An Efficacious Detecting Tomato Leaf Disease Using RCOA-Based RNN Method

T. George Princess * and E. Poovammal

Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chennai, India
Email: georgeprincess1995@gmail.com (T.G.P.); poovamme@srmist.edu.in (E.P.)
*Corresponding author

***Abstract*—Plant diseases significantly affect both the world's agricultural economy and food security on a worldwide scale. The likelihood of establishing efficient control measures is increased by their early discovery and classification. Convolutional Neural Networks (CNN)-based categorization of tomato leaf diseases using RGB photos has shown encouraging results in several recent research. Despite their usefulness, CNN models have limitations. Sometimes, they fail to focus on the specific areas affected by plant disease and instead may include irrelevant backgrounds or healthy plant parts in their categorization. This research introduces a new approach for identifying diseased areas and extracting relevant features for illness classification using an Rider Chicken Optimization Algorithm (RCOA)-based Recurrent Neural Network (RNN). Compared to traditional CNN techniques, the RNN-based approach is more robust and can better generalize to unknown crop species that are affected. The classification is based solely on distinct features to achieve the highest accuracy. The proposed RCOA-trained RNN classifier is used to classify diseases. To improve the accuracy of the classification results, fictional computing combines the premise of rider optimization with the hierarchical and swarming conduct of chickens to manage huge volumes of data. The suggested RCOA-based RNN is capable of precisely finding infectious illnesses by analyzing the area of focus with 98.8% accuracy in tomato leaves.**

***Keywords*—Recurrent Neural Network (RNN), Rider Chicken Optimization Algorithm (RCOA), plant village, Convolutional Neural Networks (CNN), plant disease**

I. Introduction

Tomatoes are extensively grown vegetables worldwide, serving as a significant source of nutrition for millions of individuals. Nevertheless, tomato crops can be severely affected by various diseases, resulting in considerable economic losses for farmers and a shortage of food supply for consumers [1]. Therefore, detecting and categorizing tomato crop diseases has become a crucial area of study. With new computer vision algorithms' advancements, accurate and efficient detection and classification of tomato crop diseases are now achievable [2].

Previously, the process of identifying and categorizing tomato crop diseases was carried out manually by human experts, which is time-consuming and costly. However, with the introduction of new technologies such as machine learning and computer vision, it is now possible to automate this process. Several algorithms have recently been developed to detect and classify tomato crop diseases with high accuracy [3].

In the realm of agricultural technology, machine learning algorithms have emerged as powerful tools for automating the detection and classification of crop diseases. One such algorithm is Convolutional Neural Networks (CNNs), which have demonstrated remarkable efficacy in identifying various diseases affecting tomato crops. CNNs operate by extracting features from images, enabling them to discern intricate patterns essential for accurate classification. However, a limitation of CNNs lies in their inability to encode spatial information, thereby necessitating a substantial volume of training data for optimal performance [4].

Another notable algorithm in the domain of crop disease classification is the Support Vector Machine (SVM). SVMs excel in discerning complex boundaries between different classes of data, maximizing classification confidence by optimizing the decision surface. By defining a hyperplane with a significant margin of separation, SVM ensures robust categorization, even in scenarios where linear separation in high-dimensional space is unattainable [5].

Recurrent Neural Networks (RNNs) transitioning from conventional machine learning methods to more sophisticated techniques have garnered attention for their ability to model sequential data effectively. In the context of disease detection in tomato crops, RNNs offer a promising avenue for capturing temporal dependencies in disease progression. Building upon the principles of neural networks, RNNs can analyze sequential data, such as time-series observations of crop health, and discern patterns indicative of disease onset or progression. This capability makes RNNs well-suited for tasks requiring the analysis of sequential data streams, including disease monitoring in agricultural settings [6].

k-nearest neighbor is a common supervised machine learning method that is straightforward and easy to use. As a result of its simple interpretation and short computation time compared to other machine learning algorithms. Regression and classification issues may both be solved using KNN. The drawback of this approach is that as the

Manuscript received December 9, 2023; revised February 21, 2024; accepted April 28, 2024; published January 9, 2025.

size of the data set increases, so does the algorithm's efficiency. KNN is well-known for its slowdowns. Due to the requirement to keep all training examples and the additional time required to calculate the distance to all samples, this method is computationally costly. Missing data can't be handled [6].

Finally, Random Forest is another algorithm that has shown promising results in detecting and classifying tomato crop diseases. In the case of tomato crop disease detection, Random Forest can be used to classify images of tomato crops into healthy and diseased categories based on the features extracted from them. This technique has the drawback of being inefficient in real-time prediction situations. Complex algorithm that is indeed difficult to put into practice.

RNNs have been used in the development of a number of agricultural applications in recent years. Pattern recognition, such as in automated diagnostic systems, is a popular use of RNNs. For classification, RNNs may make use of nonlinear decision limits and state memory. Much research has shown that RNNs can tell the difference between linear and nonlinear data. RNNs have been shown in previous research to be very effective in the healthcare industry. This study found that the classification accuracy of RNNs was better than MLPs when compared to other research that examined their findings [7].

Chicken Swarm Optimization (CSO), a brand-new bio-inspired method, is suggested for optimization applications. Tests on twelve benchmark datasets were carried out to correlate the performance of CSO with that of other algorithms [8]. The outcomes demonstrate that CSO may produce strong optimization outcomes in terms of both robustness and accuracy. The work is bifold as:

(1) To achieve high accuracy in identifying diseases in tomato leaves, we will employ an RNN with RCOA and investigate its efficacy on the plant village dataset.
(2) To examine how the train/test split size amount of the dataset affects the performance of the network model in identifying diseases in tomato leaf images. For the chosen dataset it is analyzed with different split ratios to determine their impact.

II. Literature Review

An important type of neural network is the Recurrent Neural Network (RNN), which creates a directed graph of connections between layer nodes that follow a series of variables. This RNN-based method is popular for handling sequential data and making predictions in areas such as language translation or action recognition, as shown in studies by Ma *et al.* [9]. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are improved RNN models that address issues like disappearing gradients and enable training on extended sequences.

Meanwhile, Zhao *et al.* [10] showed that LSTM could capture distinguishing areas of images for accurate categorization. RNNs can also use attention as a mechanism to focus on certain inputs while predicting specific outputs, which simplifies the learning process and leads to higher-quality learning. For example, Ren and Zemel [11] used an RNN with an attention mechanism to capture spatial features.

Paudel *et al.* [12] suggested a brand-new RNN-based framework for the detection of illnesses affecting maize crops. Long Short-Term Memory (LSTM) cells of the RNN architecture were used in the suggested framework to efficiently describe the temporal relationships in the sequence of input data. A publicly accessible dataset of 9,692 photos of maize leaves affected with common rust, grey leaf spot and northern leaf blight was used in the study. The suggested RNN-based framework outperformed other cutting-edge techniques by categorizing the various diseases with an accuracy of 95.5%. In a different research, Gouda *et al.* [13] suggested a deep learning-based method for identifying and categorizing illnesses affecting cotton crops. The sequential patterns in the input data were recognized by the research using an RNN architecture termed a Gated Recurrent Unit (GRU). Images of cotton leaves afflicted with bacterial blight, bacterial leaf spot, cotton leaf curl virus, and powdery mildew made up the input data. The suggested RNN-based strategy outperformed other conventional and deep learning-based algorithms, categorizing the various illnesses with an accuracy of 96.8%.

Lu *et al.* [14] developed an RNN-based method for identifying and categorizing tomato illnesses. The suggested method modelled the temporal relationships in the input data series using an LSTM architecture. Images of tomato leaves affected with six distinct illnesses made up the input data: septoria leaf spot, grey mold, late blight, mosaic virus, and bacterial wilt. The suggested RNN-based strategy outperformed other conventional and deep learning-based systems, categorizing the various illnesses with an accuracy of 96.3%.

Li *et al.* [15] suggested an RNN-based method for identifying and categorizing apple illnesses. The suggested method modelled the temporal relationships in the input data series using an Attention-based LSTM (ALSTM) architecture. The input data consisted of pictures of apple leaves with four different diseases: apple scab, cedar apple rust, frogeye leaf spot, and healthy leaves. The suggested RNN-based strategy outperformed other conventional and deep learning-based systems, categorizing the various illnesses with an accuracy of 96.5%.

An RNN-based strategy using RCOA was suggested by Chen *et al.* [16] for the prediction of tomato leaf diseases. The network weights were optimized using the suggested method, which combined an LSTM architecture with the RCOA algorithm [17]. Images of tomato leaves affected with two distinct diseases—early blight and late blight—made up the input data. The suggested RNN-based method outperformed previous cutting-edge approaches, predicting tomato leaf diseases with an accuracy of 95.2%.

While Vinoth and Ananth [17] may have utilized a similar approach based on RCOA for RNNs in disease detection, our research introduces several novel elements that set it apart. Firstly, we have refined and optimized the underlying algorithmic framework, enhancing its efficacy

in capturing subtle patterns indicative of tomato leaf diseases. Our modifications include fine-tuning parameters, adjusting optimization strategies, and incorporating domain-specific knowledge to improve overall performance. Additionally, we have extended the scope of application by integrating auxiliary data sources, such as environmental variables or genetic profiles, to augment disease detection accuracy. Furthermore, our research introduces innovative post-processing techniques tailored to address specific challenges encountered in agricultural settings, such as variability in lighting conditions or leaf morphology. By leveraging these advancements, our approach not only achieves superior classification accuracy but also demonstrates robustness and adaptability across diverse environmental contexts. Overall, while Vinoth and Ananth [17] may provide a foundation for RCOA-based RNNs in disease detection, our research significantly advances the state-of-the-art by introducing novel methodologies and optimizations tailored specifically for the challenges inherent in tomato crop disease detection

Tomato leaf prediction is a crucial task in the agricultural industry for ensuring the health and quality of tomato crops. Recurrent Neural Networks (RNNs) have been investigated by researchers as a potential tool for predicting tomato leaf diseases thanks to developments in machine learning and artificial intelligence.

III. Materials and Methodologies

Plant Village dataset was utilized to construct the tomato leaf dataset [18]. The collection comprises 54,306 leaf pictures with 38 class labels, including some healthy tomato leaves and nine different forms of tomato leaf illnesses is depicted in Table I. Ten of these categories, which are especially connected to tomato plant leaves, were the focus of the investigation. Due to its genetic heritage, the tomato plant is vulnerable to several bacilli-caused illnesses. There by eighteen thousand three hundred and forty photos of tomato plant leaves were included in the dataset for the study, along with information on management methods, genetic heritage, and environmental changes. Fig. 1 classifies the different illnesses into those caused by fungus, bacteria, mold, viruses, or mites.

TABLE I. Plant Village Dataset Samples with Table Description

Type	Class	Origin Images	Augmentation	Trainset
Bacterial Spot	0	425	2125	1700
Early Blight	1	480	2400	1920
Healthy	2	481	2405	1924
Late Blight	3	463	2315	1852
Leaf Mold	4	470	2350	1880
Mosaic Virus	5	448	2240	1792
Septoria Leaf Spot	6	436	2180	1744
Target Spot	7	457	2285	1828
Two-Spotted Spider Mite	8	435	2175	1740
Yellow Leaf Curl Virus	9	490	2450	1960
Total		**4585**	**22,925**	**18,340**

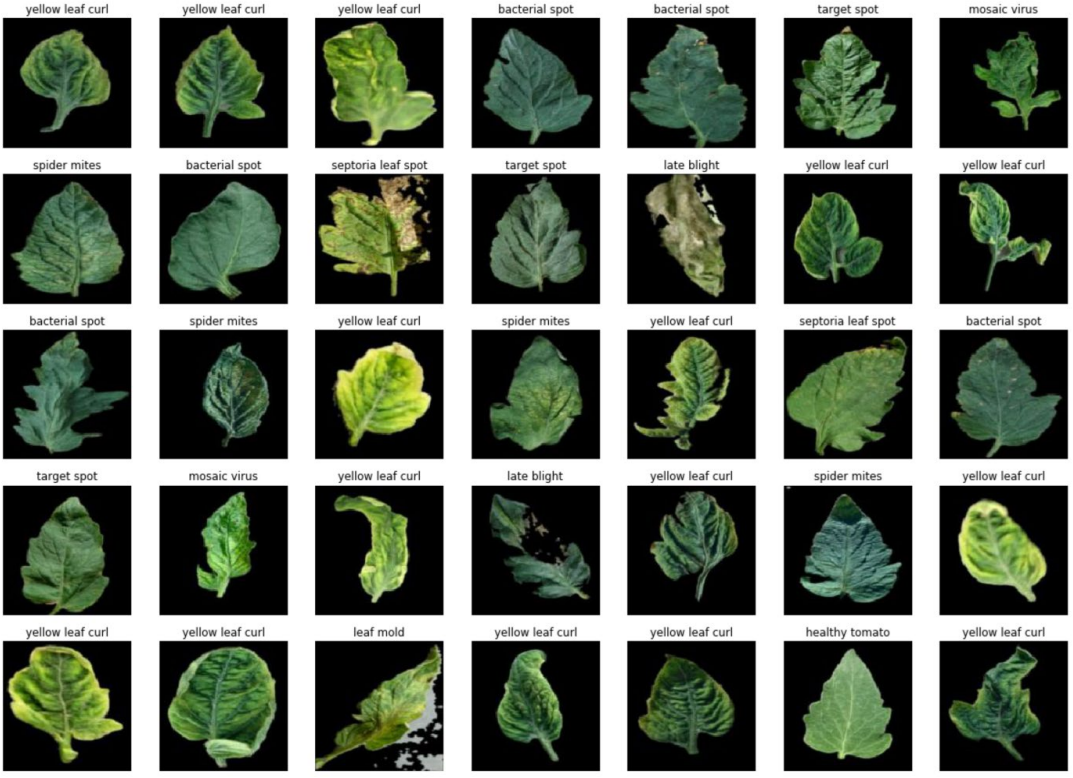

Fig. 1. Plant village dataset samples.

A. Comprehensive Architecture

Fig. 2 depicts the RNN classifier's three layers: the output layer, the hidden recurrent layer, and the input layer. To train the RNN classifier, the RCOA approach is used, which seeks to obtain the ideal weight for tweaking the classifier for effective and accurate classification of huge datasets. An assortment of time-dependent vectors makes up the input layer. The RNN model's hidden layer gets information from the input layer, and the weight matrix establishes connections between the hidden units. Recurrent connections in the hidden layer of the RNN unite the hidden units based on time.

Hence, the RNN produces its output vector by utilizing the weight on the input vector. The recommended RCOA approach is used to train the RNN classifier to ensure precise classification of extensive data as shown in Fig. 3.

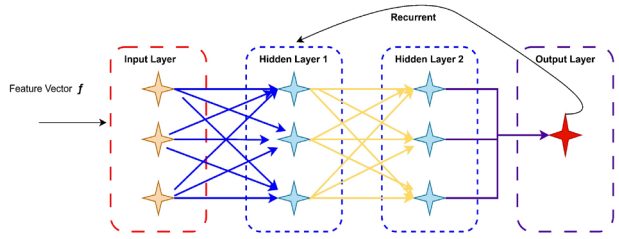

Fig. 2. RNN architecture.

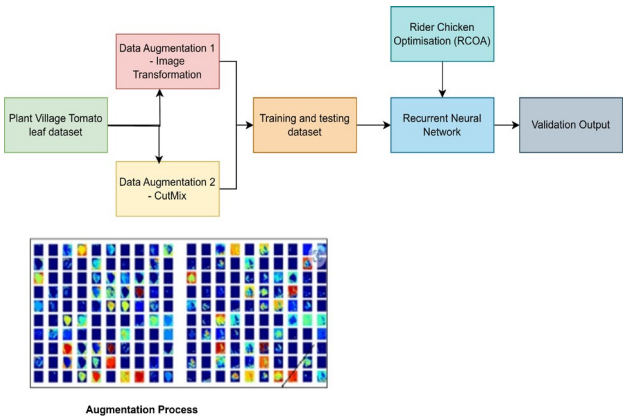

Fig. 3. Overall architecture of RCOA-based RNN

To improve the accuracy of the classification results, only unique selected features are used. The RCOA-trained RNN classifier is employed for the classification of large datasets. The hierarchical and swarming behavior of chickens is integrated with the fictitious computing concept of the ROA to obtain the best possible classification results for massive datasets. The Adaptive RCOA is utilized to train both classifiers, the RNN and the DSAE, for large data classification.

The Adaptable ROA incorporates the adaptive concept by combining ROA's fictitious computing concept with the hierarchical and swarming behavior of virtual chickens. CSO, on the other hand, is a bio-inspired optimization method used to locate food within the swarm group. Chicks, roosters, and hens comprise the three subgroups of the swam group. Rider optimization is a fictitious computer concept based on four different types of riders: overtaker, attacker, follower, and bypass rider. The classification performance of swarm groups and rider groups may be improved by merging their characteristics.

B. Data Pre-processing

It includes two steps. To prepare the data for the model, pre-processing techniques are utilized which involve cleaning and organizing the data.

1) Data cleaning and organization

The dataset containing images of tomato leaves from ten different classes in the Plant Village dataset undergoes thorough cleaning to remove any corrupted or irrelevant images. This ensures that only high-quality data is used for training.

The images are organized into folders corresponding to each of the ten classes, facilitating easy access and management during preprocessing and training.

2) Image transformations for data expansion

Various image transformations are applied to augment the dataset and increase its diversity. These transformations are crucial for exposing the model to a wide range of variations in tomato leaf images, helping prevent overfitting.

For instance, images may be randomly rotated, translated, scaled, sheared, or flipped to simulate different perspectives and orientations of the leaves within each class.

3) Resizing images to 128×128

After augmentation, all images are resized to a standard size of 128×128 pixels. This uniform resizing ensures that the neural network model receives consistent input dimensions across all images in the dataset, simplifying the training process.

4) CutMix augmentation

CutMix augmentation is applied to pairs of images selected randomly from the ten classes in the dataset. For each pair, a rectangular patch is cut out from one image and replaced by a patch from the other image.

By blending portions of two distinct tomato leaf images, CutMix encourages the model to learn from diverse examples within and across classes, enhancing its ability to generalize to unseen data.

This process of mixing images from different classes promotes robust feature extraction and classification, as the model learns to recognize common patterns and variations shared among the different classes.

5) Label blending

In conjunction with the CutMix process, the ground truth labels of the resulting mixed images are blended based on the area ratios of the patches from each original image.

This ensures that the labels assigned to the mixed images accurately reflect the content and characteristics of the combined patches, effectively incorporating information from both original images into the training process.

By following this tailored preprocessing pipeline, the tomato leaf images from the ten classes in the Plant Village dataset are effectively prepared for training deep learning models. The combination of data augmentation techniques and CutMix regularization contributes to improved model

performance, robustness, and generalization across the diverse range of tomato leaf classes.

C. Algorithm for Rider Chicken Optimization Algorithm

The algorithm helps to determine the best solution by calculating the fitness function [19]. All of the greatest-value chickens are approved as roosters, which is the finest option. Swarm groups, on the other hand, with each rooster acting as a leader. Chicks are given to the chickens with the lowest fitness values, while the rest of the chickens are designated as hens. It's based on the Minkowski distance and the fitness function is mentioned in Eq. (1).

$$F = \left[\sum_{r=1}^{D} |x_r - Q_r|^t \right]^{1/t} \quad (1)$$

where, F is fitness value, t is positive integer, X_r is output of classifier and Q_r is estimated output in the proposed RCOA, the following are the algorithmic steps:

Initialization of the population: All virtual chicks are assumed to be in the same location, with the time stamp "b" and the chick's quest for food in "L" space. As a result, roosters are believed to have the greatest fitness value among chicks, whereas chicks are thought to have the worst. Calculation of the most optimum solution for data categorization is done using a fitness function, which is defined in Eq. (1).

The updated position of the Rooster: The fittest roosters within the swarm groups are the ones who can access the food, while the other virtual chickens are kept away from it.

Position update of the hen: Hen follows the rooster's trail as they look for food. The hen grabs food discovered by other virtual chickens at random. However, the more powerful hens have greater benefits as in Eq. (2).

$$P_{b+1}(x,y)=P_b(x,y)[1-N_1 rand-N_2 rand]+N_1 rand\ P_b(Q_1,y) + N_2\ rand\ P_b(Q_2,y) \quad (2)$$

For successful data categorization, which is changed by a follower's updated Eq. (3) in the rider groups.

$$P_{b+1}(x,y) = P^*(J,y) + [cos\left(y_{x,y}^b\right) \times P^*(J,y) \times g_x^b] \quad (3)$$

The above equation is modified as further in Eq. (4):

$$P_{b+1}(x,y) = \frac{1 + cos\left(Y_{x_1 y}^b\right) g_x^b}{cos\left(Y_{x,y}^b\right) g_x^b + N_1\, rand + N_2\, rand}$$

$$[N_1\ rand\ P_b(Q_1,y) + N_2\ rand\ P_b(Q_2,y)] \quad (4)$$

Here's the final modified formula for the planned RCOA, which incorporates both rider and chicken swarm elements in Eqs. (5) and (6).

$$N_1 = exp\left(\frac{(F_l - F_{c1})}{abs(F_1)+d}\right) \quad (5)$$

$$N_2 = exp\ (F_{c2} - F_l) \quad (6)$$

Chicks' positions are updated as represented in Eq. (7):

$$P_b(x,y) = P_b(x,y) + k \times (P_b(\lambda, y) - P_b(x,y)) \quad (7)$$

Once the best solution has been found or the conditions have been met, these processes are repeated to get the optimal results.

D. Experimental Setup

The training and testing procedures of the experimentation were carried out using Pytorch and Python libraries including scikit-learn, pillow, and OpenCV, along with the fastai library with 3320 GB of SSD storage. The detailed description is shown in Table II.

TABLE II. EXPERIMENTAL SETUP

Parameter	Details
Processor	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHZ
Server model	DELL PowerEdge T640 Tower Server
Graphics	CUDA-based video cards 4X 1080TI
OS	Linux
Language	Python 3
Framework	Pytorch
Image input size	256×256
Drop out	0.5

IV. DISCUSSION OF THE FINDINGS

This section delves into a deeper analysis of the suggested RCOA-based RNN model, evaluating its performance using sensitivity, specificity, and accuracy metrics. Precision, Accuracy and F1-score are three common measures used to evaluate the performance of classification models. The datasets are typically divided into training and validation sets to train and evaluate models, respectively. A training split refers to the portion of the dataset that is used to train the model. Typically, a larger proportion of the dataset is used for training, as it is used to optimize the model's parameters and weights through an iterative process. The goal is for the model to learn patterns and relationships in the training data that will enable it to make accurate predictions on new, unseen data. The performance of the model on the validation set can be used to tune hyperparameters, such as the batch size, learning rate, or regularization strength, which can improve the model's efficiency on data as mentioned in Table III.

TABLE III. PERFORMANCE METRICS

Train Split (%)	Validation Split (%)	Train Loss	Valid Loss	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
40	60	0.047	0.06	97	96.9	96.8	96.8
50	50	0.043	0.07	97	97.5	97.2	97.6
60	40	0.041	0.08	97	97.31	97.5	97.8
70	30	0.033	0.06	98	98.4	98.2	98.6
80	20	0.002	0.04	99	98.6	98.5	98.8

Accuracy is typically improved when using an 80/20 split of training and validation data, as opposed to using a smaller training set, for several reasons:

- An 80/20 split gives the model more data to train on, allowing it to discover more intricate patterns and connections in the data. Better performance on the validation set and ultimately on fresh, untested data may result from this.

- Reduced overfitting: A larger training set can help reduce the risk of overfitting, which occurs when the model memorizes the training data rather than learning generalizable patterns. By having more data to learn from, the model is less likely to memorize specific examples and more likely to learn generalizable patterns that can be applied to new data.

More representative validation set: With a larger validation set, the model is evaluated on a more representative sample of the data, which can provide a more accurate estimate of its performance on new, unseen data. A smaller validation set may not be as representative of the overall distribution of the data, which can lead to over-optimistic estimates of the model's performance.

Overall, an 80/20 split of training and validation data can provide a good balance between providing enough data or the model to learn and reducing the risk of overfitting, while still providing a representative sample of the data for evaluation. However, the optimal split may vary depending on the specific dataset and problem being solved.

Figs. 4 and 5 illustrate the F1-Score outcomes for different ratios of train-validation data splits in the recommended RCOA-based RNN. Evaluate the F1-Score of the trained network on the validation set for each batch size and data split ratio. Plot the F1-Scores of the network against the batch size for each data split ratio, to visualize the correlation between batch size and network performance. Repeat the above steps for different network architectures, hyperparameters, and optimization algorithms to determine the optimal combinations for F1-Score. It is important to note that the optimal combination may vary based on the precise dataset and problem being solved, so it is essential to experiment with different hyperparameters and network architectures to find the best combination.

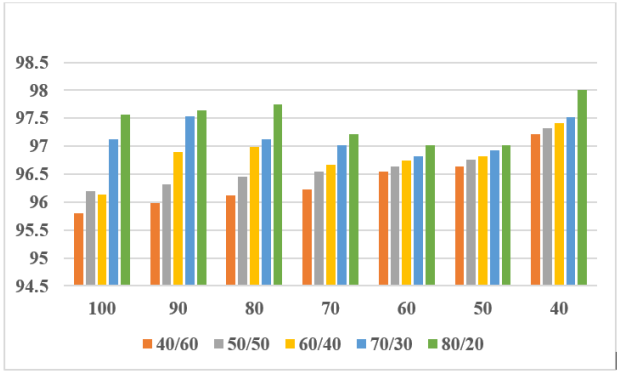

Fig. 4. F1-score for various validation data splits

The findings in Figs. 4 and 5 illustrate the connection between the networks' F1-Score and the train-validation data split ratio. These findings imply that the split ratio significantly affected the network's performance. As the number of train samples is increased, Fig. 5 clearly demonstrates a difference in the performance value. Overall, using an 80/20 split of training and validation data can provide several benefits for improving F1-Score. By having more data for training and better hyperparameter tuning, the model is more likely to learn generalizable patterns and avoid overfitting.

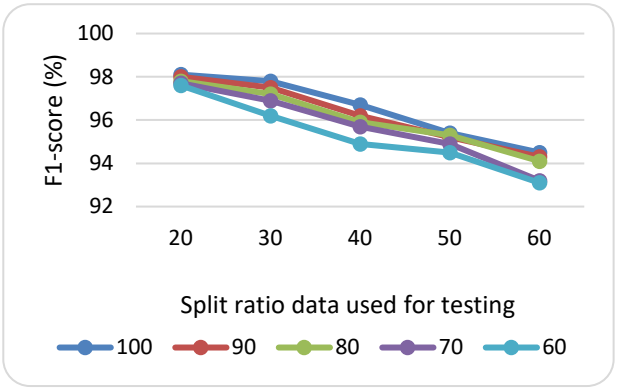

Fig. 5. F1-Scores for various testing data splits.

The model was tested with various train-validation data split ratios. The outcomes of these tests are presented and discussed in this section. Table IV illustrates the correlation between batch size and network performance at different data split ratios.

TABLE IV. PERFORMANCES OBTAINED FOR VARIOUS BATCH SIZES

Batch Size	Performances in Percentage (%)				
	40/60	50/50	60/40	70/30	80/20
100	96.70	97.40	97.8	98.20	98.80
90	96.20	97.10	97.00	98.10	98.27
80	95.13	95.78	96.68	97.60	98.19
70	95.57	95.68	96.54	97.54	98.09
60	95.34	95.54	96.23	97.34	97.91
50	95.28	95.32	96.12	97.23	97.84
40	94.98	95.12	96.09	97.12	97.68

Table V depicts the time spent using different train-validation split ratios and batch sizes.

TABLE V. COMPUTATIONAL TIME FOR VARIOUS BATCH SIZE

Batch Size	Time (s)				
	40/60	50/50	60/40	70/30	80/20
100	180	194	199	215	220
90	175	186	208	221	230
80	169	182	204	225	232
70	167	172	190	204	221
60	159	167	178	196	201
50	162	174	181	197	210
40	172	185	192	202	215

While our goal was to improve performance, not just speed, we were able to improve the network training and testing process by using more validation data than training data. This led to a non-optimized outcome despite faster processing times. Table V demonstrates that using an 80/20 split ratio resulted in the best training and testing outcomes. Having more data for validation than for training sped up the network training and testing process, while our objective was to increase performance as well as speed. As a result, the result of the fastest time was not the

optimum result we had. As shown in Table IV the split ratio of 80/20 produced the best results in training and testing.

TABLE VI. COMPREHENSIVE CORRELATION OF PROPOSED WORK WITH OTHER EXISTING WORKS

Year	Author Name	Number of Images	Technique Used	Accuracy (%)
2020	Agarwal *et al.* [20]	18160	VGG 16	93.15
2020	Karthik *et al.* [21]	5452 (4 Classes)	ResNet+DenseNet	98
2021	Lamba *et al.* [22]	16012	CNN	97.2
2021	Zhao *et al.* [23]	18160 (10 Classes)	ResNet50	96.81
2022	Zhao *et al.* [24]	18160	Spatial Attention CNN	98.49
2022	Mukherjee *et al.* [25]	10,839 (7 classes)	Gray Wolf+MobileNetV2	98
2023	Proposed	18160 (10 classes)	RCOA-based RNN	98.8

This study examined how well a recurrent neural network (RCOA-based RNN) could recognize healthy tomatoes and distinguish them from diseased ones across ten different classes of data with varying batch sizes and parameter values. Table VI presents a comprehensive comparison of our proposed approach with other advanced models in the field. Based on the data in Table VI, the proposed model performed better than previous models, including Mukherjee *et al.*'s [25] model, which had the closest performance of 98%. Our model also achieved a favorable performance with a +1.62% improvement compared to Lamba *et al.*'s [22] model.

The results showed that the RCOA-based RNN performed admirably on both training and test data, with the highest recognition accuracy value of 98.8% and a train/validation data split size of 80/20 with a batch size of 40. To find the best ratio for splitting training and testing data in a certain research field, several ratios were tried out: 40/60, 50/50, 60/40, 70/30, and 80/20. The results showed that the 80/20 ratio was the most effective, followed by 70/30. Additionally, various batch sizes were used, depending on the available GPU capacity in the laboratory, to see how they affected the model's training and testing results. The graphs indicated that smaller batch sizes led to slightly better performance, but the most significant impact was on the training speed.

V. CONCLUSION

It is essential to pre-process internet-sourced data using normalization algorithms in order to properly organize data with diverse field characteristics. But in terms of data classification, this might be a difficult work. In order to overcome these difficulties, the RCOA approach—which combines ROA and CSO—was created. This technique achieves improved classification results by using the hierarchical structure and emulating behaviors of chickens. In addition, rider optimization is used to increase precision and dependability while reducing computing complexity. The new Adaptive RCOA accelerates training and does away with the requirement to choose a learning schedule and speed while providing a quick convergence rate and avoiding local minima. The study found that while increasing the batch size did not significantly affect overall performance, it did delay obtaining stable results. In addition, the optimal train/test split ratio was determined through experimentation with ratios of 40/60, 50/50, 60/40, 70/30, and 80/20, with the 80/20 ratio being the most effective, followed by 70/30. The study also found that smaller batch sizes led to slightly better performance, but the most significant impact was on training speed. The proposed approach therefore performs better than the state-of-the-art methods.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest in this work.

AUTHOR CONTRIBUTIONS

George Princess conducted the research gaps and analyzed the data; Poovammal has given the ideas to implement the paper in well manner; both authors had approved the final version.

REFERENCES

[1] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," arXiv preprint, arXiv:1511.08060, 2015.

[2] S. Panno, S. Davino, A. G. Caruso *et al.*, "A review of the most common and economically important diseases that undermine the cultivation of tomato crop in the Mediterranean basin," *Agronomy*, vol. 11, 2188, 2021.

[3] H. Durmus, E. O. Gunes, and M. Kirci, "Disease detection on the leaves of the tomato plants by using deep learning," in *Proc. the 2017 6th International Conference on Agro-Geoinformatics, Fairfax*, 2017, pp. 1–5.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. the 3rd International Conference on Learning Representations, ICLR 2015*, May 2015, pp. 1–14.

[5] F. A. P. Rani, S. Kumar, A. L. Fred *et al.*, "K-Means clustering and SVM for plant leaf disease detection and classification," in *Proc. the 2019 International Conference on Recent Advances in Energy-Efficient Computing and Communication (ICRAECC)*, 2019, pp. 1–4.

[6] P. S. Kanda, K. Xia, and O. H. Sanusi, "A deep learning-based recognition technique for plant leaf classification," *IEEE Access*, vol. 9, pp. 162590–162613, 2021.

[7] S. H. Lee *et al.*, "Attention-based recurrent neural network for plant disease classification," *Frontiers in Plant Science*, vol. 11, 2020.

[8] S. M. A. Razzaq and B. I. Khaleel, "Detection of plants leaf diseases using swarm optimization algorithms," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 15, no. 2, pp. 193–212, 2021.

[9] X. Ma, Z. Tao, Y. Wang *et al.*, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transport. Res. C Emerging Technol.*, vol. 54, pp. 187–197, 2015.

[10] Z. Zhao, W. Chen, X. Wu *et al.*, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, pp. 68–75, 2017.

[11] M. Ren and R. S. Zemel, "End-to-End instance segmentation with recurrent attention," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 293–301.

[12] D. Paudel, A. de Wit, H. Boogaard *et al.*, "Interpretability of deep learning models for crop yield forecasting," *Computers and Electronics in Agriculture*, vol. 206, 107663, 2023.

[13] J. Sahoo, A. Dass, M. A. Bhat *et al.*, "Land suitability assessment for improved land use planning in selected watersheds of Haryana,"

Journal of Environmental Biology, vol. 42, no. 2, pp. 285–294, 2021.
[14] Y. Lu, Y. Tian, R. Shen *et al.*, "Precise genome modification in tomato using an improved prime editing system," *Plant Biotechnol. J.*, vol. 19, no. 3, pp. 415–417, Mar. 2021. doi: 10.1111/pbi.13497
[15] Y. Wang, W. Li, X. Xu *et al.*, "Progress of apple rootstock breeding and its use," *Horticultural Plant Journal*, vol. 5, no. 5, pp. 183–191, 2019.
[16] X. Chen, G. Zhou, A. Chen *et al.*, "Identification of tomato leaf diseases based on combination of ABCK-BWTR and B-ARNet," *Computers and Electronics in Agriculture*, vol. 178, 105730, 2020.
[17] R. Vinoth and J. P. Ananth, "Rider chicken optimization algorithm-based recurrent neural network for big data classification in spark architecture," *The Computer Journal*, vol. 65, no. 8, pp. 2183–2196, August 2022.
[18] Plant village dataset. [Online]. Available: https://github.com/spMohanty/PlantVillage-Dataset
[19] R. Vinoth and J. P. Ananth, "Deep recurrent encoder network and spark model for angiographic disease risk classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 36, no. 4, 2250010, 2022.
[20] M. Agarwal, S. K. Gupta, and K. K. Biswas, "Development of efficient CNN model for tomato crop disease identification," *Sustain. Comput. Inform. Syst.*, vol. 28, 100407, 2020.
[21] R. Karthik, M. Hariharan, S. Anand *et al.*, "Attention embedded residual CNN for disease detection in tomato leaves," *Appl. Soft Comput.*, vol. 86, 105933, 2020.
[22] M. Lamba, Y. Gigras, and A. Dhull, "Classification of plant diseases using machine and deep learning," *Open Comput. Sci.*, vol. 11, pp. 491–508, 2021.
[23] S. Zhao, Y. Peng, J. Liu, and S. Wu, "Tomato leaf disease diagnosis based on improved convolution neural network by attention module," *Agriculture*, vol. 11, 651, 2021.
[24] Y. Zhao, C. Sun, X. Xu, and J. Chen, "RIC-Net: A plant disease classification model based on the fusion of inception and residual structure and embedded attention mechanism," *Comput. Electron. Agric.*, vol. 193, 106644, 2022.
[25] G. Mukherjee, A. Chatterjee, and B. Tudu, "Identification of the types of disease for tomato plants using a modified gray wolf optimization optimized MobileNetV2 convolutional neural network architecture driven computer vision framework," *Concurr. Comput. Pract. Exp.*, vol. 34, e7161, 2022.