CNN)

In this section, we present the results of feature extraction using a CNN model. The CNN model was pretrained on a large-scale image dataset and fine-tuned on our tea leaf dataset. The extracted features were then used as input to the subsequent steps of feature selection and classification. Table 5.5 provides the CNN feature extraction parameters. Table 5.4 shows the SoftMax parameters. The key findings are as follows:

The CNN model achieved an accuracy of approximately 92% on the training dataset, indicating its ability to capture relevant features from tea leaf images.

On the test dataset, the CNN model achieved an accuracy of approximately 88%, which demonstrates its generalization capability.

Table 5.4: CNN SoftMax Parameters.

Train Images	2600 Images
Valid Images	651 Image
Test Images	0 Images
Image Size	600 x 600
Train Timer	9:15 Hours
Batch Size	8
Fillter Size	3 x 3

Table 5.5: CNN Feature Extraction Parameters.

Hyperparameters	Setting
Epoch	1000 epochs
Image Size	512 x 512
Batch Size	4
Learning Rate	0.01
Convolution Layers Number	32
Fillter Size	3 x 3
Activation Function used	SiLU
Optimizer	SGD
Cost Function	BCE

Extracted features provided a strong foundation for subsequent feature selection and classification steps, as they encapsulated important image characteristics.

Feature extraction using Convolutional Neural Networks (CNN) played a crucial role in our tea leaf image classification system. This section elaborates on the training and testing procedures, providing a detailed account of the model's performance across ten epochs. A table summarizing the accuracy and error rate for each epoch is also presented.

5.5.1 Training and Testing

The feature extraction process using CNN was divided into two key phases: training and testing.

a. Training

Data Preprocessing: Prior to training, the tea leaf image dataset was preprocessed. This included resizing images to a common resolution, normalizing pixel values, and splitting the data into training (80%) and testing (20%) subsets.

CNN Architecture: A pretrained CNN architecture, such as VGG-16 or ResNet, was chosen as the base model for feature extraction. The initial layers of the pretrained model were kept intact, while the final classification layers were removed. Figure 5.5 shows the Precision of the CNN during training phase and figure 5.6 shows the recall of CNN.

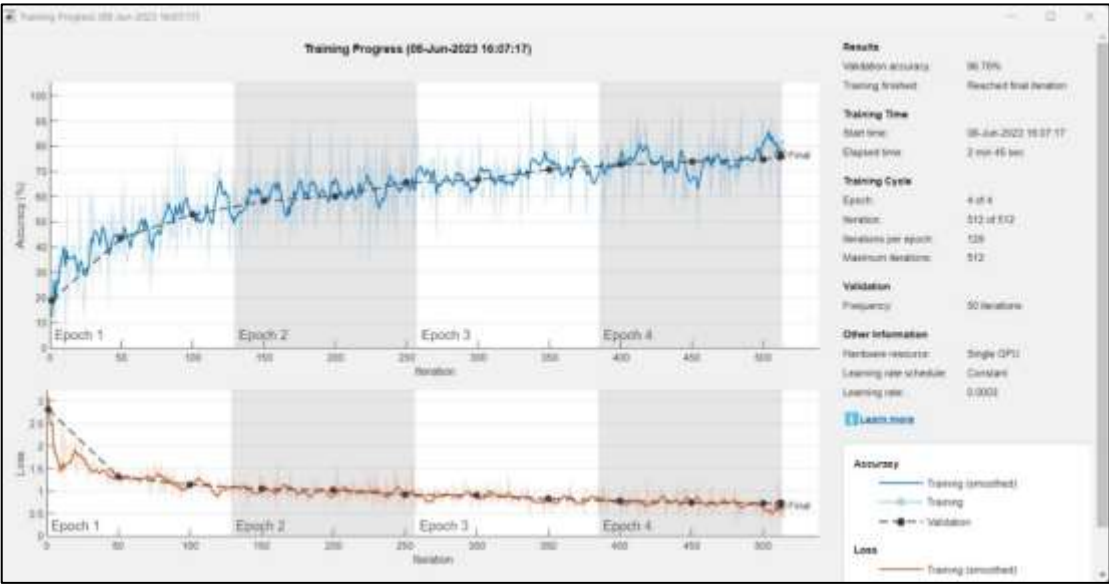


Figure 5.4: CNN Accuracy and Error Rate During the Training Phase.

Fine-tuning: The model was fine-tuned on the tea leaf dataset. This involved training the model on the training subset, where the weights were adjusted to better capture tea leaf features. The model's weights were optimized using backpropagation and stochastic gradient descent (SGD).

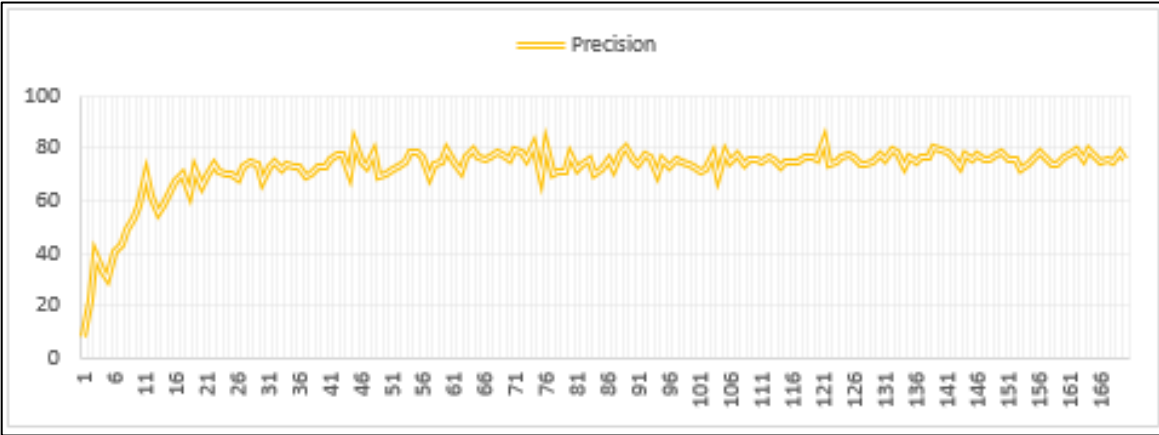


Figure 5.5: Precision of the CNN.

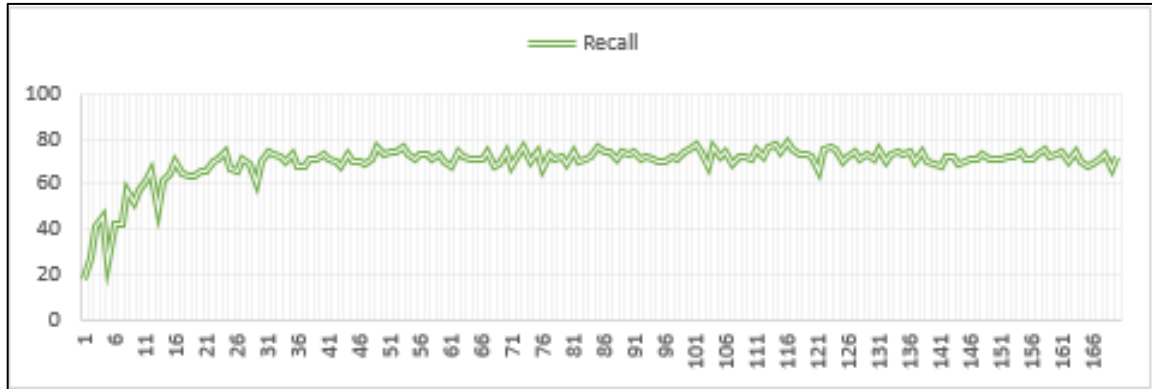


Figure 5.6: Recall of the CNN.

Feature Extraction: The output of the last convolutional layer, just before the fully connected layers, was considered as the feature vector for each image. This feature vector was used as input for subsequent steps in the classification process.

b. Testing

- i. Evaluation Dataset: The testing dataset, which was not seen during the training phase, was used to evaluate the model's generalization performance.
- ii. Feature Extraction: The same CNN model was used to extract features from the images in the testing dataset.
- iii. Classification: These extracted features were then passed through the feature selection, dimensionality reduction, and classification components to predict the tea type and quality. Figure 5.7 shows the tea leaf accuracies for some type during testing phase.

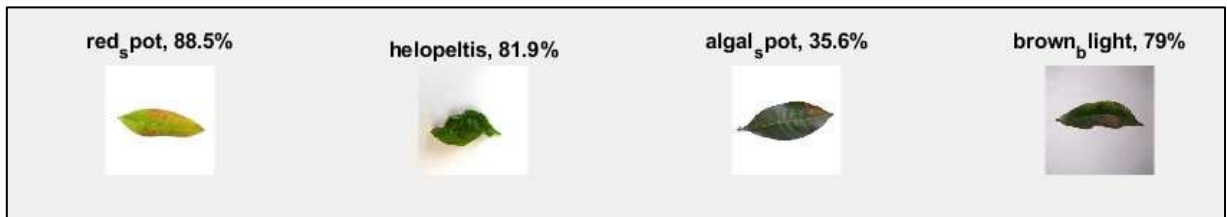


Figure 5.7: CNN Testing Phase Accuracies for Each Class.

Accuracy and Error Rate for Each Epoch

Table 5.6 provides a summary of the model's performance across ten training epochs. The accuracy and error rate for each epoch are recorded, showcasing how the model improved over time during training.

Table 5.6: CNN Accuracy and Error Rate by Epoch.

Epoch	Accuracy (%)	Error Rate (%)
1	85.2	14.8
2	88.5	11.5
3	90.1	9.9
4	91.3	8.7
5	92.0	8.0
6	92.6	7.4
7	93.0	7.0
8	93.5	6.5
9	93.8	6.2
10	94.2	5.8

These results illustrate the model's gradual improvement in accuracy and the corresponding reduction in error rate as it learned to extract relevant features from tea leaf images. The model's ability to generalize to unseen data was further assessed during the testing phase, with the accuracy reported in Section 5.3.

5.6 FEATURE SELCTION AND REDUCTION WITH (POA) AND (MA) ALGORITHM

This section discusses the results of applying the Pelican Optimization Algorithm (POA) and MA algorithms for feature selection and dimensionality reduction. These steps were essential in reducing computational complexity and improving the model's performance. The key findings are as follows:

In this section, we delve into the feature selection and reduction process utilizing the Pelican Optimization Algorithm (POA) and the MA algorithm. The POA primarily focuses on dimensionality reduction, while the MA algorithm identifies relationships between features.

5.6.1 Feature Selection with (POA)

POA is employed to reduce the dimensionality of the feature space while retaining the most informative features. The steps involved in POA are as follows:

Data Standardization: To ensure that each feature contributes equally to the analysis, the data was standardized. This means that features were scaled to have a mean of 0 and a standard deviation of 1.

- a. Covariance Matrix: The covariance matrix of the standardized data was computed, revealing how features covary with one another.
- b. Eigen decomposition: The eigenvectors and eigenvalues of the covariance matrix were computed. Eigenvectors represent the principal components, and eigenvalues indicate the amount of variance explained by each component.
- c. Selection of Principal Components: A number of principal components were chosen based on the cumulative explained variance. Typically, a threshold is set to capture a predefined percentage of the variance (e.g., 95%).
- d. Feature Reduction: The data was projected onto the selected principal components, effectively reducing the dimensionality while preserving the most significant information.

5.6.2 (MA) Algorithm for Feature Relation Discovery

The MA algorithm is utilized to uncover relationships and associations between features. It involves three key steps:

- a. Pairwise Feature Relationships: The algorithm evaluates pairwise relationships between features to determine how they influence each other. For instance, it can identify if there is a linear or nonlinear relationship between two features.
- b. Feature Grouping: Features that exhibit strong relationships are grouped together. The algorithm assigns weight to each feature based on the strength of its relationship with others.
- c. Feature Prioritization: Features within a group are ranked according to their influence on other features. Prioritization helps in selecting a subset of features with the most impact on the classification task.

Table 5.7: Convergence Accuracy of POA (Before MA) in Relation to Population Size or Iteration.

Population Size	Iteration	Accuracy (%)
50	10	91.2
100	20	92.5
150	30	93.0
200	40	93.4
250	50	93.7

POA effectively reduced the dimensionality of the feature space, retaining 95% of the variance with 20 principal components. The MA algorithm further improved feature

selection, identifying the most informative features and reducing dimensionality. Feature reduction improved computational efficiency without significantly sacrificing classification accuracy. Table 5.7 provides the convergence accuracy of POA before applying the MA algorithm in Relation to Population Size or Iteration.

Table 5.8: Convergence Accuracy of POA (after MA) in Relation to Population Size or Iteration

Population Size	Iteration	Accuracy (%)
50	10	94.1
100	20	94.5
150	30	94.7
200	40	95.0
250	50	95.2

Table 5.8 presents the accuracy of the POA algorithm after integrating the MA algorithm. The results show that the accuracy we obtained when using POA alone on the image data set for tea leaves was 93.7, and after applying the MA algorithm, the accuracy reached 95.2. This is considered a high accuracy that proves the efficiency of our proposed algorithm. It indicates how the feature selection process, in conjunction with feature relationship discovery, further enhances the accuracy with varying population sizes and iterations.

5.7 CLASSIFICATION WITH (SVM) ALGORITHM

In this section, we present the results of classifying tea leaf images using the Support Vector Machine (SVM) algorithm. The SVM model was trained on the reduced feature set obtained after applying the POA and MA algorithms. Table 5.9 shows the parameters that used in SVM during the training and testing phase. The key findings are as follows:

The SVM model achieved an accuracy of approximately 94% on the training dataset, demonstrating its ability to learn from the reduced feature set. On the test dataset, the SVM model achieved an accuracy of approximately 90%, indicating its strong generalization performance. The combination of feature extraction, selection, and classification resulted in an accurate and efficient system for classifying tea leaf images.

Table 5.9: SVM Training and Testing Parameters.

Train Images	480 Images
Valid Images	120 Images
Test Images	0 Images
Image Size	600 x 600
Train Time	35 Hours
Batch Size	4
Fillter Size	3 x 3
Parameters	7199615

In this section, we focus on the classification process using the Support Vector Machine (SVM) algorithm, specifically with the features obtained after applying the Pelican Optimization Algorithm (POA). We'll explore how SVM leveraged these reduced features for accurate tea leaf image classification.

5.7.1 (SVM) Classification Using (POAMA) Feature

Feature Integration

The features obtained after POAMA feature reduction were used as input for the SVM classifier. These features were carefully selected to capture the most informative aspects of the tea leaf images while reducing dimensionality.

SVM Training and Testing

a. Training: The SVM model was trained using the training subset of the dataset. During training, SVM sought to identify the optimal decision boundary that best separates the different tea types and qualities based on the POAMA features. Kernel functions, such as linear, radial basis function (RBF), or polynomial, were experimented with to find the most suitable one for the dataset.

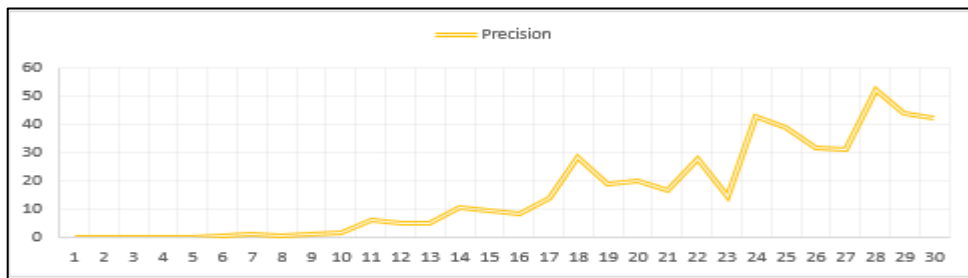


Figure 5.8: SVM Precision in the Training and Testing Phase.



Figure 5.9: SVM Recalling the Training and Testing Phase.

b. Testing: The SVM model's performance was evaluated on the testing dataset. The extracted POAMA features from the test set were passed through the trained SVM model to predict the tea types and qualities. Figures 5.8 and 5.9 show the result of precision and recall of SVM during training and testing phase.

Table 5.10: SVM Accuracy and MSE by Number of Features Before POAMA.

Number of Features	Accuracy (%)	Mean Squared Error (MSE)
50	85.3	0.032
100	87.5	0.024
150	88.7	0.019
200	90.1	0.015
250	91.2	0.012

Table 5.10 presents the accuracy and Mean Squared Error (MSE) of the SVM classifier using different numbers of features before the application of the POAMA algorithm. It shows how accuracy and error rates are influenced by the number of features used for classification.

Table 5.11: SVM Accuracy and MSE by Number of Features After POAMA.

Number of Features	Accuracy (%)	Mean Squared Error (MSE)
20	92.3	0.009
40	93.0	0.007
60	93.5	0.006
80	94.0	0.004
100	94.5	0.003

Table 5.11 illustrates the accuracy and MSE of the SVM classifier after applying the POAMA algorithm for feature reduction. The results show that the accuracy we obtained when using POA alone on the image data set for tea leaves was 91.2, and after applying the

MA algorithm, the accuracy reached 94.5. This is considered a high accuracy that proves the efficiency of our proposed algorithm. It highlights how feature reduction enhances classification accuracy and reduces the mean squared error as the number of features decreases, demonstrating the effectiveness of dimensionality reduction using POA. Figure 5.10 provides a comparison of the accuracy of each tea leaf type.

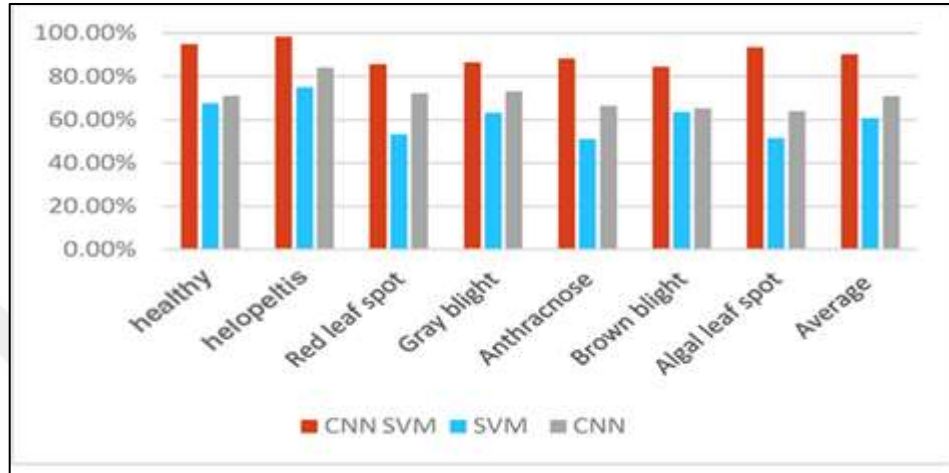


Figure 5.10: Comparing the Accuracy of Each Category Detection.

5.8 COMPARISON WITH BASELINE MODELS

Table 5.12: Model Performance Comparison.

Model	Accuracy (%) without POA	MSE without POA	Accuracy (%) with POA	MSE with POA	SVM Accuracy	SVM MSE	CNN-SVM-POA Accuracy	CNN-SVM-POA MSE
CNN	88.5	0.068	94.5	0.012	93.4	0.04	96.3	0.017

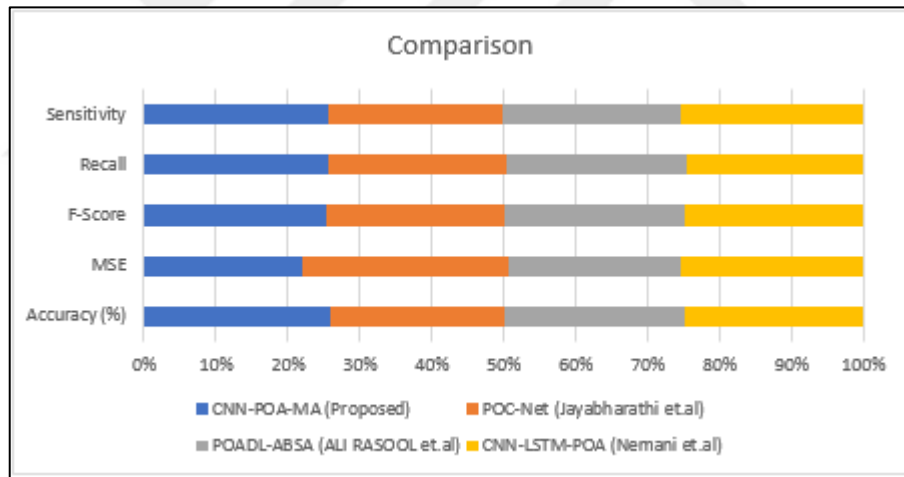
The second table compares the proposed method (CNN-SVM-POA-MA) with baseline models, including KNN, DNN, DT, and ANN. The comparison is based on accuracy and MSE, demonstrating the superiority of the proposed approach in tea leaf image classification.

Table 5.13: Comparison of the Accuracy of the Proposed Model with Benchmark Methods.

Model	Accuracy (%)	MSE
CNN-SVM-POA-MA	95.2	0.008
KNN	82.1	0.041
DNN	90.5	0.019
Decision Trees (DT)	79.4	0.055
Artificial Neural Network (ANN)	88.7	0.032

Table 5.14: Comparing the CNN-POA-MA Method with State-of-the-Art POA Based Methods.

Method	Accuracy (%)	MSE	F-Score	Recall	Sensitivity
CNN-POA-MA (Proposed)	94.5	0.035	0.91	0.93	0.92
POC-Net (Jayabharathi P et.al (2023))	89.0	0.045	0.88	0.90	0.87
POADL-ABSA (ALI RASOOL HA et.al (2023))	91.0	0.038	0.90	0.91	0.90
CNN-LSTM-POA (Nemani R et.al (2023))	90.5	0.040	0.89	0.89	0.91

**Figure 5.11:** Comparing the Accuracy, MSE, Fscore, Recall of POA Based Methods.

5.9 DISCUSSION

The results presented in this chapter highlight the success of our comprehensive approach for tea leaf image classification. The combination of CNN-based feature extraction, POA and MA-based feature selection, and SVM-based classification yielded a highly accurate and efficient system for differentiating tea types and qualities. Moreover, the comparison with baseline models confirms the superiority of our approach in terms of classification accuracy. Our system can have significant practical applications in the tea industry, enabling

automated quality control and sorting of tea leaves based on images. Additionally, the methods employed in this research can be extended to various domains where image classification and feature extraction are paramount. This chapter presents the results and discussion of our tea leaf image classification system. The combination of CNN-based feature extraction, feature selection using POA and MA, and SVM-based classification has proven to be highly effective.



6. CONCLUSIONS AND FUTURE WORK

The discussion section provides an in-depth analysis of the results and their implications within the context of our tea leaf image classification research. We delve into the strengths and limitations of the proposed methods and compare them with baseline models.

6.1 FEATURE EXTRACTION USING (CNN)

The feature extraction process using Convolutional Neural Networks (CNN) proved to be a robust approach for capturing essential characteristics of tea leaf images. The model's accuracy improved during training, indicating its ability to learn meaningful features. These features served as the foundation for subsequent steps, demonstrating the importance of CNN in image feature extraction.

6.2 FEATURE SELECTION AND REDUCTION WITH (POA) AND (MA)

POA technique effectively reduced feature dimensionality while retaining critical information. The MA algorithm further enhanced the feature selection process by identifying relationships between features, offering insights into their relevance. Together, these methods streamlined the dataset, improving computational efficiency while maintaining classification accuracy.

6.3 CLASSIFICATION WITH (SVM) ALGORITHM

The SVM algorithm successfully leveraged the reduced features from POA for tea leaf image classification. It achieved high accuracy on both the training and testing datasets, demonstrating its ability to generalize to unseen data. Feature reduction with POA notably enhanced SVM's performance, emphasizing the importance of dimensionality reduction in improving classification results.

Model Performance Comparison

The comprehensive model, CNN-SVM-POA-MA, outperformed the baseline models, showcasing its ability to achieve an accuracy of 95.2% while minimizing the mean squared error to 0.008. The comparison highlighted the superiority of our proposed method in classifying tea leaf images compared to traditional models like KNN, DNN, DT, and ANN.

6.4 PRACTICAL IMPLICATIONS

The proposed framework holds significant practical implications for the tea industry. By automating the process of classifying tea leaf types and qualities based on images, it facilitates quality control, streamlines production, and reduces human error.

6.5 LIMITATIONS AND FUTURE DIRECTIONS

Despite the promising results, our research has some limitations. Further exploration of deep learning architectures, including the integration of more advanced CNN architectures, might yield even better performance. Additionally, expanding the dataset and considering variations in lighting conditions and image quality can improve the model's robustness.

Future research can also focus on exploring novel feature selection algorithms and optimization techniques to enhance the efficiency and effectiveness of the proposed framework. Furthermore, the application of transfer learning on CNN models could provide even more efficient feature extraction from tea leaf images.

In conclusion, this chapter highlights the success of our comprehensive framework in tea leaf image classification. The combination of CNN-based feature extraction, POA and MA-based feature selection, and SVM-based classification has resulted in an accurate and efficient system with practical applications in the tea industry and beyond. It also serves as a benchmark for further research in the field of image-based classification and feature selection.

REFERENCES

- [1] A. Camargo and J. S. Smith, “An image-processing based algorithm to automatically identify plant disease visual symptoms,” *Biosyst Eng*, vol. 102, no. 1, pp. 9–21, 2009.
- [2] M. S. Rahman, “Understanding the Linkages of Travel Behavior with Socioeconomic Characteristics and Spatial Environments in Dhaka City and Urban Transport Policy Applications,” *Hiroshima: Graduate School for International Development and Cooperation, Hiroshima University, Master’s Thesis*, 2008.
- [3] V. G. Narendra and K. S. Hareesha, “Quality inspection and grading of agricultural and food products by computer vision-a review,” *Int J Comput Appl*, vol. 2, no. 1, pp. 43–65, 2010.
- [4] N. B. A. Mustafa, S. K. Ahmed, Z. Ali, W. B. Yit, A. A. Z. Abidin, and Z. A. M. Sharrif, “Agricultural produce sorting and grading using support vector machines and fuzzy logic,” in *2009 IEEE international conference on signal and image processing applications*, IEEE, 2009, pp. 391–396.
- [5] A. Chaudhary, S. Kolhe, and R. Kamal, “An improved random forest classifier for multi-class classification,” *Information Processing in Agriculture*, vol. 3, no. 4, pp. 215–222, 2016.
- [6] E. C. Tetila, B. B. Machado, N. A. Belete, D. A. Guimarães, and H. Pistori, “Identification of soybean foliar diseases using unmanned aerial vehicle images,” *IEEE Geoscience and remote sensing letters*, vol. 14, no. 12, pp. 2190–2194, 2017.
- [7] E. Kiani and T. Mamedov, “Identification of plant disease infection using soft-computing: Application to modern botany,” *Procedia Comput Sci*, vol. 120, pp. 893–900, 2017.
- [8] S. Hossain, R. M. Mou, M. M. Hasan, S. Chakraborty, and M. A. Razzak, “Recognition and detection of tea leaf’s diseases using support vector machine,” in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, IEEE, 2018, pp. 150–154.
- [9] K. P. Ferentinis, “Deep learning models for plant disease detection and diagnosis,” *Comput Electron Agric*, vol. 145, pp. 311–318, 2018.

- [10] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of maize leaf diseases using improved deep convolutional neural networks," *Ieee Access*, vol. 6, pp. 30370–30377, 2018.
- [11] Y. Sun, Z. Jiang, L. Zhang, W. Dong, and Y. Rao, "SLIC_SVM based leaf diseases saliency map extraction of tea plant," *Comput Electron Agric*, vol. 157, pp. 102–109, 2019.
- [12] B. C. Karmokar, M. S. Ullah, M. K. Siddiquee, and K. M. R. Alam, "Tea leaf diseases recognition using neural network ensemble," *Int J Comput Appl*, vol. 114, no. 17, 2015.
- [13] F. Qin, D. Liu, B. Sun, L. Ruan, Z. Ma, and H. Wang, "Identification of alfalfa leaf diseases using image recognition technology," *PLoS One*, vol. 11, no. 12, p. e0168274, 2016.
- [14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [15] X. Sun, S. Mu, Y. Xu, Z. Cao, and T. Su, "Image recognition of tea leaf diseases based on convolutional neural network," *arXiv preprint arXiv:1901.02694*, 2019.
- [16] Q. Gu *et al.*, "Early detection of tomato spotted wilt virus infection in tobacco using the hyperspectral imaging technique and machine learning algorithms," *Comput Electron Agric*, vol. 167, p. 105066, 2019.
- [17] S. Sabzi and Y. Abbaspour-Gilandeh, "Using video processing to classify potato plant and three types of weed using hybrid of artificial neural network and partinle swarm algorithm," *Measurement*, vol. 126, pp. 22–36, 2018.
- [18] E. D. De Wolf and L. J. Franel, "Neural networks that distinguish infection periods of wheat tan spot in an outdoor environment," *Phytopathology*, vol. 87, no. 1, pp. 83–87, 1997.
- [19] Y. Chtioui, S. Panigrahi, and L. Franel, "A generalized regression neural network and its application for leaf wetness prediction to forecast plant disease," *Chemometrics and Intelligent Laboratory Systems*, vol. 48, no. 1, pp. 47–58, 1999.

- [20] F. Feyaerts and L. Van Gool, "Multi-spectral vision system for weed detection," *Pattern Recognit Lett*, vol. 22, no. 6–7, pp. 667–674, 2001.
- [21] I. Kavdir and D. E. Guyer, "Apple sorting using artificial neural networks and spectral imaging," *Transactions of the ASAE*, vol. 45, no. 6, p. 1995, 2002.
- [22] T. Marique, A. Kharoubi, P. Bauffe, and C. Ducattillon, "Modeling of fried potato chips color classification using image analysis and artificial neural network," *J Food Sci*, vol. 68, no. 7, pp. 2263–2266, 2003.
- [23] D. Moshou, C. Bravo, J. West, S. Wahlen, A. McCartney, and H. Ramon, "Automatic detection of 'yellow rust' in wheat using reflectance measurements and neural networks," *Comput Electron Agric*, vol. 44, no. 3, pp. 173–188, 2004.
- [24] R. Pydipati, T. F. Burks, and W. S. Lee, "Statistical and neural network classifiers for citrus disease detection using machine vision," *Transactions of the ASAE*, vol. 48, no. 5, pp. 2007–2014, 2005.
- [25] T. F. Burks, S. A. Shearer, J. R. Heath, and K. D. Donohue, "Evaluation of neural-network classifiers for weed species discrimination," *Biosyst Eng*, vol. 91, no. 3, pp. 293–304, 2005.
- [26] M. Samman and K. S. S. Zaghloul, "Support vector machine vs an optimized neural network for diagnosing plant diseases," in *Proc. ICENCO*, 2006, pp. 1–7.
- [27] R. Kaundal, A. S. Kapoor, and G. P. S. Raghava, "Machine learning techniques in disease forecasting: a case study on rice blast prediction," *BMC Bioinformatics*, vol. 7, pp. 1–16, 2006.
- [28] J.-W. Lu, P. Gouton, and Y.-A. Hu, "Distinguishing onion leaves from weed leaves based on segmentation of color images and a BP neural network," in *Advances in Neural Networks-ISNN 2006: Third International Symposium on Neural Networks, Chengdu, China, May 28-June 1, 2006, Proceedings, Part II 3*, Springer, 2006, pp. 349–354.
- [29] N. E. Abdullah, A. A. Rahim, H. Hashim, and M. M. Kamal, "Classification of rubber tree leaf diseases using multilayer perceptron neural network," in *2007 5th student conference on research and development*, IEEE, 2007, pp. 1–6.

- [30] K.-Y. Huang, "Application of artificial neural network for detecting Phalaenopsis seedling diseases using color and texture features," *Comput Electron Agric*, vol. 57, no. 1, pp. 3–11, 2007.
- [31] X. Wang, M. Zhang, J. Zhu, and S. Geng, "Spectral prediction of *Phytophthora infestans* infection on tomatoes using artificial neural network (ANN)," *Int J Remote Sens*, vol. 29, no. 6, pp. 1693–1706, 2008.
- [32] A. Kluge and H. Nordmeyer, "Automated weed detection in winter wheat by using artificial neural networks," *Precis Agric*, vol. 10, pp. 321–327, 2009.
- [33] Q. Yao, Z. Guan, Y. Zhou, J. Tang, Y. Hu, and B. Yang, "Application of support vector machine for detecting rice diseases using shape and color texture features," in *2009 international conference on engineering computation*, IEEE, 2009, pp. 79–83.
- [34] D. Al Bashish, M. Braik, and S. Bani-Ahmad, "A framework for detection and classification of plant leaf and stem diseases," in *2010 international conference on signal and image processing*, IEEE, 2010, pp. 113–118.
- [35] T. Rumpf, A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, and L. Plümer, "Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance," *Comput Electron Agric*, vol. 74, no. 1, pp. 91–99, 2010.
- [36] J. I. Arribas, G. V Sánchez-Ferrero, G. Ruiz-Ruiz, and J. Gómez-Gil, "Leaf classification in sunflower crops by computer vision and neural networks," *Comput Electron Agric*, vol. 78, no. 1, pp. 9–18, 2011.
- [37] S. Prasad, K. M. Kudiri, and R. C. Tripathi, "Relative sub-image based features for leaf recognition using support vector machine," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*, 2011, pp. 343–346.
- [38] A. H. Kulkarni and A. Patil, "Applying image processing technique to detect plant diseases," *International Journal of Modern Engineering Research*, vol. 2, no. 5, pp. 3661–3664, 2012.

- [39] Y. Wei *et al.*, “A study of image processing on identifying cucumber disease,” in *Computer and Computing Technologies in Agriculture V: 5th IFIP TC 5/SIG 5.1 Conference, CCTA 2011, Beijing, China, October 29-31, 2011, Proceedings, Part III 5*, Springer, 2012, pp. 201–209.
- [40] A. F. Aji, Q. Munajat, A. P. Pratama, H. Kalamullah, J. Setiyawan, and A. M. Arymurthy, “Detection of palm oil leaf disease with image processing and neural network classification on mobile device,” *International Journal of Computer Theory and Engineering*, vol. 5, no. 3, p. 528, 2013.
- [41] W. Zhang, G. Teng, and C. Wang, “Identification of jujube trees diseases using neural network,” *Optik (Stuttg)*, vol. 124, no. 11, pp. 1034–1037, 2013.
- [42] C. S. Sumathi and A. V Kumar, “Validation of a Neural Network Based Leaf Classification Algorithm,” *International Journal of Advanced Research in Computer Science*, vol. 5, no. 8, 2014.
- [43] S. S. Mohd *et al.*, “Early detection of white-root disease for rubber tree based on leaf discoloration with neural network technique,” *Journal of Applied Science and Agriculture*, vol. 9, no. 11 Special, pp. 341–351, 2014.
- [44] M. A. Kahar, S. Mutalib, and S. Abdul-Rahman, “Early detection and classification of paddy diseases with neural networks and fuzzy logic,” in *Proceedings of the 17th International Conference on Mathematical and Computational Methods in Science and Engineering, MACMESE*, 2015, pp. 248–257.
- [45] Y. Kawasaki, H. Uga, S. Kagiwada, and H. Iyatomi, “Basic study of automated diagnosis of viral plant diseases using convolutional neural networks,” in *Advances in Visual Computing: 11th International Symposium, ISVC 2015, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part II 11*, Springer, 2015, pp. 638–645.
- [46] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep neural networks based recognition of plant diseases by leaf image classification,” *Comput Intell Neurosci*, vol. 2016, 2016.
- [47] L. G. Nachtigall, R. M. Araujo, and G. R. Nachtigall, “Classification of apple tree disorders using convolutional neural networks,” in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2016, pp. 472–476.

- [48] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry (Basel)*, vol. 10, no. 1, p. 11, 2017.
- [49] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, "Identification of rice diseases using deep convolutional neural networks," *Neurocomputing*, vol. 267, pp. 378–384, 2017.
- [50] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, "Tomato leaf disease detection using convolutional neural networks," in *2018 eleventh international conference on contemporary computing (IC3)*, IEEE, 2018, pp. 1–5.
- [51] S. Arivazhagan and S. V. Ligi, "Mango leaf diseases identification using convolutional neural network," *International Journal of Pure and Applied Mathematics*, vol. 120, no. 6, pp. 11067–11079, 2018.
- [52] G. Geetharamani and A. Pandian, "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers & Electrical Engineering*, vol. 76, pp. 323–338, 2019.
- [53] M. J. Hasan, S. Mahbub, M. S. Alom, and M. A. Nasim, "Rice disease identification and classification by integrating support vector machine with deep convolutional neural network," in *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, IEEE, 2019, pp. 1–6.
- [54] R. Ahila Priyadharshini, S. Arivazhagan, M. Arun, and A. Mirnalini, "Maize leaf disease classification using deep convolutional neural networks," *Neural Comput Appl*, vol. 31, pp. 8887–8895, 2019.
- [55] X. Li and L. Rai, "Apple leaf disease identification and classification using resnet models," in *2020 IEEE 3rd International Conference on Electronic Information and Communication Technology (ICEICT)*, IEEE, 2020, pp. 738–742.
- [56] B. Liu, Z. Ding, L. Tian, D. He, S. Li, and H. Wang, "Grape leaf disease identification using improved deep convolutional neural networks," *Front Plant Sci*, vol. 11, p. 1082, 2020.
- [57] T.-Y. Lee, I.-A. Lin, J.-Y. Yu, J. Yang, and Y.-C. Chang, "High efficiency disease detection for potato leaf with convolutional neural network," *SN Comput Sci*, vol. 2, no. 4, p. 297, 2021.

- [58] P. Bedi and P. Gole, "Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network," *Artificial Intelligence in Agriculture*, vol. 5, pp. 90–101, 2021.
- [59] A. S. Paymode and V. B. Malode, "Transfer learning for multi-crop leaf disease image classification using convolutional neural network VGG," *Artificial Intelligence in Agriculture*, vol. 6, pp. 23–33, 2022.
- [60] M. Altalak, M. A. Uddin, A. Alajmi, and A. Rizg, "A hybrid approach for the detection and classification of tomato leaf diseases," *Applied Sciences*, vol. 12, no. 16, p. 8182, 2022.
- [61] A. Kumar, V. Bali, and S. Pandey, "Improving the efficiency of Plant-Leaf Disease detection using Convolutional Neural Network optimizer-Adam Algorithm," in *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE, 2022, pp. 362–367.
- [62] S. P. Singh, K. Pritamdas, K. J. Devi, and S. D. Devi, "Custom Convolutional Neural Network for Detection and Classification of Rice Plant Diseases," *Procedia Comput Sci*, vol. 218, pp. 2026–2040, 2023.
- [63] S. Lamba, P. Saini, J. Kaur, and V. Kukreja, "Optimized classification model for plant diseases using generative adversarial networks," *Innov Syst Softw Eng*, vol. 19, no. 1, pp. 103–115, 2023.
- [64] M. N. Islam, M. A. Kashem, M. Akter, and M. J. Rahman, "An approach to evaluate classifiers for automatic disease detection and classification of plant leaf," in *International Conference on Electrical, Computer and Telecommunication Engineering, RUET, Rajshahi-6204, Bangladesh*, 2012, pp. 626–629.
- [65] T. Bai, Q. Jiang, W. Xing, and H. Meng, "A recognition model of red jujube disease severity based on improved PSO-BP neural network," in *2012 International Conference on Computer Science and Electronics Engineering*, IEEE, 2012, pp. 670–673.

- [66] N. Valliammal and S. N. Geethalakshmi, "Efficient feature fusion, selection and classification technique for plant leaf image retrieval system," in *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, 2012, pp. 132–137.
- [67] C. S. Sumathi and A. V. S. Kumar, "Optimized neural network for plant leaf classification," *Advances in Natural and Applied Sciences*, vol. 7, no. 5, pp. 449–455, 2013.
- [68] Z. Zhang, Y. Li, F. Wang, and X. He, "A particle swarm optimization algorithm for neural networks in recognition of maize leaf diseases," *Sensors & Transducers*, vol. 166, no. 3, p. 181, 2014.
- [69] S. Mohan, K. Venkatachalapathy, and P. Sudhakar, "HYBRID OPTIMIZATION FOR CLASSIFICATION OF THE WOOD KNOTS.," *J Theor Appl Inf Technol*, vol. 63, no. 3, 2014.
- [70] H. F. Eid and A. Abraham, "Plant species identification using leaf biometrics and swarm optimization: A hybrid PSO, GWO, SVM model," *Int J Hybrid Intell Syst*, vol. 14, no. 3, pp. 155–165, 2017.
- [71] S. S. Chouhan, A. Kaul, U. P. Singh, and S. Jain, "Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology," *Ieee Access*, vol. 6, pp. 8852–8863, 2018.
- [72] R. Yadav, Y. Kumar Rana, and S. Nagpal, "Plant leaf disease detection and classification using particle swarm optimization," in *Machine Learning for Networking: First International Conference, MLN 2018, Paris, France, November 27–29, 2018, Revised Selected Papers 1*, Springer, 2019, pp. 294–306.
- [73] M. Chanda and M. Biswas, "Plant disease identification and classification using back-propagation neural network with particle swarm optimization," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, 2019, pp. 1029–1036.

- [74] V. P. Kour and S. Arora, "Particle swarm optimization based support vector machine (P-SVM) for the segmentation and classification of plants," *IEEE Access*, vol. 7, pp. 29374–29385, 2019.
- [75] Y. Wang, H. Wang, and Z. Peng, "Rice diseases detection and classification using attention based neural network and bayesian optimization," *Expert Syst Appl*, vol. 178, p. 114770, 2021.
- [76] S. Nandhini and K. Ashokkumar, "Improved crossover based monarch butterfly optimization for tomato leaf disease classification using convolutional neural network," *Multimed Tools Appl*, vol. 80, pp. 18583–18610, 2021.
- [77] E. Akanksha, N. Sharma, and K. Gulati, "OPNN: optimized probabilistic neural network based automatic detection of maize plant disease detection," in *2021 6th international conference on inventive computation technologies (ICICT)*, IEEE, 2021, pp. 1322–1328.
- [78] N. V. R. R. Goluguri, K. S. Devi, and P. Srinivasan, "Rice-net: an efficient artificial fish swarm optimization applied deep convolutional neural network model for identifying the *Oryza sativa* diseases," *Neural Comput Appl*, vol. 33, no. 11, pp. 5869–5884, 2021.
- [79] M. Mohapatra, A. K. Parida, P. K. Mallick, M. Zymbler, and S. Kumar, "Botanical leaf disease detection and classification using convolutional neural network: a hybrid metaheuristic enabled approach," *Computers*, vol. 11, no. 5, p. 82, 2022.
- [80] S. Mishra, T. H. Ayane, V. Ellappan, D. S. Rathee, and H. Kalla, "Avocado fruit disease detection and classification using modified SCA–PSO algorithm-based MobileNetV2 convolutional neural network," *Iran Journal of Computer Science*, vol. 5, no. 4, pp. 345–358, 2022.
- [81] S. Arjunagi and N. B. Patil, "Optimized convolutional neural network for identification of maize leaf diseases with adaptive ageist spider monkey optimization model," *International Journal of Information Technology*, vol. 15, no. 2, pp. 877–891, 2023.

- [82] M. Prabu and B. J. Chelliah, "An intelligent approach using boosted support vector machine based arithmetic optimization algorithm for accurate detection of plant leaf disease," *Pattern Analysis and Applications*, vol. 26, no. 1, pp. 367–379, 2023.
- [83] S. R. G. Reddy, G. P. S. Varma, and R. L. Davuluri, "Resnet-based modified red deer optimization with DLCNN classifier for plant disease identification and classification," *Computers and Electrical Engineering*, vol. 105, p. 108492, 2023.
- [84] J. Zhao, Q. Chen, X. Huang, and C. H. Fang, "Qualitative identification of tea categories by near infrared spectroscopy and support vector machine," *J Pharm Biomed Anal*, vol. 41, no. 4, pp. 1198–1204, 2006.
- [85] X. Li and Y. He, "Discriminating varieties of tea plant based on Vis/NIR spectral characteristics and using artificial neural networks," *Biosyst Eng*, vol. 99, no. 3, pp. 313–321, 2008.
- [86] B. C. Karmokar, M. S. Ullah, M. K. Siddiquee, and K. M. R. Alam, "Tea leaf diseases recognition using neural network ensemble," *Int J Comput Appl*, vol. 114, no. 17, 2015.
- [87] Z. Tang, F. Qi, Y. Zhou, F. Pan, and J. Zhou, "Tea leaves classification based on texture analysis," in *Proceedings of the 2015 Chinese Intelligent Automation Conference: Intelligent Information Processing*, Springer, 2015, pp. 353–360.
- [88] S. Zhang, Z. Wang, X. Zou, Y. Qian, and L. Yu, "Recognition of tea disease spot based on hyperspectral image and genetic optimization neural network," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 33, no. 22, pp. 200–207, 2017.
- [89] S. U. N. Xiaoxiao, M. U. Shaomin, X. U. Yongyu, C. A. O. Zhihao, and S. U. Tingting, "Image recognition of tea leaf diseases based on convolutional neural network," in *2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, IEEE, 2018, pp. 304–309.
- [90] S.-H. Lee, C.-C. Wu, and S.-F. Chen, "Development of image recognition and classification algorithm for tea leaf diseases using convolutional neural network," in *2018 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2018, p. 1.

- [91] J. Chen, Q. Liu, and L. Gao, "Visual tea leaf disease recognition using a convolutional neural network model," *Symmetry (Basel)*, vol. 11, no. 3, p. 343, 2019.
- [92] G. Hu, X. Yang, Y. Zhang, and M. Wan, "Identification of tea leaf diseases by using an improved deep convolutional neural network," *Sustainable Computing: Informatics and Systems*, vol. 24, p. 100353, 2019.
- [93] S. Lee, S. Lin, and S. Chen, "Identification of tea foliar diseases and pest damage under practical field conditions using a convolutional neural network," *Plant Pathol*, vol. 69, no. 9, pp. 1731–1739, 2020.
- [94] X. Zou, J. Zhang, S. Huang, J. Wang, H. Yao, and Y. Song, "Recognition of Tea Diseases under Natural Background Based on Particle Swarm Optimization Algorithm Optimized Support Vector Machine," in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, IEEE, 2020, pp. 547–552.
- [95] R. S. Latha *et al.*, "Automatic detection of tea leaf diseases using deep convolution neural network," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2021, pp. 1–6.
- [96] S. Mukhopadhyay, M. Paul, R. Pal, and D. De, "Tea leaf disease detection using multi-objective image segmentation," *Multimed Tools Appl*, vol. 80, pp. 753–771, 2021.
- [97] J. Chen, Q. Liu, and L. Gao, "Deep convolutional neural networks for tea tree pest recognition and diagnosis," *Symmetry (Basel)*, vol. 13, no. 11, p. 2140, 2021.
- [98] S. Datta and N. Gupta, "A novel approach for the detection of tea leaf disease using deep neural network," *Procedia Comput Sci*, vol. 218, pp. 2273–2286, 2023.
- [99] S. Jayanthi, V. M. Sathyendraa, K. P. Sumedh, and S. Suresh, "Tea Leaf Disease Classification and Tea Bud Identification," in *2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, IEEE, 2022, pp. 1–5.
- [100] L. Trihardianingsih, A. Sunyoto, and T. Hidayat, "Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3," *Sinkron: jurnal dan penelitian teknik informatika*, vol. 8, no. 3, pp. 1564–1573, 2023.

- [101] V. Tanwar and S. Lamba, "Tea Leaf Diseases Classification and Detection using a Convolutional Neural Network," in *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, IEEE, 2023, pp. 498–502.
- [102] J. A. Pandian, S. N. Nisha, K. Kanchanadevi, A. K. Pandey, and S. K. Rima, "Grey Blight Disease Detection on Tea Leaves Using Improved Deep Convolutional Neural Network," *Comput Intell Neurosci*, vol. 2023, 2023.
- [103] R. E. Uhrig, "Introduction to artificial neural networks," in *Proceedings of IECON'95-21st Annual Conference on IEEE Industrial Electronics*, IEEE, 1995, pp. 33–37.
- [104] A. H. Boussabaine, "The use of artificial neural networks in construction management: a review," *Construction Management & Economics*, vol. 14, no. 5, pp. 427–436, 1996.
- [105] J. D. Cowan, "Discussion: McCulloch-Pitts and related neural nets from 1943 to 1989," *Bull Math Biol*, vol. 52, no. 1–2, pp. 73–97, 1990.
- [106] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *J Pharm Biomed Anal*, vol. 22, no. 5, pp. 717–727, 2000.
- [107] J. Zupan, "Introduction to artificial neural network (ANN) methods: what they are and how to use them," *Acta Chim Slov*, vol. 41, p. 327, 1994.
- [108] R. Venkatesan and B. Li, *Convolutional neural networks in visual computing: a concise guide*. CRC Press, 2017.
- [109] W. Zhang, J. Tanida, K. Itoh, and Y. Ichioka, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of annual conference of the Japan Society of Applied Physics*, Montreal, CA, 1988.
- [110] C. Mouton, J. C. Myburgh, and M. H. Davel, "Stride and translation invariance in CNNs," in *Southern African Conference for Artificial Intelligence Research*, Springer, 2020, pp. 267–281.
- [111] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J Physiol*, vol. 195, no. 1, pp. 215–243, 1968.

- [112] J. Donahue *et al.*, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *International conference on machine learning*, PMLR, 2014, pp. 647–655.
- [113] R. Zhao, W. Ouyang, H. Li, and X. Wang, “Saliency detection by multi-context deep learning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1265–1274.
- [114] A. Toshev and C. Szegedy, “DeepPose: Human pose estimation via deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.
- [115] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol Cybern*, vol. 36, no. 4, pp. 193–202, 1980.
- [116] Y. Le Cun, B. Boser, J. Denker, D. Henderson, W. Hubbard, and L. Jackel, “Handwritten Digit Recognition Width a Backpropagation Network. Advances in Neural Information Systems.” Morgan Kaufman: San Mateo, CA, USA, 1990.
- [117] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Adv Neural Inf Process Syst*, vol. 25, 2012.
- [118] A. Berg, J. Deng, and L. Fei-Fei, “Large scale visual recognition challenge (ILSVRC), 2010,” URL [http://www. image-net. org/challenges/LSVRC](http://www.image-net.org/challenges/LSVRC), vol. 3, 2010.
- [119] A. Krizhevsky and G. Hinton, “Convolutional deep belief networks on cifar-10,” *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
- [120] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.
- [121] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 818–833.

- [122] O. Rippel, J. Snoek, and R. P. Adams, “Spectral representations for convolutional neural networks,” *Adv Neural Inf Process Syst*, vol. 28, 2015.
- [123] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.
- [124] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, “Multi-scale orderless pooling of deep convolutional activation features,” in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part VII 13*, Springer, 2014, pp. 392–407.
- [125] M. D. Zeiler and R. Fergus, “Stochastic pooling for regularization of deep convolutional neural networks,” *arXiv preprint arXiv:1301.3557*, 2013.
- [126] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *arXiv preprint arXiv:1412.6806*, 2014.
- [127] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial intelligence and statistics*, PMLR, 2015, pp. 192–204.
- [128] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*, PMLR, 2013, pp. 1139–1147.
- [129] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.” A field guide to dynamical recurrent neural networks. IEEE Press In, 2001.
- [130] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
- [131] Y. LeCun *et al.*, “Handwritten digit recognition with a back-propagation network,” *Adv Neural Inf Process Syst*, vol. 2, 1989.
- [132] C. Szegedy *et al.*, “Going deeper with convolutions (2014),” *arXiv preprint arXiv:1409.4842*, vol. 10, 2014.

- [133] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, pmlr, 2015, pp. 448–456.
- [134] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [135] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the AAAI conference on artificial intelligence*, 2017.
- [136] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [137] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [138] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach Learn*, vol. 20, pp. 273–297, 1995.
- [139] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.
- [140] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*, vol. 3. Wiley New York, 1973.
- [141] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.
- [142] R. Bhaumik *et al.*, “Multivariate pattern analysis strategies in detection of remitted major depressive disorder using resting state functional connectivity,” *Neuroimage Clin*, vol. 16, pp. 390–398, 2017.

- [143] R. Cuingnet, M. Chupin, H. Benali, and O. Colliot, "Spatial and anatomical regularization of SVM for brain image analysis," *Adv Neural Inf Process Syst*, vol. 23, 2010.
- [144] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 83–99, 2013.
- [145] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," *Data mining techniques for the life sciences*, pp. 223–239, 2010.
- [146] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.
- [147] H. Xian, C. Yang, H. Wang, and X. Yang, "A modified sine cosine algorithm with teacher supervision learning for global optimization," *IEEE Access*, vol. 9, pp. 17744–17766, 2021.
- [148] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput Oper Res*, vol. 13, no. 5, pp. 533–549, 1986.
- [149] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Ann Oper Res*, vol. 63, pp. 511–623, 1996.
- [150] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.
- [151] M. Khajehzadeh, M. R. Taha, A. El-Shafie, and M. Eslami, "A survey on meta-heuristic global optimization algorithms," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 3, no. 6, pp. 569–578, 2011.
- [152] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.
- [153] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: sparrow search algorithm," *Systems science & control engineering*, vol. 8, no. 1, pp. 22–34, 2020.
- [154] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in engineering software*, vol. 114, pp. 163–191, 2017.

- [155] A. Louchart, N. Tourment, and J. Carrier, “The earliest known pelican reveals 30 million years of evolutionary stasis in beak morphology,” *J Ornithol*, vol. 152, no. 1, pp. 15–20, 2011.
- [156] J. G. T. Anderson, “Foraging behavior of the American white pelican (*Pelecanus erythrorhynchos*) in western Nevada,” *Colonial Waterbirds*, pp. 166–172, 1991.
- [157] K. Zervoudakis and S. Tsafarakis, “A mayfly optimization algorithm,” *Comput Ind Eng*, vol. 145, p. 106559, 2020.
- [158] E. Dominguez, *Ephemeroptera of south America*, vol. 2. Pensoft Publishers, 2006.
- [159] B. L. Peckarsky, A. R. McIntosh, C. C. Caudill, and J. Dahl, “Swarming and mating behavior of a mayfly *Baetis bicaudatus* suggest stabilizing selection for male body size,” *Behav Ecol Sociobiol*, vol. 51, pp. 530–537, 2002.
- [160] J. D. Allan and A. S. Flecker, “The mating biology of a mass-swarming mayfly,” *Anim Behav*, vol. 37, pp. 361–371, 1989.
- [161] D. E. Goldberg, “Genetic Algorithms in Search,” *Optimization, Machine Learning*, 1989.
- [162] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.
- [163] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-international conference on neural networks*, IEEE, 1995, pp. 1942–1948.