Chelonian Conservation And Biology

Vol. 18 No. 2 (2023) | https://www.acgpublishing.com/ | ISSN - 1071-8443

DOI: doi.org/10.18011/2023.11(2).1187.1207

COTTON LEAF DISEASE CLASSIFICATION USING GREY WOLF OPTIMIZATION BASED DEEP NEURAL NETWORK

[1] T. Kalaiselvi and [2] V. Narmatha

[1] Research Scholar, [2] Assistant Professor

Email: [1]trkalaiselvi82@gmail.com Email: [2]balaji.narmatha8@gmail.com

Department of Computer and Information Science, Faculty of Science, Annamalai University, Annamalai nagar, Tamilnadu, India-608002.

ABSTRACT

One of the most current study issues in the field of agriculture is the classification of disease from a plant's leaf images. The use of image processing techniques to identify plant diseases in agricultural plants will reduce the need for farmers to protect their crops. For the image acquisition, pictures of cotton plant leaves for normal, bacterial blight, anthracnose, cercospora leaf spot, and alternaria disease datasets are collected from www.kaggle.com. A Gath-Geva (G-G) Fuzzy clustering is used to separate the diseased area from the normal component. The classification of cotton leaf diseases are carried out by using Grey Wolf Optimization (GWO) based Deep Neural Network (DNN).

Keywords: Grey wolf optimization, G-G Fuzzy Clustering, Cotton leaf, Image processing and Neural Network

1. Introduction

Cotton, known as "white gold" and a major income source for 58% of India's population, covers 2.5% of plantable land globally. Detecting microscopic issues in vast cotton fields is challenging. Technology offers solutions: drones and satellite imagery for precision monitoring, AI for data analysis, genetic modification for disease resistance, and integrated pest management. These innovations can enhance cotton yield, quality, and sustainability, benefiting both farmers and economies. Karthika et al. [1] used K-means clustering and multi-class SVM to classify cotton leaf diseases. In contrast, Bhushanamu et al.[2] employed active contour, Fourier feature descriptor, and a 1D Deep Neural Network to detect curl disease across different leaf sizes and orientations. Alves et al. [3] developed a cotton pest classification system using RGB field images and a novel ResNet34* deep learning model, achieving 98% f-score for pest identification. Chen et al. [4] utilized bidirectional LSTM units in a recurrent neural network to predict 10 cotton pests and

diseases based on climate factors, achieving a 0.95 AUC. These studies offer valuable advancements in automated pest and disease management for cotton agriculture.

Khairnar, K., & Goje, N [5] proposed an automated system for early leaf spot disease segmentation in cotton plants. Utilizing image processing and K-means clustering, the method accurately detects diseases like Bacterial Blight, Alternaria leaf spot, and Cercospora leaf spot, enabling effective disease identification. In 2021, Caldeira et al [6]. presented a deep learning approach for cotton leaf screening, using GoogleNet and Resnet50 CNN models. With precision rates of 86.6% and 89.2%, respectively, these models surpassed traditional methods, indicating potential for rapid, reliable plant inspections and improved cotton crop management.

Chi B.J et al.[7] conducted research on intercropping's impact on reducing cotton pests and diseases. The study explores intercropping's mechanisms, risks, and prospects for pest management. The goal is to offer insights for ecologically friendly strategies that minimize chemical use and cut cotton production costs. Singh et al.[8] introduced the chromatic aberration method for identifying cotton bolls under natural lighting, essential for cotton harvesting robots. Outperforming other algorithms, this method achieved a 91.05% identification rate with low false positives (6.99%) and false negatives (4.88%), proving its precision in accurate cotton boll detection during harvesting. Tripathy, S [9] employed Gaussian low-pass filtering and thresholding-based segmentation for cotton disease diagnosis. Nagaraju, M., & Chawla, P. [10] achieved 96.4% accuracy in cotton disease and pest detection using a DNN model. Both methods enhance agricultural disease identification and management.

The existing literature indicates a scarcity of authors addressing optimization techniques for weight updates in Deep Neural Networks (DNNs). This study aims to fill this gap by utilizing the Grey Wolf Optimization (GWO) algorithm for DNN weight updates. The research employs the Gath-Geva (G-G) fuzzy clustering algorithm for segmentation and combines it with a Grey Wolf Optimization-based Deep Neural Network for classification. The study focuses on identifying diseases like Bacterial blight, Anthracnose, Cercospora leaf spot, and Alternaria in cotton plants, leveraging a dataset comprising 10000 images. This approach integrates advanced optimization techniques and neural networks to enhance disease identification accuracy in agricultural applications.

The proposed system involves image collection, median filter pre-processing, Gath-Geva fuzzy clustering-based segmentation, and disease classification using a Grey Wolf Optimization-based Deep Neural Network. This process is visually represented in Figure 1, demonstrating the sequential flow of operations from image processing to disease classification.

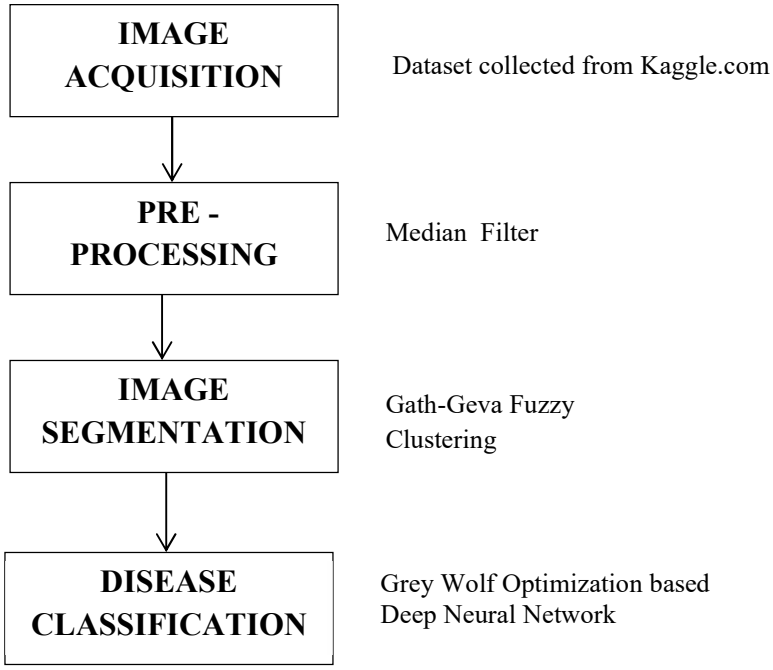

Fig1. Proposed system flow diagram.

The study uses a robust median filter for noise reduction [11]. Segmentation categorizes the pre-processed image into leaf, root, and soil regions [12]. The Gath-Geva (G-G) Fuzzy Clustering Algorithm improves subsequent analysis accuracy, aiding disease diagnosis and classification in the proposed system.

3. Segmentation using Gath-Geva Fuzzy Clustering Algorithm

The Gath-Geva (G-G) clustering algorithm employs a distance norm based on FMLE introduced by Bezdek and Dunn [13]. It is similar to the G-K algorithm, with an exponential term in the distance norm computation for quicker decrease. The iterative process is conducted using MATLAB software, repeating steps for j values starting from 1.

Step (1): Calculate the cluster prototypes (means):

$$V_i^{(j)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(j-1)})^w Z_k}{\sum_{k=1}^{N} (\mu_{ik}^{(j-1)})^w}, 1 \le i \le c \quad (1)$$

Step (2): Calculate the cluster covariance matrices:

$$F_{wi} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(j-1)})^{w} (Z_k - V_i^{(j)})(Z_k - V_i^{(j)})^{T}}{\sum_{k=1}^{N} (\mu_{ik}^{(j-1)})^{w}}, \quad (2)$$

The priori probability of selecting cluster i is given by

$$P_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{ik}^{(j-1)} \quad (3)$$

where $1 \le i \le c$,

Step (3): Compute the distances:

$$D_{ik}^2(z_k, v_i) = \frac{\sqrt{\det(F_{wi})}}{P_i} * \exp\left(\frac{1}{2}\left(z_k - v_i^j\right)^T F_{wi}^{-1}\left(z_k - v_i^j\right)\right)$$

where $1 \le i \le c, 1 \le k \le N$ (4)

Step (4): Update the partition matrix:

If $D_{ik}F_{wi} > 0$ for $1 \le i \le c, 1 \le k \le N$,

$$\mu_{ik}^{(j)} = \frac{1}{\sum_{n=1}^{c} (D_{ik}F_{wi}/D_{nk}F_{wi})^{2/(m-1)}}$$

Otherwise

$$\mu_{ik}^{(j)} = 0 \text{ if } D_{ik}F_{wi} > 0, \text{ and } \mu_{ik}^{(j)} \in [0,1] \quad (5)$$

$$\text{with } \sum_{i=1}^{c} \mu_{ik}^{(j)} = 1$$

Until $\| U^{(j)} - U^{(j-1)} \| < \varepsilon$.

4. Grey Wolf Optimizer (GWO)

Grey Wolf Optimization (GWO), algorithm by Mirjalili et al. [14], draws from gray wolves' social behavior to address optimization problems. It initializes a population of "wolves" as potential solutions. Iteratively updating their positions, GWO mimics collaboration and hierarchy to explore and converge toward optimal solutions through a sequence of steps.

Step (1), the alpha wolf (α) with the best fitness is identified. The beta (β) and delta (δ) wolves are the second and third best, respectively. Positions of these wolves update using their current positions and others' to optimize.

Step (2) Encircling Prey:

(a) Update the positions of the remaining wolves by imitating the encircling behavior of wolves when hunting prey.

(b) Calculate the distance between each wolf and the alpha, beta, and delta wolves.

(c) Adjust the position of each wolf by moving it closer to the prey (i.e., the alpha, beta, or delta wolf) using specific equations.

Step (3) Exploration and Exploitation:

(a) Encourage exploration by incorporating randomization into the search process.
(b) Update the positions of the wolves using randomization and a search agent's randomness parameter.

(c) Adjust the position of each wolf by adding a random value to its current position.

These steps are repeated for a predetermined number of iterations or until a convergence criterion is met. The convergence criterion can be a maximum number of iterations, reaching a specific fitness threshold, or other stopping conditions.The GWO algorithm balances exploration and exploitation by using the alpha, beta, and delta wolves as search leaders and incorporating randomization to explore new areas of the search space. This approach allows GWO to effectively search for global optima in complex optimization problems.

Grey wolves (Canis lupus) are apex predators in the Canidae family, living in packs with 5–12 members on average. They exhibit a strict social dominance hierarchy, as illustrated in Fig. 2.

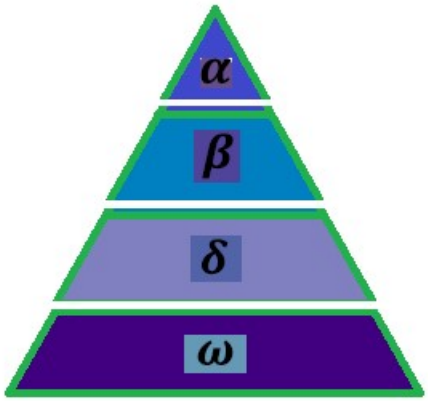

Fig. 2. Hierarchy of grey wolf (dominance decreases from top down).

The study mathematically models grey wolves' hunting and social hierarchy to design the Grey Wolf Optimization (GWO) algorithm. Equations are proposed to represent encircling behavior in optimization tasks.

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \quad (6)$$

$$\vec{X}(t+1) = \vec{X}_P(t) - \vec{A}.\vec{D} \quad (7)$$

The variable 't' denotes the current iteration., $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\overrightarrow{X_p}\ (t)$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf. The vectors $\vec{A}$and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \quad (8)$$

$$\vec{C} = 2.\vec{r}_2 \quad (9)$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and $\overrightarrow{r_1}, \overrightarrow{r_2}$ are random vectors in [0, 1]. In GWO, the top three best solutions are stored, guiding other agents to update positions. Formulas are used to adjust positions based on these top agents.

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_\alpha - \vec{X}\right|,\ \vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}\right|,\ \vec{D}_\delta = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}\right| \quad (10)$$

$$\overrightarrow{X}_1 = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha),\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta),\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta), \quad (11)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (12)$$

In GWO algorithm, the mathematical models of the social hierarchy consist of; tracking, encircling, and attacking prey are description in Mirjalili et al. [17].

5 Classification using Grey Wolf Optimization based DNN

Grey Wolf Optimization (GWO) is a metaheuristic algorithm inspired by the hunting behavior of grey wolves in nature. It can be used as an optimization technique for various problems, including weight updation in Convolutional Neural Networks (DNNs). However, it's worth noting that the standard backpropagation algorithm with gradient descent is the most commonly used method for weight updation in DNNs.

Step(1) Define the objective function: The objective function represents the fitness measure that you want to optimize. In the context of weight updation in DNNs, network's performance metric, such as accuracy used as the objective function to maximize.

Step(2) Initialize the grey wolf population: Create an initial population of grey wolves. Each wolf represents a potential solution, where the solution corresponds to a set of weights in the DNN. The population size can be determined based on the problem complexity and computational resources.

Step(3) Evaluate the fitness: For each wolf in the population, evaluate the DNN's performance by setting the weights according to the wolf's solution and computing the corresponding fitness value using the objective function.

Step(4) Update the weights using GWO: Employ the GWO algorithm to update the weights based on the fitness values of the wolves. The update equation for the weights can be derived from the position update equation in the standard GWO algorithm, adapted to the weight update process in DNNs. The equation typically involves updating the weights using a combination of the current weights, best weights, and the average weights of the wolves.

Step(5) Apply convergence criteria: Determine the stopping criteria for the algorithm. It can be a predefined number of iterations or a threshold for the fitness value. If the criteria are not met, repeat steps 3 and 4 until convergence.

Step(6) Retrieve the best solution: Once the algorithm converges, retrieve the best set of weights found during the optimization process. These weights represent the solution that optimizes the objective function.

Step(7) Update the DNN weights: Finally, update the weights of the DNN using the best solution obtained from the GWO algorithm. This can involve assigning the weights to the corresponding layers in the DNN architecture.

The DNN architecture, as depicted in Fig. 3, consists of two hidden layers aimed at learning the input-output mapping. The architecture incorporates preference weight fitness consideration. During training, GWO iteratively updates hidden layer node weights, calculated using Eqn. (13).

$$N = \sqrt{a + b} + c \tag{13}$$

In this context, the input layer nodes are 'a', the output layer nodes are 'b', the hidden layer nodes are 'n', and a constant value between [1,10] is denoted as 'c'. To enable nonlinear fitness, a hidden layer activation function is employed. In this study, the softmax function is used as the activation function, denoted as:

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_i)} \tag{14}$$

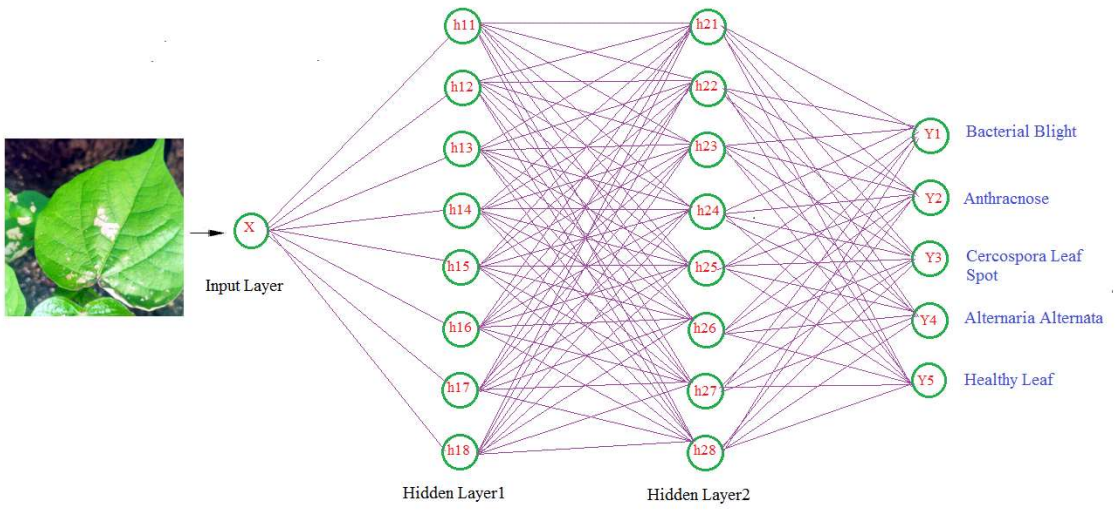

Fig.3. Architecture of DNN with two hidden layers.

The input data of the network is termed as x and it is activated by the mapping function M_f.

$$M_f = smax\ (\omega_i X + \beta_i) \tag{15}$$

The loss form can be computed as

$$S(w_S, b_S; x, 1) = \frac{1}{2m}\sum_{j=1}^{m}\left\|h_i(w_S, b_S; x) - l_j\right\|^2 \tag{16}$$

Here, 'Ws' and 'bs' are subsets of biases, and 'm' is the count of neurons in the hidden layer. Cross-entropy serves as the DNN's loss function for training and testing. It notably enhances sigmoid and softmax model performance. The cross-entropy loss is calculated using Eqn. (17).

$$C_E = \frac{1}{N}\sum_{K=1}^{N}\left[Y_k log\hat{Y}_k + (1 - Y_k)\log\ (1 - \hat{Y}_k)\right] \tag{17}$$

In the equation, 'n' represents training samples, 'Yk' is actual output, and 'Yk' is expected output. GWO optimizes DNN weights, refining fitness. Updated values guide solutions. Best solutions persist, replaced or retained based on improvement. The process iterates until termination criteria. Weight updating via GWO is shown in Fig. 3(b).

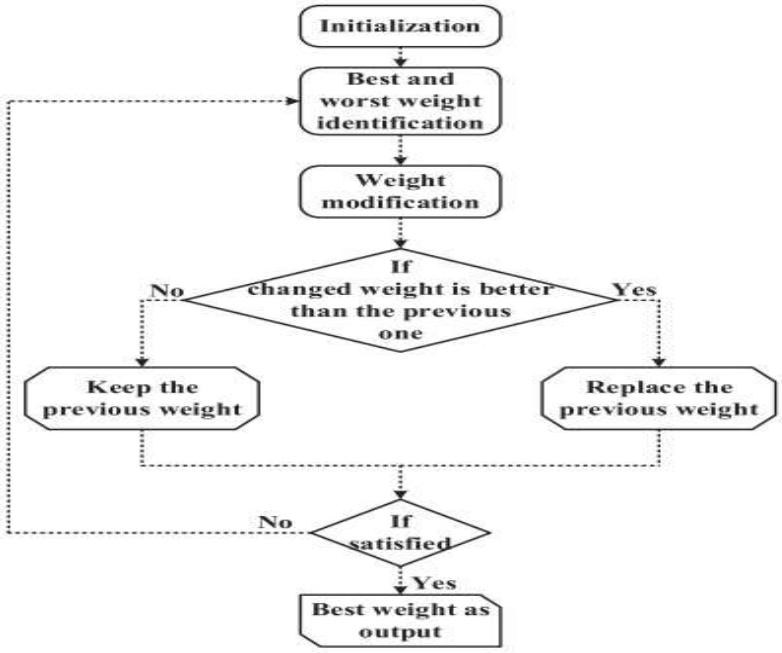

Fig.3(b). Flow chart of Weight updation using GWO.

5. Experimental Results and Discussion

The obtained weight matrices for input layer to hidden layer 1, hiddenlayer1 to hidden layer2 and hidden layer2 to output layer are tabulated in Table 1,2 and 3 respectively.

Table 1. Weight Matrix for Input to Hidden layer1

Neuron	H11	H12	H13	H14	H15	H16	H17	H18	H19	H110	H111	H112	H113	H114	H115	H116	H117	H118	H119	H120
X	1.062545	-1.07871	0.176041	-0.09908	-1.85361	-0.74631	-2.02397	0.54812	0.479694	0.417084	-1.82494	0.840013	-0.38127	0.302144	-1.1801	-0.99713	-0.79075	0.266206	0.652625	-0.40335

Table 2. Weight Matrix for Hidden layer1 to Hidden layer2

Hidden Neuron	H21	H22	H23	H24	H25	H26	H27	H28	H29	H210	H211	H212	H213	H214	H215	H216	H217	H218	H219	H220
H11	1.248907	-0.7341	-0.75175	0.379134	0.517345	-0.41443	2.105517	-1.14917	0.79782	1.427052	-0.97595	1.342006	0.159957	0.303905	-0.01354	-0.2293	-0.40236	-0.29779	-0.50548	-1.23951
H12	0.254302	-1.31133	-1.40544	0.78444	-0.0333	0.004106	0.708791	-0.27755	1.313281	0.098277	-0.40677	1.928647	-0.56617	0.380918	1.526195	-1.6617	-0.74882	-0.78013	0.364785	1.479306

H13	-1.05047	0.648327	2.017473	-0.16859	-0.30687	-0.32981	-0.31733	-0.2503	0.396805	2.731013	0.97161	-0.09983	0.449855	0.480625	0.131862	-1.62642	-0.74186	0.971332	-0.53667	-1.56668
H14	-0.58327	0.920207	0.764964	0.616735	0.780345	-0.55558	-0.37134	0.016124	0.264532	-0.44337	-2.0397	1.469765	-0.37445	-0.7708	-0.13291	-1.40969	0.74355	-1.02217	1.15665	-3.00716
H15	-1.12591	0.077569	0.558733	1.856998	-0.08488	0.734638	-0.42314	-1.42264	0.56754	0.174296	-0.53107	0.096486	1.092855	-1.15592	-2.737	0.813589	0.589692	-1.48523	1.140209	-0.31551
H16	-0.49951	0.250399	0.448598	0.130444	-0.2766	-0.96681	0.348964	-0.96829	-0.35286	-0.97569	0.604751	-0.64591	-1.33086	-0.7964	1.396697	-1.02008	-2.40621	1.298777	-0.89135	2.148818
H17	0.555646	-0.15901	0.208231	0.041804	0.988642	0.794513	-0.73778	-1.5173	0.417955	-1.10475	-0.08541	-0.12563	-0.20074	-0.3942	0.054138	0.525842	-0.04946	-0.16511	1.086572	-0.91276
H18	0.697	1.094	0.2295	3.6733	-0.106	0.9981	0.1528	0.9177	-1.621	0.0851	0.4979	1.1271	0.8839	0.6634	0.4819	0.8356	-0.163	1.5695	-0.133	-1.358

	37	56	77	94	39	68	28	58	33	55	95	99	69	36	24	62	67	81	21	34
H19	-0.35734	-1.83152	-0.8534	-1.78833	1.398344	-0.26678	0.844619	-2.30881	0.02333	0.797386	-1.27709	1.634099	-0.28028	-1.00165	0.471222	-1.25516	0.00392	0.806329	0.139891	0.444875
H110	1.186908	-1.33713	-1.4049	-0.97598	0.053796	-0.89397	0.146966	-0.28985	0.438789	-2.00122	-0.20653	1.472383	-0.49175	-0.22358	1.340155	-0.63338	-0.60105	-2.00774	-1.35585	1.490115
H111	0.752351	1.52545	0.638481	-0.58924	0.980149	0.638568	0.207593	0.42219	-0.49221	-0.86204	0.160539	-1.78655	-0.59178	2.605697	0.935458	0.778614	0.295174	0.648225	-0.20216	-1.96724
H112	0.347657	0.137049	0.56725	1.041201	0.745917	2.331187	-1.34267	-0.79201	-0.78891	-0.50383	-2.39791	1.067865	-0.05038	-1.09775	0.687245	0.553699	0.163784	1.127126	-0.66909	-0.39866
H113	-0.3885	0.152086	-0.2812	-0.61184	-0.10391	0.116027	1.160136	1.140033	0.605516	1.55745	1.932995	0.663951	-1.50736	0.625229	0.657853	-0.11729	-0.11075	0.273603	1.314245	-0.14835
H1	0.00	-0.6	-0.9	0.23	0.07	-1.5	1.42	-0.2	1.10	-0.4	0.08	0.56	1.35	-0.2	1.11	-0.8	1.57	-1.4	1.04	0.41

14	5053	5445	6725	2293	2105	5119	5933	6034	35	2523	5808	8106	3021	9019	9911	3776	6527	6486	4837	3626
H115	-0.206006	0.260405	-0.74018	1.175628	0.502973	-1.5224	1.000791	-0.66302	0.473289	-0.50378	-0.66874	-0.40033	0.706903	-0.34872	0.422094	-0.61601	-0.37394	2.281283	-0.7165	2.422928
H116	-0.58852	-1.30303	0.506932	0.556241	-0.16393	-0.42396	0.430333	-0.44163	-0.94468	-0.03307	1.71765	-1.43242	0.896832	0.604334	0.671165	-0.12272	0.630718	-0.29133	-0.87954	0.447362
H117	-0.09761	-0.59344	-0.07256	-0.22138	-1.08633	0.049871	0.808059	1.310906	1.540343	-1.4181	0.320159	0.177933	-0.36835	-0.23226	-0.43928	0.559824	0.686822	-1.46412	0.382298	-1.39719
H118	0.056212	-0.66988	0.759424	1.572058	0.022353	-2.16942	1.972725	1.181733	-0.26695	0.053021	-0.21361	0.109749	-0.75909	-0.48104	0.828042	0.285614	-2.75952	0.388618	0.301741	0.184007
H119	1.732232	0.112615	1.414349	-0.94058	-1.14397	-1.4388	-1.76782	-1.51089	-0.58109	0.81973	0.704795	1.709245	-0.06373	0.175814	1.412387	1.137881	-0.1388	-1.09981	0.063287	0.293036

H120	1.48619	-0.3825	-0.76181	-0.07784	-0.66958	1.29594	0.952817	0.386894	0.006984	-0.61999	-0.89444	0.589623	1.073914	-0.00982	-1.49134	1.027008	-1.16271	0.230493	-1.50975	0.360461

Table 3. Weight Matrix for Hidden Layer2 to output

Neuron	H21	H22	H23	H24	H25	H26	H27	H28	H29	H210	H211	H212	H213	H214	H215	H216	H217	H218	H219	H220
Y1	1.028662	1.004564	0.720538	0.14465	0.872821	0.758731	-0.08298	-0.56509	0.472884	0.3815	0.059515	0.53737	-0.00063	-0.96268	-1.52861	-0.47044	0.833209	-0.59805	-0.50061	-0.84944
Y2	-0.20968	0.469154	2.438443	1.220945	0.992067	-1.00258	-2.37227	2.031811	1.474721	-0.45519	1.714052	0.655834	0.885622	1.441415	-0.83581	-0.70362	-1.16367	-0.85907	-0.07158	-0.45899
Y3	-0.76869	0.292836	0.988736	-0.44668	-0.6417	0.976393	-0.16845	-2.13779	-1.57971	-1.97489	-0.77174	0.858429	0.953137	0.790254	-0.7424	0.198127	1.50988	-1.73639	2.165601	0.847359
Y4	0.99	0.15	0.89	0.40	0.85	0.07	1.71	1.15	-0.6	0.04	-0.2	1.96	-0.7	-0.9	1.01	0.30	0.83	1.63	-0.1	-1.1

	5	9	5	8	7	8	4	6	8	7	0	0	7	0	1	2	0	6	1	4
	8	8	0	4	4	8	4	1	5	1	0	0	4	4	5	1	7	3	0	2
	1	3	9	3	8	6	2	7	4	8	3	4	3	2	6	4	8	4	2	6
	1	1	7	9	8	7		6	2		1	5	3	9	5	3		9		
Y	1.	0.	-	-	-	0.	0.	-	-	0.	-	-	-	-	1.	-	-	1.	-	-
5	4	0	1.	0.	1.	5	1	0.	1.	2	0.	0.	1.	1.	2	1.	0.	5	0.	1.
	4	2	9	8	6	9	0	5	6	1	0	2	5	2	9	2	6	5	4	1
	7	6	6	8	7	9	0	4	4	2	0	9	8	4	5	7	8	6	2	5
	5	4	2	4	2	6	6	7	2	3	9	8	0	0	6	8	0	8	8	3
	4	8	3	5	7	9	3	4	9	9	8	0	8	3	8	0	0	4	7	5
	1	6	1	5	7	1	3	8	5	7	3	9	9	2	8	2	8	1	5	8

The following are the output images that have been pre-processed, segmented and classified using the above mentioned proposed algorithms.

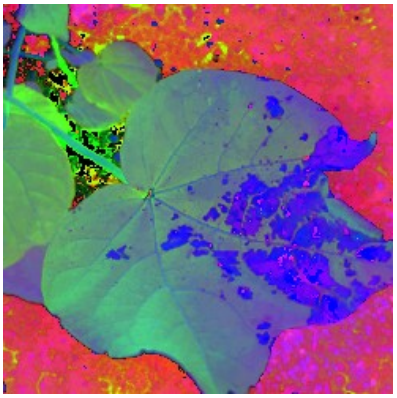

Fig.4 (a) . Input Image 1 **Fig.4 (b) Pre-processed image** **Fig.4(c) Segmented image**

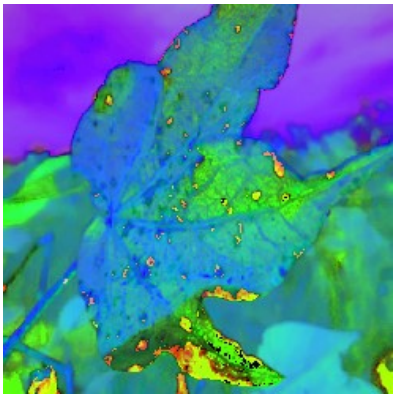

Fig.5 (a) . Input Image 1 **Fig.5 (b) Pre-processed image** **Fig.5(c) Segmented image**



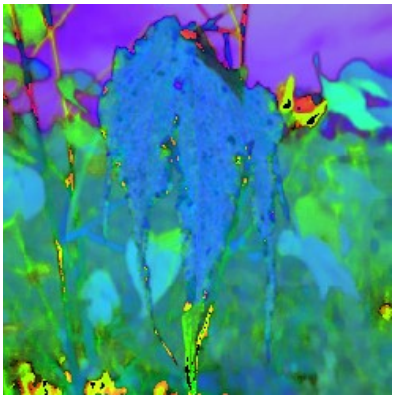

Fig.6 (a) . Input Image 1 **Fig.6 (b) Pre-processed image** **Fig.6(c) Segmented image**



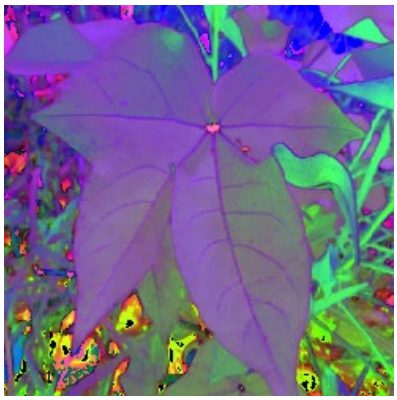

Fig.7 (a) . Input Image 1 **Fig.7 (b) Pre-processed image** **Fig.7(c) Segmented image**



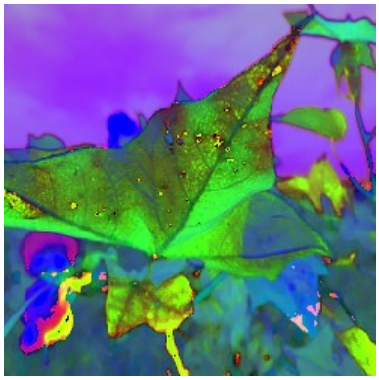

Fig.8 (a) . Input Image 1 **Fig.8 (b) Pre-processed image** **Fig.8(c) Segmented image**



Fig.9 (a) . Input Image 1 **Fig.9 (b) Pre-processed image** **Fig.9(c) Segmented image**



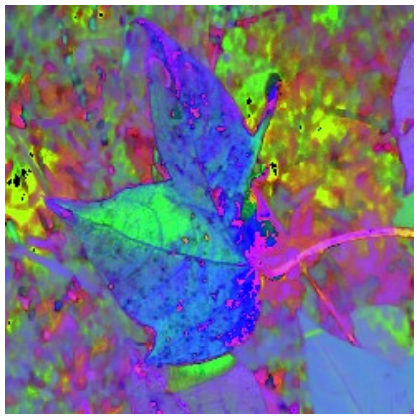

Fig.10 (a) . Input Image 1 **Fig.10 (b) Pre-processed image** **Fig.10(c) Segmented image**



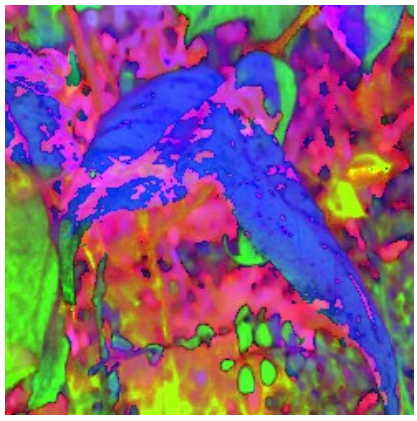

Fig.11 (a) . Input Image 1 **Fig.11 (b) Pre-processed image** **Fig.11(c) Segmented image**



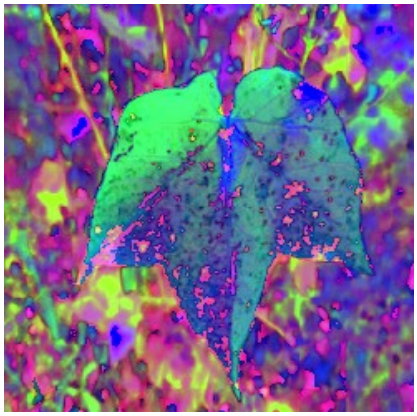

Fig.12 (a) . Input Image 1 Fig.12 (b) Pre-processed image Fig.12(c) Segmented image



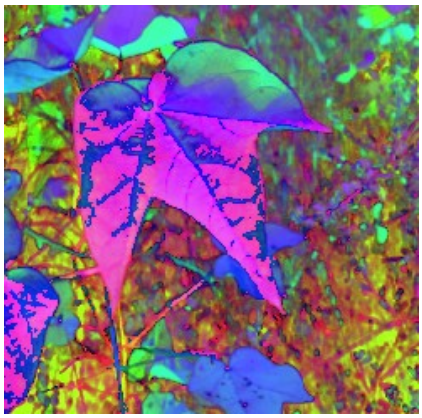

Fig.13 (a) . Input Image Fig.13 (b) Pre-processed image Fig.13(c) Segmented image



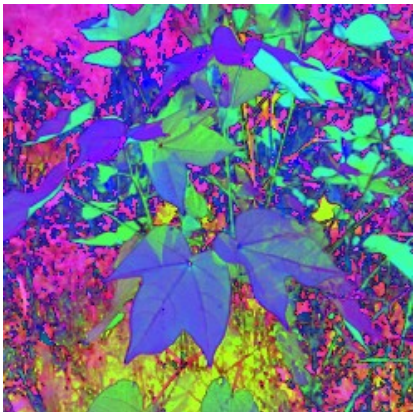

Fig.14 (a) . Input Image 1 Fig.14 (b) Pre-processed image Fig.14(c) Segmented image

6.1 Classification Performance Metrics

A dataset of 10,000 images was employed to classify four cotton diseases: Bacterial Blight, Anthracnose, Cercospora Leaf Spot, Alternaria, and Healthy leaves. Further details can be found in Table 4.

Table 4. Classification Accuracy

S.No	Disease	Samples taken	GWO-DNN Rightly detected
1.	Bacterial blight	2000	1926
2.	Anthracnose	2000	1920
3.	Cercospora Leaf Spot	2000	1916
4.	Alternaria Alternata	2000	1956
5.	Healthy Leaf	2000	1946
	Total	10000	9664

Other performance metrics such as precision, recall, F-Score and specificity are calculated and presented in Table 2.

Table 5. Classification Performance Metrics.

S.No	Name of the Disease	Samples	Precision	Recall	F-Score	Specificity
1	Bacterial blight	2000	0.89	0.91	0.90	0.91
2	Anthracnose	2000	0.89	0.89	0.89	0.90
3	Cercospora Leaf Spot	2000	0.90	0.89	0.88	0.92
4	AlternariaAlternata	2000	0.90	0.88	0.91	0.90
5	Healthy Leaves	2000	0.90	0.88	0.89	0.90

The graphical representation of Precision, recall, F-score and specificity for various cotton diseases are presented in Fig15. The classification accuracy graph is presented in Fig. 16.

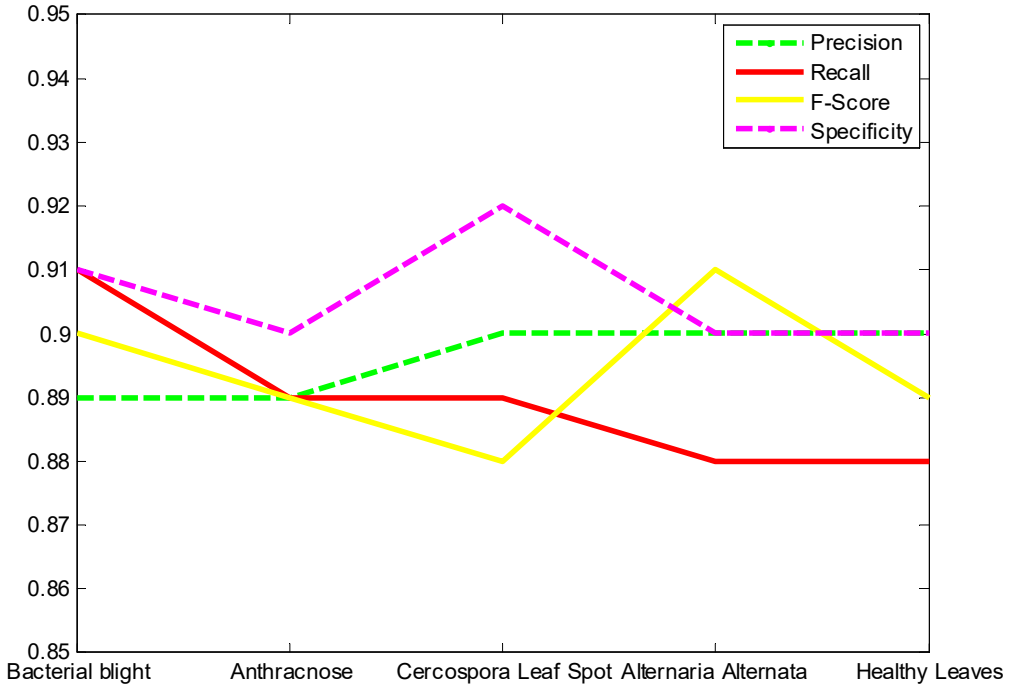

Fig.15. The graphical representation of Precision, recall, F-score and specificity for various cotton diseases using GWO-DNN Classifier.

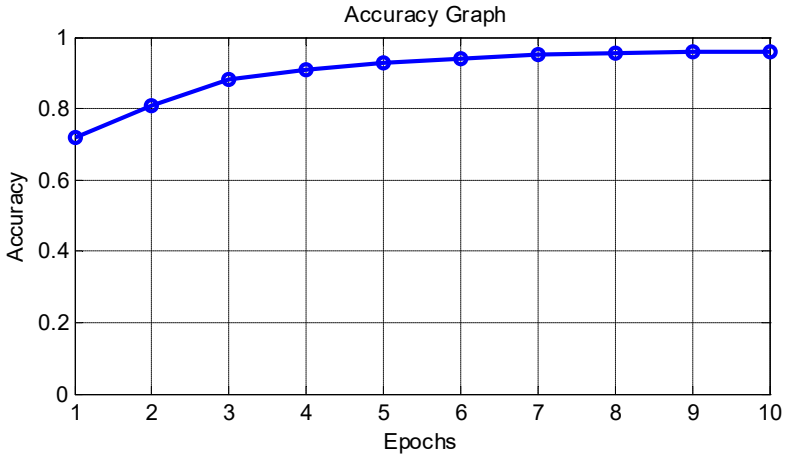

Fig.16. Response of classification accuracy.

5. Conclusion

This paper delves into automated disease detection and classification within cotton plants, utilizing a methodology comprising Gath-Geva clustering for segmentation and GWO-DNN for classification. The experimental findings highlight the GWO-DNN classifier's effectiveness, achieving an impressive accuracy of 96.64%. Presently, the system focuses on single leaf images as input, while future developments aim to expand its scope to batch processing, enabling a more

comprehensive analysis covering various parts of the plant. This advancement could significantly enhance disease monitoring and contribute to improved agricultural practices. The combined utilization of segmentation and classification techniques showcases the potential of such integrated approaches to address challenges in the field of plant disease detection, marking a significant stride towards automated and accurate agricultural management.

REFERENCE

1. Karthika, J., Santhose, M., & Sharan, T. (2021, May). Disease detection in cotton leaf spot using image processing. In *Journal of Physics: Conference Series* (Vol. 1916, No. 1, p. 012224). IOP Publishing.
2. Bhushanamu, M. B. N., Rao, M. P., & Samatha, K. (2020). Plant curl disease detection and classification using active contour and Fourier descriptor. *European Journal of Molecular & Clinical Medicine*, *7*(05), 2020.
3. Alves, A. N., Souza, W. S., & Borges, D. L. (2020). Cotton pests classification in field-based images using deep residual networks. *Computers and Electronics in Agriculture*, *174*, 105488.
4. Chen, P., Xiao, Q., Zhang, J., Xie, C., & Wang, B. (2020). Occurrence prediction of cotton pests and diseases by bidirectional long short-term memory networks with climate and atmosphere circulation. *Computers and Electronics in Agriculture*, *176*, 105612.
5. Khairnar, K., & Goje, N. (2020). Image processing based approach for diseases detection and diagnosis on cotton plant leaf. In *Techno-Societal 2018: Proceedings of the 2nd International Conference on Advanced Technologies for Societal Applications-Volume 1* (pp. 55-65). Springer International Publishing.
6. Caldeira, R. F., Santiago, W. E., & Teruel, B. (2021). Identification of cotton leaf lesions using deep learning techniques. *Sensors*, *21*(9), 3169.
7. Chi, b. J., zhang, d. M., & dong, h. Z. (2021). Control of cotton pests and diseases by intercropping: a review. *Journal of Integrative Agriculture*, *20*(12), 3089-3100.
8. Singh, N., Tewari, V. K., Biswas, P. K., Pareek, C. M., & Dhruw, L. K. (2021). Image processing algorithms for in-field cotton boll detection in natural lighting conditions. *Artificial Intelligence in Agriculture*, *5*, 142-156.
9. Tripathy, S. (2021, November). Detection of cotton leaf disease using image processing techniques. In *Journal of Physics: Conference Series* (Vol. 2062, No. 1, p. 012009). IOP Publishing.
10. Nagaraju, M., & Chawla, P. (2021, September). Plant Disease Classification using DDNN-19 Convolutional Neural Networks. In *2021 9th international conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO)* (pp. 1-6). IEEE.

11. Devasena, D., & Jagadeeswari, M. (2017). FPGA Implementation of Speckle noise removal in Real Time Medical Images. *Journal of Medical imaging and Health Informatics*, *7*(6), 1263-1270.
12. Kalaiselvi, T., & Narmatha, V. (2023). Cotton Crop Disease Detection Using FRCM Segmentation and Convolution Neural Network Classifier. In *Computational Vision and Bio-Inspired Computing: Proceedings of ICCVBIC 2022* (pp. 557-577). Singapore: Springer Nature Singapore.
13. Bezdek, J. C., & Dunn, J. C. (1975). Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, *100*(8), 835-838.
14. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, *69*, 46-61.