Journal of Theoretical and Applied Information Technology
15th June 2024. Vol.102. No. 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REVOLUTIONIZING COTTON FARMING: CO-CNN INTEGRATION FOR DISEASE IDENTIFICATION AND YIELD PREDICTION

S.GOVINDASAMY[1], D.JAYARAJ[2]

[1]Research Scholar, Department of Computer & Information Science, Annamalai University, Chidambaram, Tamilnadu, India.
[2]Assistant Professor, Department of Computer Science Govt. Arts & Science College, Vadalur, Tamilnadu, India.
E-mail: [1]govindasamy1412@gmail.com, [2]jayarajvnr@gmail.com

ABSTRACT

In agriculture, accurate identification of cotton plant diseases and prediction of yield are crucial for ensuring crop health and optimizing production. This abstract explores the integration of Cassowary Optimization (CO) with Convolutional Neural Networks (CNNs) to enhance cotton plant disease identification and yield prediction. The CO-CNN framework demonstrates superior performance in accurately classifying instances and capturing underlying patterns in the data. By leveraging the dynamic optimization capabilities of CO, the model effectively optimizes CNN parameters, leading to improved convergence and performance. Results across various performance metrics, including Classification Accuracy, F-Measure, Fowlkes-Mallows Index, and Matthews Correlation Coefficient, showcase the efficacy of the CO-CNN model in addressing the complexities of real-world classification tasks. This innovative approach holds significant promise for empowering farmers and agronomists with advanced tools for early disease detection, yield prediction, and informed decision-making in crop management.

Keywords: Cotton Plant Disease - Yield prediction – CNN – Cassowary Optimization – Disease Identification.

1. INTRODUCTION

Agriculture the backbone of many economies worldwide, encompasses a diverse array of practices aimed at cultivating crops and rearing livestock. Among these practices, cotton farming holds significant importance due to cotton's versatility and widespread use in various industries. Cotton farming involves meticulous planning, from soil preparation to harvest, to ensure optimal growth and yield[1]. However, one persistent challenge faced by cotton farmers is the prevalence of leaf diseases. These diseases, caused by various pathogens such as fungi, bacteria, and viruses, can significantly impact cotton plants' health and productivity[2]. Among the most common leaf diseases affecting cotton plants are Fusarium wilt, Alternaria leaf spot, and bacterial blight. Fusarium wilt, caused by the fungus Fusarium oxysporum, leads to wilting and eventual death of the plant[3]. Alternaria leaf spot, caused by the fungus Alternaria alternata, manifests as dark lesions on leaves, reducing photosynthetic capacity[1]. Bacterial blight, caused by the bacterium Xanthomonas campestris pv. malvacearum, results in water-soaked lesions and defoliation[4].

Managing leaf diseases in cotton farming requires a combination of preventive measures and timely interventions. These may include crop rotation, use of disease-resistant varieties, proper sanitation practices, and application of fungicides or bactericides when necessary. By implementing effective disease management strategies, cotton farmers can mitigate the impact of leaf diseases and sustainably improve their crop yields[5].

In modern agricultural practices, advanced technologies play a pivotal role in disease identification and crop yield prediction. One such technology is Convolutional Neural Networks (CNN), a type of deep learning algorithm used for image recognition and analysis. CNNs are adept at analyzing vast amounts of agricultural data, including images of crops and diseased plants captured by drones or sensors[6]. By examining intricate patterns and features within these images, CNNs can accurately identify various diseases affecting crops. This capability enables farmers to swiftly detect and respond to disease outbreaks, thus minimizing crop losses and ensuring optimal yields[7].

CNNs are instrumental in predicting crop yields by analyzing factors such as weather patterns, soil quality, and historical yield data[8]. By processing these diverse datasets, CNNs can generate predictive models that forecast crop yields with remarkable precision. This predictive capability empowers farmers to make informed decisions regarding planting schedules, irrigation strategies, and resource allocation, ultimately maximizing crop productivity and profitability[9].

Bio-inspired computing enhances CNN by mimicking biological processes. Drawing inspiration from nature, algorithms emulate behaviors observed in living organisms [10]–[29]. For instance, genetic algorithms simulate natural selection to optimize CNN parameters[30]. Similarly, swarm intelligence algorithms replicate the collective behavior of organisms like ants or bees to improve CNN efficiency. These bio-inspired approaches offer novel solutions to enhance CNN performance, enabling more accurate disease identification and crop yield prediction in agriculture. By integrating principles from the natural world, bio-inspired computing contributes to the advancement of CNN technology in agricultural applications [31].

1.1. Problem Statement

In agricultural practices, the accurate identification of crop diseases and prediction of crop yields are paramount for ensuring optimal productivity and mitigating losses. Conventional methods for disease identification and yield prediction often lack efficiency and precision. Challenges arise from the complexity of analyzing vast datasets comprising diverse environmental factors and crop conditions. Existing approaches may struggle to adapt to dynamic changes in disease patterns and environmental variables. To address these issues, leveraging CNN in conjunction with bio-inspired computing offers a promising solution. By harnessing principles inspired by natural systems, such as genetic algorithms and swarm intelligence, CNNs can enhance their ability to analyze intricate patterns within agricultural data.

1.2. Motivation

The challenges posed by inaccurate disease identification and crop yield prediction in agriculture demand innovative solutions. Conventional methods fall short in effectively harnessing the wealth of data available and adapting to dynamic environmental conditions. Hence, there arises a pressing need to explore alternative approaches that can revolutionize agricultural practices. By leveraging the power of CNN and incorporating principles inspired by nature through bio-inspired computing, we aim to address these challenges head-on. Through this interdisciplinary approach, we strive to enhance the precision and efficiency of disease identification and yield prediction, thus empowering farmers to make informed decisions and optimize crop productivity.

1.3. Objectives

The objective of this research initiative is to develop innovative solutions for improving disease identification and crop yield prediction in agriculture. Drawing from the challenges outlined in the introduction, we are motivated to explore alternative approaches that can revolutionize agricultural practices. By leveraging CNN and integrating principles inspired by nature through bio-inspired computing, our aim is to enhance the precision and efficiency of disease identification and yield prediction. Through interdisciplinary collaboration and leveraging advanced technologies, we aspire to empower farmers with tools and insights to make informed decisions and optimize crop productivity. Ultimately, the objective is to contribute to a more sustainable and resilient agricultural future by addressing critical issues faced by farmers worldwide.

2. LITERATURE REVIEW

"Lightweight Inception Networks" [32] involve the utilization of Inception Networks, architecture for a deep neural network, for disease recognition and detection tasks in rice plants. The Inception Network's lightweight version is designed to balance model complexity and computational efficiency, making it suitable for resource-constrained environments. The network is trained on a labelled rice plant image dataset covering various disease types. By leveraging the power of Inception Networks, the approach captures and learns discriminative features from the input images, enabling accurate disease recognition and detection. The lightweight design of the network ensures efficient inference and facilitates real-time applications. "IoT-Based Pattern Recognition" [33] combines multiple classification algorithms to form an ensemble model that collectively predicts crop disease patterns. The system utilizes IoT devices like sensors and cameras to collect real-time data from the fields. The collected data is then processed and analyzed using pattern recognition techniques to identify and

classify crop diseases. The ensemble model aggregates the outputs of individual classifiers to make accurate and robust predictions. The system enables remote monitoring and timely intervention in crop disease management by leveraging IoT technology. Combining ensemble classification and IoT-based pattern recognition provides a comprehensive and efficient approach to crop disease monitoring, contributing to improved agricultural practices and higher crop yields.

"EfficientNetV2" [9] employs the EfficientNetV2 model, a state-of-the-art deep neural network architecture, for disease detection in cardamom plants. The model is trained on a labelled cardamom leaf image dataset covering various disease types. EfficientNetV2 captures and learns discriminative features from the input images, enabling accurate disease detection. By leveraging the power of deep learning, the approach offers a robust solution for identifying and classifying cardamom plant diseases. The EfficientNetV2 model's architecture balances model complexity and computational efficiency, facilitating real-time disease detection applications. "Soil Sensors-Based Prediction System" [34] involves collecting data from soil sensors, which measure various parameters related to soil health and environmental conditions. The collected data is subjected to exploratory analysis to uncover patterns and relationships between soil factors and plant diseases. Machine learning algorithms are then employed to develop predictive models based on the analyzed data. These models learn from historical data to predict the occurrence and severity of plant diseases. By integrating soil sensor data, exploratory data analysis, and machine learning, the system provides an effective tool for early detection and prevention of plant diseases.

"Northern Maize Leaf Blight Detection" [5] leverages convolutional neural networks (CNNs) to analyze leaf images and accurately identify disease symptoms. The system achieves high detection accuracy by employing a pre-trained CNN model while handling complex field environments effectively. The proposed approach effectively extracts discriminative features from the input images and learns to differentiate between healthy and infected maize leaves. Deep learning enables robust and efficient detection of NMLB, which is crucial for timely intervention and disease management. The model's ability to handle complex field conditions enhances its applicability in real-world scenarios, improving the overall accuracy and reliability of Northern Maize Leaf Blight Detection. "Single Stream Modified MobileNet" [35] is designed as a single stream network, enabling efficient feature extraction and classification. The whale-controlled entropy-based optimization technique further enhances the performance by fine-tuning the model's parameters. By leveraging this framework, accurate recognition of citrus fruit diseases can be achieved. The modified MobileNet V2 architecture effectively captures relevant features from the input images, while the optimization framework ensures optimal parameter adjustments for improved classification accuracy. Combining these techniques results in a robust and efficient system for disease recognition in citrus fruits.

"Deep Neural Network Model" [36] utilizes a convolutional neural network (CNN) to analyze images and accurately identify disease symptoms in citrus fruits and leaves. By training the CNN model on a large dataset of annotated images, the system learns to distinguish between healthy and infected samples with high accuracy. The deep neural network architecture enables extracting meaningful features from the input images, helping reliable disease detection. This approach offers a robust and automated solution for disease identification in citrus fruits and leaves, which is crucial for timely intervention and disease management. Using deep learning techniques enhances the accuracy and efficiency of disease detection, making it a valuable tool for the agricultural industry. "Deep Features Extraction Model" [37] combines transfer learning and the Vision Transformer model to effectively extract discriminative features from plant images. By leveraging a pre-trained model and fine-tuning it on a dataset of labelled plant images, this model achieves high accuracy in classifying different plant diseases. The transfer learning technique enables the model to leverage knowledge from large-scale datasets, while the Vision Transformer architecture captures spatial relationships in the input images. Combining these techniques results in a robust and efficient model for plant disease classification.

"Grape Leaf Esca Disease Detection" [38] utilizes a compressed CNN model, reducing computational complexity while maintaining high detection accuracy. By leveraging this lightweight model, the system achieves real-time detection of Esca disease in grape leaves. The low-cost and low-power design makes it suitable for deployment in resource-constrained environments. The compressed CNN effectively analyzes input images and identifies disease symptoms, providing accurate and timely detection of Esca disease. "Grape Leaf Spot Identification" [39] leverages the

power of GANs to generate synthetic samples that closely resemble the limited available dataset. Training the GAN model on a small number of labelled images, it learns to create real examples of grape leaf spots, enhancing the training data. The fine-grained GAN architecture captures the intricate details of the disease symptoms, enabling accurate identification of grape leaf spots. This approach addresses the challenge of limited samples by augmenting the dataset with synthetic samples, thereby improving the robustness and reliability of the classification model. Using fine-grained GANs in grape leaf spot identification expands the possibilities of disease detection under constrained data conditions, providing valuable insights for agricultural disease management.

"Huanglongbing Detection Method" [40] leverages the power of deep neural networks to analyze leaf images and accurately identify HLB symptoms. The model is trained on a large dataset of annotated images by utilizing transfer learning, enabling it to learn relevant features and patterns associated with HLB. This transfer learning approach enhances the model's ability to detect HLB even with limited training data. The deep neural network architecture effectively extracts discriminative features from the input images, improving the accuracy and reliability of HLB detection. "RiceBioS" [41] utilizes edge computing technology to analyze real-time data collected from rice fields. The system can detect and classify various biotic stress factors affecting rice crops by deploying edge devices with advanced sensors and image processing capabilities. The Edge-as-a-Service model ensures efficient data processing and analysis at the edge, reducing latency and enabling timely decision-making for farmers. The RiceBioS approach empowers farmers with quick and accurate identification of biotic stress, allowing them to take proactive measures to mitigate the damage. By harnessing the power of Edge-as-a-Service, RiceBioS enhances the resilience and productivity of rice farming while supporting sustainable agricultural practices.

"Random Forest (RF)" [42] has become widely adopted for identifying cotton leaf diseases due to its resilient operational framework. This method utilizes a collection of decision trees, each trained on a random segment of the training dataset. During classification, RF amalgamates the predictions from all trees to formulate the final decision. This ensemble method mitigates the influence of individual trees, enhancing the accuracy and dependability of disease identification. RF introduces variability by choosing a subset of input features at each node, bolstering its resilience and capacity for generalization. In the realm of cotton leaf disease identification, RF has demonstrated exceptional performance in precisely categorizing various disease types based on their symptomatic patterns. Its proficiency in managing high-dimensional data, handling missing values, and offering insights into feature significance renders it an invaluable tool for automated disease identification and agricultural decision-making.

"Support Vector Machines (SVM)" [43] have garnered considerable interest in the realm of cotton leaf disease identification owing to their robust operational framework. SVM functions by projecting the input data into a higher-dimensional feature space via a kernel function, aiming to ascertain an optimal hyperplane that effectively segregates various disease classes with maximal margin. SVM achieves superior generalization and adeptly manages intricate relationships between disease symptoms and their respective classes. Additionally, SVM introduces the concept of support vectors, which denote the data points positioned closest to the decision boundary by maximizing the margin. These support vectors play a pivotal role in delineating the decision boundary and significantly contribute to the overall classification accuracy. Within the domain of cotton leaf disease identification, SVM has exhibited noteworthy performance in accurately categorizing diverse disease types based on their symptomatic profiles. Its proficiency in handling high-dimensional data, accommodating nonlinearity, and delineating effective decision boundaries renders it a valuable asset for automated disease identification and agricultural decision-making.

3. CASSOWARY OPTIMIZATION IN CONVOLUTIONAL NEURAL NETWORKS (CO-CNN)

3.1. Convolutional Neural Networks (CNNs)

The techniques used in deep learning, a branch of machine learning, are modelled by how the brain's neural networks are structured and operate. The goal is to train computers to use massive volumes of data for understanding and decision-making. By utilizing several layers of nonlinear transformations, deep learning algorithms endeavour to represent data at high abstraction. Algorithms are trained to do tasks like picture and audio recognition, natural language processing, and decision-making by undergoing these

transformations, which allow them to acquire information at different levels of abstraction.

Regarding computer vision and image recognition, Convolutional Neural Networks (CNNs) are among the most important deep learning architectures. Convolutional neural networks (CNNs) are a subset of DNNs that can learn feature hierarchies from data systematically and dynamically. Their strength is their ability to process incoming data with a grid-like structure, such as photographs. The convolutional layers of a CNN are its brains; these layers use convolutional operations to sift through incoming data, much like filters or kernels. These filters detect features like edges, corners, textures, or other patterns in the input images. Convolution uses a filter to slide over the input data and compute dot products to create feature maps. Convolutional neural networks (CNNs) can learn more complicated information sequentially stack convolutional layers.

CNNs typically include pooling layers, such as max pooling or average pooling, following the convolutional layers. Reduce the spatial dimensionality of the feature maps acquired from the convolutional layers by using pooling layers to downsample them. Because of the reduction in computational complexity and the creation of spatial invariance, the network can recognize features independently of their precise placement in the input.

Downsampling with pooling layers reduces the spatial dimensionality of the feature maps obtained from the convolutional layers. This downsampling allows the network to detect features regardless of their exact location in the input by reducing computational complexity and creating spatial invariance. The steps involved in training a CNN can be summarized as follows:

- **Data Preprocessing:** Prepare the dataset by preprocessing the photographs. Data normalization, resizing, or augmentation to increase diversity may be part of this process.
- **Architecture Design:** Specify the CNN's design, including its size, the amount of convolutional and pooling layers, and the fully connected and filtering layers.
- **Forward Propagation:** Pass the input images through the network, applying convolution, activation functions (e.g., ReLU), and pooling operations to generate feature maps.
- **Loss Computation:** Using an appropriate loss function, like categorical cross-entropy for classification problems, determine the discrepancy between the anticipated output and the ground truth labels.
- **Backpropagation:** Using optimization methods such as stochastic gradient descent (SGD) or its variations, propagate the mistake backwards through the network to update the weights and biases.
- **Parameter Optimization:** Optimize the model's prediction accuracy by adjusting the network's parameters to minimize the loss function.
- **Evaluation:** It is essential to test the trained model's generalizability to new data by running it on a different validation dataset.
- **Fine-tuning:** If desired, tweak the model's hyperparameters or use methods like transfer learning to use pre-trained models already trained for comparable tasks.

By following these steps, CNNs can be trained effectively to recognize patterns and make predictions in various domains, ranging from image classification to object detection and segmentation.

3.2. Cassowary Optimization (CO)

Cassowary Optimization (CO) is a heuristic optimization algorithm inspired by the foraging behavior of the cassowary bird, native to tropical forests. It mimics the bird's efficient search strategy for finding food in complex environments. CO operates by iteratively adjusting the parameters of a solution space based on the evaluation of an objective function. Unlike traditional optimization methods, CO dynamically balances exploration and exploitation, allowing it to efficiently navigate diverse solution landscapes. By iteratively updating solutions while considering constraints, CO converges towards optimal or near-optimal solutions. This approach makes CO suitable for a wide range of optimization problems, particularly those characterized by non-linearity, multimodality, and complex constraints. CO's ability to adapt to changing environments and efficiently explore solution spaces makes it a promising optimization technique in various domains, including engineering, finance, and biology.

A) Dynamic Exploration:

In the initial step of CO, establishing a framework for dynamically exploring the search space to locate promising regions that might

efficiently contain optimal solutions. Dynamic Exploration aims to balance exploitation and Exploration, enabling the algorithm to navigate the solution space effectively while avoiding premature convergence to local optima. The algorithm dynamically adjusts its exploration strategy based on the evolving characteristics of the search space. This dynamic adaptation helps efficiently explore local and global regions, enhancing the algorithm's ability to discover diverse solutions. The exploration process is guided by the inherent properties of the optimization problem and the information gathered during the search.

The Dynamic Exploration phase involves formulating adaptive exploration mechanisms that regulate the exploration-exploitation trade-off. One such mechanism could be the incorporation of a dynamic exploration parameterα, which controls the degree of Exploration in the search space. This parameter can be adjusted during optimization based on the algorithm's performance and convergence behaviour.

Integrating adaptive strategies within the search algorithm is a common approach to implementing dynamic Exploration. For instance, a dynamic adjustment of the step size δ in the search direction can facilitate the Exploration of diverse regions. The step size adaptation can be governed by a function that responds to the local landscape of the objective function, represented mathematically in Eq.(1).

$$\delta_k = \frac{\delta_{max}}{1 + exp\left(-\beta\left(f(x_k) - f(x_{best})\right)\right)} \quad (1)$$

where δ_k represents the adjusted step size at iteration k, δ_{max} is the maximum step size, $f(x_k)$ denotes the objective function value at iteration k, $f(x_{best})$ It is the best objective function value encountered so far, and β controls the rate of adaptation.

Incorporating adaptive mutation mechanisms can further enhance dynamic Exploration. For instance, a mutation parameter μ can be adaptively adjusted to control the diversity of solutions generated during the search process. The mutation operator can be mathematically represented in Eq.(2).

$$x_i' = x_i + \mu.(x_{best} - x_i) \quad (2)$$

where x_i' represents the mutated solution, x_i is the current solution, and x_{best} Denotes the best solution found so far.

Leveraging dynamic population management strategies can contribute to compelling Exploration. By dynamically adjusting the population size based on the convergence status and the diversity of solutions, the algorithm can allocate computational resources more efficiently towards Exploration or exploitation. A dynamic population size N can be determined using a mechanism that balances the exploration-exploitation trade-off, is shown in Eq.(3).

$$N_k = roud\left(N_{max} - \frac{k}{\max\_iterations}.(N_{max} - N_{min})\right) \quad (3)$$

where N_k represents the population size at iteration k, N_{max} and N_{min} Denote the maximum and minimum population sizes, respectively, and $max\_iterations$ is the maximum number of iterations.

B) Adaptive Behavior:

Adaptive Behavior aims to imbue the algorithm with the capability to adjust its strategies and parameters in real time based on the feedback obtained during the optimization process. This adaptability enables CO to navigate complex and dynamic search spaces effectively, improving convergence and solution quality.

The algorithm dynamically adjusts its Behavior and parameters to optimize its performance and adapt to the evolving characteristics of the problem. Adaptive Behavior encompasses various aspects, including parameter adaptation, strategy adjustment, and response to environmental changes.

Adaptive Behavior entails formulating adaptive mechanisms that govern the algorithm's behaviour and parameter settings. One such mechanism involves adaptive step size adjustment, where the step size δ is dynamically updated based on the progress of the optimization process. This adaptive step size represented mathematically in Eq.(4).

$$\delta_k = \delta_{max}.exp\left(-\frac{k}{\tau}\right) \quad (4)$$

where δ_k represents the adjusted step size at iteration k, δ_{max} Is the maximum step size, τ is a parameter controlling the rate of adaptation.

Incorporating adaptive mutation mechanisms can enhance the algorithm's adaptive Behavior. The mutation rate μ can be adaptively adjusted to balance Exploration and exploitation. An adaptive mutation rate is shown in Eq.(5).

$$\mu_k = \mu_{max}.exp\left(-\frac{k}{\tau'}\right) \quad (5)$$

where τ' is the initial state, μ_k is the rate of mutation at iteration k, μ_{max} is the mutation rate at its maximum and Manages the pace at which mutations are adapted.

Adaptive Behavior involves the dynamic adjustment of strategies employed by the algorithm. For instance, the selection of operators such as crossover and mutation can be dynamically determined based on the performance and convergence status of the algorithm. This adaptive strategy selection can be represented mathematically in Eq.(6).

$$Operator_k = \begin{cases} Crossover, & if\ condition_1 \\ Mutation, & if\ condition_2 \end{cases} \quad (6)$$

where $Operator_k$represents the selected operator at iteration k, and $condition_1$and $condition_2$ are adaptive conditions based on the algorithm's Behavior.

Persistence and Resilience

Persistence and Resilience involve formulating mechanisms that promote continuous progress and robustness in the optimization process. One such mechanism is incorporating persistence factors that encourage the algorithm to explore promising search space regions. This can be achieved through the integration of a persistence factor γ into the objective function, guiding the algorithm towards areas with potentially higher fitness values.

$$f_{persist}(x) = f(x) + \gamma.penalty(x) \quad (7)$$

where in Eq.(7), $f_{persist}(x)$ represents the modified objective function with the persistence factor, and $penalty(x)$ penalizes solutions that deviate from promising regions.

Resilience mechanisms can be incorporated to enable the algorithm to recover from disruptions and maintain robust performance. One approach is to introduce adaptive damping factors that regulate the impact of disturbances on the optimization process. The damping factor δ can be adaptively adjusted based on the magnitude of disturbances encountered during the search is represented mathematically in Eq.(8).

$$\delta_k = \delta_{max}.exp\left(-\frac{disturbance_k}{\tau''}\right) \quad (8)$$

where δ_k denotes the adjusted damping factor at iteration k, δ_{max} is the maximum damping factor, $disturbance_k$ quantifies the magnitude of a disturbance at iteration k, and τ''controls the rate of adaptation.

D) Multi-Objective Optimization

It extends the optimization framework to scenarios where multiple conflicting objectives must be simultaneously optimized. This step aims to equip the algorithm with the capability to handle such multi-objective optimization problems effectively, enabling it to discover trade-off solutions that represent a balance between competing objectives. The algorithm is extended to handle multiple objective functions simultaneously. Improving one target could result in the deterioration of others; this is a common occurrence in objective functions. Finding a collection of solutions that constitute the optimal trade-off, where no objective can be enhanced without deteriorating another objective, is referred to as the Pareto frontier.

Multi-Objective Optimization involves formulating mechanisms for handling multiple objective functions and identifying Pareto-optimal solutions. A typical strategy uses scalarization techniques to reduce the optimization issue from several objectives to a single target. The weighted sum technique is one such strategy; it uses weighted coefficients to combine many objectives into one is shown in Eq.(9).

$$f(x) = \sum_{i=1}^{m} w_i.f_i(x) \quad (9)$$

where $f(x)$ represents the combined objective function, $f_i(x)$ denotes the individual objective functions and w_i are the weights assigned to each objective. The weights are typically adjusted to explore different regions of the Pareto frontier.

The algorithm can employ mechanisms for maintaining diversity and coverage of solutions along the Pareto frontier. This may involve integrating diversity-preserving mechanisms, such as crowding distance or niche formation, to ensure the algorithm explores a wide range of trade-off solutions.

The mechanisms for selecting solutions from the Pareto frontier can be incorporated based

on preferences or decision-making criteria. One common approach is to use dominance-based selection methods, such as the non-dominated sorting genetic algorithm (NSGA-II), which categorizes solutions into Pareto fronts based on dominance relationships and selects solutions from diverse fronts to maintain diversity along the Pareto frontier.

E) Communication and Collaboration

Communication and Collaboration aim to harness the collective intelligence of the algorithm by facilitating the exchange of information, sharing of knowledge, and coordinating efforts among individuals. The algorithm is extended to incorporate mechanisms for individuals to communicate, collaborate, and coordinate their actions. This collaborative approach enables the algorithm to leverage synergy and cooperation among individuals, enhancing Exploration, exploitation, and optimization performance.

Communication and Collaboration involve formulating mechanisms for individuals to exchange information and coordinate their actions. One approach is to incorporate communication channels through which individuals can share knowledge, experiences, and solutions. This may involve the formulation of communication functions that govern the exchange of information among individuals:

$$comm_{ij} = exp\left(-\frac{d_{ij}}{\sigma^2}\right) \quad (10)$$

where in Eq.(10), $comm_{ij}$ represents the communication strength between individuals i and j, d_{ij} denotes the distance between individuals and σ^2 controls the rate of communication decay with distance.

Collaboration can be facilitated through mechanisms for individuals to collaborate on tasks or share computational resources. For instance, individuals can form collaborative groups to tackle specific sub-problems or share computational workload collectively is shown in Eq.(11).

$$group\_task_{ij} = \begin{cases} 1, & if\ individuals\ i\ and\ j\ collaborate\ on \\ 0, & otherwise \end{cases} \quad (11)$$

where $group\_task_{ij}$ denotes the assignment of collaborative tasks between individuals i and j. The mechanisms for coordinating actions among individuals can be integrated to ensure coherence and alignment of efforts towards common objectives. This may involve the formulation of coordination strategies that synchronize the actions of individuals based on shared goals or objectives is represented mathematically in Eq.(12).

$$coordination_{ij} = cos(\theta_{ij}) \quad (12)$$

where $coordination_{ij}$ represents the coordination strength between individuals i and j, and θ_{ij} denotes the angle between their respective action vectors.

F) Local Search Refinement

Local Search Refinement aims to enhance the exploration and exploitation capabilities of the algorithm by incorporating mechanisms for fine-tuning solutions in the vicinity of promising regions identified during the optimization process. The algorithm employs local search techniques to refine solutions and exploit local information within the search space. Local search refinement enables the algorithm to zoom in on promising regions and improve the quality of solutions by iteratively exploring the neighbourhood of candidate solutions.

Local Search Refinement involves formulating mechanisms for performing local search operations around promising solutions. One approach is to employ gradient-based optimization methods, such as gradient descent or Newton's method, to update solutions towards local optima iteratively. The update rule for gradient descent can be expressed mathematically in Eq.(13).

$$x_{k+1} = x_k - \eta \nabla f(x_k) \quad (13)$$

where x_k represents the current solution, η denotes the step size, and $\nabla f(x_k)$ denotes the gradient of the objective function concerning x_k.

Local Search Refinement can incorporate mechanisms for exploring the neighbourhood of solutions using local exploration operators. One such operator is the mutation operator, which introduces small perturbations to solutions to examine nearby regions of the search space.

$$x' = x + \epsilon \quad (14)$$

where in Eq.(14), x' represents the perturbed solution, x denotes the original solution, and ϵ represents a small perturbation vector.

The mechanisms for local exploitation of promising regions can be integrated to effectively exploit information gathered during optimization. This may involve the formulation of exploitation

strategies that prioritize Exploration in areas with high potential for improvement is shown in Eq.(15).

$$x_{new} = x_{current} + \alpha . explore(x_{current}) \quad (15)$$

where x_{new} represents the refined solution, $x_{current}$ denotes the current solution, α represents the exploration factor, and $explore(x_{current})$ represents the direction of Exploration based on local information.

G) Constraint Handling:

Constraint Handling is crucial for addressing optimization problems with constraints, ensuring that solutions generated by the algorithm satisfy both the objective function requirements and the imposed restrictions. The algorithm is extended to effectively incorporate mechanisms for handling constraints. These mechanisms aim to guide the search process towards feasible regions of the solution space while maintaining the Exploration of the objective function landscape.

Constraint Handling involves formulating mechanisms for enforcing constraints and guiding the search towards feasible solutions. One common approach is to penalize infeasible solutions by modifying the objective function to include penalty terms. The modified objective function can be expressed in Eq.(16).

$$f_{penalized}(x) = f(x) + \sum_{i=1}^{n} \lambda_i . g_i(x)^2 \quad (16)$$

where $f_{penalized}(x)$ represents the penalized objective function, $f(x)$ denotes the original objective function, λ_i represents penalty coefficients, $g_i(x)$ denotes the i-th constraint function, and n is the total number of constraints.

Mechanisms for handling inequality constraints can be incorporated using barrier or penalty methods. Barrier methods introduce barriers around infeasible regions to prevent the algorithm from exploring them, while penalty methods impose penalties on infeasible solutions to discourage their selection. The barrier function can be expressed in Eq.(17).

$$B(x) = -\sum_{i=1}^{n} \frac{1}{g_i(x)} \quad (17)$$

where $B(x)$ represents the barrier function and $g_i(x)$ denotes the i-th constraint function. The Mechanisms for handling equality constraints can be integrated using Lagrange multipliers or penalty methods. Lagrange multipliers introduce additional variables to enforce equality constraints, while penalty methods penalize violations of equality constraints in the objective function. The penalty function for equality constraints can be expressed in Eq.(18).

$$P(x) = \sum_{i=1}^{m} \lambda_i . |h_i(x)| \quad (18)$$

where $P(x)$ represents the penalty function for equality constraints, λ_i denotes penalty coefficients, $h_i(x)$ denotes the i-th equality constraint function, and m is the total number of equality constraints.

H) Diversity Preservation:

Diversity Preservation aims to maintain a diverse population of solutions throughout the optimization process, ensuring the algorithm explores many promising regions in the solution space. The algorithm incorporates mechanisms for preserving diversity among individuals, preventing premature convergence to suboptimal solutions, and promoting solution space exploration. It also involves formulating mechanisms for maintaining diversity within the population of solutions. One approach is to incorporate diversity measures that quantify the population's dissimilarity or spread of solutions. One commonly used diversity measure is the crowding distance, which measures the average distance of a solution to its nearest neighbours is represented mathematically in Eq.(19).

$$Crowding\_Distance(x_i) = \frac{1}{k}\sum_{j=1}^{k} \|x_i - x_j\| \quad (19)$$

where $Crowding\_Distance(x_i)$ represents the crowding distance of the solution x_i, x_j denotes the nearest neighbour of x_i, and k is the number of nearest neighbours considered.

This mechanisms for promoting diversity can be incorporated into selection and reproduction operators to ensure that individuals from diverse regions of the solution space are retained and propagated. One approach is to introduce diversity-based selection mechanisms that prioritize individuals with low crowding distances or high dissimilarity is shown in Eq.(20).

$$Selection\_Probability(x_i) = \frac{Crowding\_Distance(x_i)}{\sum_{i=1}^{N} Crowding\_Distance(x_i)} \quad (20)$$

where $Selection\_Probability(x_i)$ represents the probability of selecting a solution x_i for

reproduction, and N is the total number of solutions in the population.

Mechanisms for promoting niche formation and speciation can be integrated to encourage the emergence of diverse subpopulations within the population. This can be achieved through the introduction of niche radius parameters and mechanisms for promoting competition and cooperation among individuals within niches is mathematically represented in Eq.(21).

$$Niche_{Radius(x_i)} = \frac{1}{Crowding_{Distance(x_i)}} \quad (21)$$

where $Niche\_Radius(x_i)$ represents the niche radius of the solution x_i, and $Crowding\_Distance(x_i)$ is used to determine the size of the niche.

I) Termination Criteria

Termination Criteria are the guidelines for determining when to halt the optimization process, ensuring that computational resources are utilized efficiently and effectively. The algorithm incorporates mechanisms for evaluating convergence, assessing solution quality, and determining when to stop the optimization process based on predefined criteria. Termination Criteria involve formulating conditions for assessing convergence and solution quality. One commonly used criterion is to monitor the convergence of the objective function values over successive iterations. A convergence criterion based on the change in the best objective function value (f_{best}) over iterations can be expressed in Eq.(22).

$$\frac{\left|f_{best}^{(k)} - f_{best}^{(k-1)}\right|}{f_{best}^{(k)}} \leq \epsilon \quad (22)$$

where $f_{best}^{(k)}$ represents the best objective function value at iteration k, and ϵ denotes a small tolerance threshold.

Termination Criteria can incorporate mechanisms for assessing solution diversity and coverage of the solution space. One approach is to monitor the spread of solutions in the population using diversity measures such as the average crowding distance or the standard deviation of solution distances is represented mathematically in Eq.(23).

$$Diversity\_Measure \leq \delta \quad (23)$$

where $Diversity\_Measure$ represents a measure of solution diversity, and δ denotes a predefined threshold. The Mechanisms for assessing computational resources and budget can be integrated into Termination Criteria to ensure that the optimization process does not exceed predefined limits. This may involve monitoring the number of iterations, function evaluations, or computational time is mathematically represented in Eq.(24).

$$Resource\_Usage \leq Budget \quad (24)$$

where $Resource\_Usage$ represents the cumulative resource usage during the optimization process, and $Budget$ denotes the predefined resource budget.

3.3. CO-CNN (Cassowary Optimization for Convolutional Neural Networks)

It is a novel approach that leverages Cassowary Optimization (CO) to optimize Convolutional Neural Networks (CNNs) effectively. CO-CNN aims to enhance the performance of CNNs by dynamically adjusting network parameters and architecture based on the evolving optimization landscape. Here are the steps to optimize CNN with Cassowary Optimization.

- **Problem Formulation:** Define the optimization problem for CO-CNN, specifying the objective function to be optimized, which typically involves minimizing the classification error or maximizing accuracy on a given dataset.
- **Representation of Solutions:** Represent CNN architectures and parameters as solutions in the optimization space. This includes defining the network topology, such as the number of layers, filter sizes, and activation functions, as well as the parameters of each layer.
- **Initialization:** Initialize a population of CNN architectures and parameters using random or heuristic methods. Each solution represents a potential CNN configuration to be optimized.
- **Evaluation:** Evaluate the performance of each CNN configuration in the population using the specified objective function, typically through training and validation on a subset of the dataset.
- **Optimization Process:** Apply Cassowary Optimization to update the population of CNN configurations iteratively. During each iteration, solutions are refined based on their performance and the optimization landscape.

- **Dynamic Exploration:** Incorporate dynamic exploration mechanisms to effectively explore diverse regions of the solution space. This involves adjusting parameters such as learning rates, filter sizes, and network architectures to promote Exploration while avoiding premature convergence.
- **Adaptive Behavior:** Integrate adaptive strategies to dynamically adjust CNN parameters and architectures based on the optimization progress. This includes adapting learning rates, activation functions, and dropout rates to optimize convergence and solution quality.
- **Local Search Refinement:** Implement techniques to fine-tune CNN configurations and exploit local information in the optimization landscape. This may involve gradient-based optimization methods or neighbourhood search algorithms to improve solution quality.
- **Constraint Handling:** Address constraints inherent in CNN optimization, such as computational resource limitations or architectural constraints. Ensure that CNN configurations satisfy constraints while optimizing performance using penalty methods or constraint satisfaction techniques.
- **Termination Criteria:** Define termination criteria to stop the optimization process effectively. Termination may occur based on convergence of the objective function, resource utilization constraints, or predefined budget limits.

CO-CNN optimizes CNN architectures and parameters effectively, improving performance and generalization on various tasks, including image classification, object detection, and image segmentation. CO-CNN's adaptive and dynamic nature allows it to navigate the complex optimization landscape of CNNs efficiently, resulting in superior performance compared to traditional optimization methods.

A) Problem Formulation:

Formulating the optimization problem and defining the objective function to be optimized. Inspired by Cassowary Optimization, CO-CNN aims to maximize the performance of CNN by minimizing classification error or maximizing accuracy on a given dataset. The objective function is typically defined based on the discrepancy between predicted and actual class labels for input samples.

The objective function $Obj(W)$ can be formulated as the average loss over a dataset D of N samples is shown in Eq.(25).

$$Obj(W) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i) \quad (25)$$

where W represents the parameters of the CNN model, y_i denotes the actual class label for the i-th sample, $\hat{y}_i$ denotes the predicted class label, and $L(y_i, \hat{y}_i)$ the loss function quantifies the discrepancy between the actual and predicted labels. The optimization problem can be formulated as a minimization problem, where the goal is to find the optimal set of parameters. W^* that minimizes the objective function is mathematically represented in Eq.(26).

$$W^* = \underset{W}{\operatorname{argmin}}\, Obj(W) \quad (26)$$

The objective function captures the performance of the CNN model in terms of its ability to classify input samples accurately. By minimizing the objective function, CO-CNN aims to optimize the parameters of the CNN model to achieve higher classification accuracy and better generalization on unseen data. The problem formulation may also incorporate regularization terms to prevent overfitting and encourage smoother solutions. Regularization can be achieved by adding a regularization term to the objective function, penalizing large parameter values:

$$Obj_{reg}(W) = Obj(W) + \lambda R(W) \quad (27)$$

where in Eq.(27), λ is the regularization parameter, and $R(W)$ is the regularization term that penalizes large parameter values.

B) Representation of Solutions

In CNNs, solutions refer to different network architectures and configurations, including the number of layers, filter sizes, activation functions, and other architectural parameters. A solution S in CO-CNN can be represented as a vector of parameters $p = [p_1, p_2, \ldots\ldots, p_n]$, where each parameter p_i corresponds to a specific architectural aspect of the CNN model. The vector p encapsulates the entire configuration of the CNN model, enabling the optimization process to explore different architectural possibilities are mathematically represented in Eq.(28).

$$[p_1, p_2, \ldots\ldots, p_n] \quad (28)$$

The representation of solutions can be augmented with encoding schemes to handle categorical and discrete parameters. For instance, the number of layers in the CNN architecture may be encoded as a categorical variable, where each category corresponds to a different number of layers is shown in Eq.(29).

$$Number\ of\ Layers = \{1,2,3, \ldots\ldots, L_{max}\} \quad (29)$$

Other architectural parameters, such as filter sizes and activation functions, can be encoded using appropriate schemes to facilitate the optimization process. The representation of solutions can include mechanisms for specifying parameter ranges and constraints. For example, the range of filter sizes may be constrained to a predefined interval is shown in Eq.(30).

$$Filter\ Size\ \in [F_{min}, F_{max}] \quad (30)$$

where F_{min} and F_{max} denote the minimum and maximum allowable filter sizes, respectively. Solutions in CO-CNN may incorporate parameter sharing and transfer learning mechanisms, allowing the optimization process to leverage pre-trained models or shared parameters across different tasks or domains.

C) Initialization

This phase involves initializing a population of CNN architectures and parameters. The initialization process aims to create an initial set of solutions that will undergo optimization through the Cassowary Optimization algorithm. The initialization of solutions can be represented as follows. Let $P = [p_1, p_2, \ldots\ldots, p_N]$ denote the population of solutions, where each p_i represents an individual solution vector. Initialization involves generating random or predefined values for each parameter within the specified ranges and constraints is represented in Eq.(31).

$$P = [p_1, p_2, \ldots\ldots, p_N] \quad (31)$$

The initialization process can incorporate diversity-promoting mechanisms to ensure a diverse population of solutions. This may involve introducing randomness or heuristic strategies to generate solutions that explore different regions of the solution space is mathematically represented in Eq.(32).

$$Diversity\_Measure = \frac{1}{N}\sum_{i=1}^{N} Distance(p_i, Centroid) \quad (32)$$

where $Distance(p_i, Centroid)$ represents the distance between the solution p_i and the centroid of the population. The initialization process may include mechanisms for incorporating prior knowledge or domain-specific information into the population generation process. This can be achieved by biasing the initialization towards effective configurations in similar tasks or domains is shown in Eq.(33).

$$p_i = p_i + Domain\_Knowledge \quad (33)$$

The initial solutions in CO-CNN may involve strategies for warm-starting the optimization process. This can be done by initializing a portion of the population with solutions obtained from previous optimization runs or pre-trained models is represented in Eq.(34).

$$P = [p_1, p_2, \ldots, p_N] + Previous\_Solutions \quad (34)$$

This initialization scheme allows CO-CNN to explore various architectural possibilities and configurations effectively, improving performance on CNN-related tasks.

D) Evaluation

The evaluation of a CNN configuration p_i can be represented by the objective function $Obj(p_i)$, quantifying the discrepancy between predicted and actual class labels for input samples.

$$Obj(p_i) = \frac{1}{N}\sum_{j=1}^{N} L(y_j, \hat{y}_j) \quad (35)$$

where in Eq.(35), N represents the number of samples in the dataset, y_j denotes the actual class label for j-th sample, and $\hat{y}_j$ denotes the predicted class label. $L(y_j, \hat{y}_j)$ the loss function quantifies the discrepancy between the actual and predicted labels.

The evaluation process may include mechanisms for assessing additional performance metrics such as accuracy, precision, recall, or F1-score, depending on the specific task and objectives is mathematically represented in Eq.(36).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (36)$$

where TP denotes true positives, TN denotes true negatives, FP denotes false positives, and FN denotes false negatives. The evaluation of CNN configurations can incorporate mechanisms for handling overfitting and generalization. This may involve cross-validation, regularization, or early stopping to prevent overfitting on the training data and promote generalization to unseen data are mathematically represented in Eq.(37).

$$Obj_{reg}(p_i) = Obj(p_i) + \lambda R(p_i) \quad (37)$$

The regularisation term that penalizes high parameter values is denoted by $\lambda R(p_i)$, and the regularisation parameter is denoted by λ. The evaluation process in CO-CNN may include mechanisms for computational efficiency, such as mini-batch training or parallel processing, to expedite the evaluation of multiple CNN configurations simultaneously.

E) Optimization Process:

The optimization process in CO-CNN can be represented using iterative update equations that adjust the parameters of each CNN configuration. p_i based on their performance evaluated by the objective function $Obj(p_i)$ is mathematically represented in Eq.(38)

$$p_i^{(t+1)} = p_i^{(t)} + \Delta p_i^{(t)} \quad (38)$$

where $p_i^{(t)}$ represents the parameters of CNN configuration p_i at iteration t, and $\Delta p_i^{(t)}$ denotes the update applied to the parameters in the current iteration. The optimization process may incorporate adaptive mechanisms to dynamically adjust the step sizes or learning rates of the updates based on the convergence behaviour and progress of the optimization process is represented in Eq.(39).

$$\Delta p_i^{(t)} = \alpha . \nabla Obj\left(p_i^{(t)}\right) \quad (39)$$

where α represents the learning rate and $\nabla Obj\left(p_i^{(t)}\right)$ denotes the gradient of the objective function concerning the parameters of the CNN configuration $p_i^{(t)}$. The optimization process may also include mechanisms for promoting diversity among solutions to prevent premature convergence and encourage solution space exploration. This can be achieved by introducing diversity-promoting terms in the update equations that promote solutions to move away from each other.

$$\Delta p_i^{(t)} = \beta . Diversity\_Measure \quad (40)$$

where in Eq.(40), β represents a diversity-promoting factor, and $Diversity\_Measure$ quantifies the diversity among solutions in the population.

The optimization process in CO-CNN may incorporate mechanisms for handling constraints and enforcing architectural constraints during the parameter updates. This ensures the updated CNN configurations remain valid and feasible throughout the optimization process is represented in Eq.(41).

$$p_i^{(t+1)} = Projection\left(p_i^{(t)} + \Delta p_i^{(t)}\right) \quad (41)$$

where $Projection(\cdot)$ represents a projection operator that enforces constraints on the updated parameters.

Dynamic Exploration:

The Dynamic Exploration can be achieved by introducing adaptive mechanisms that adjust the exploration parameters or strategies based on the convergence behaviour and progress of the optimization process. One approach is to dynamically change the learning rates or step sizes of the parameter updates to control the magnitude of Exploration is represented in Eq.(42).

$$\alpha^{(t+1)} = \alpha^{(t)} \times exp(-\gamma . t) \quad (42)$$

where $\alpha^{(t)}$ represents the learning rate at iteration t, and γ is a decay factor that controls the rate of decrease of the learning rate over iterations. This Dynamic Exploration can involve mechanisms for balancing Exploration and exploitation to effectively navigate the trade-off between exploring new regions of the solution space and exploiting promising areas. This can be achieved by introducing adaptive mechanisms that prioritize Exploration during early iterations and gradually shift towards exploitation as the optimization progresses are shown in Eq.(43).

$$Exploration\_Probability^{(t)} = min(1, \beta . t) \quad (43)$$

where $Exploration\_Probability^{(t)}$ represents the probability of selecting exploration strategies at iteration t, and β is a scaling factor that controls the rate of increase of the exploration probability over iterations.

Dynamic Exploration in CO-CNN can incorporate mechanisms for adjusting the diversity-promoting strategies based on the convergence behaviour of the optimization process. This may involve dynamically adjusting the diversity-promoting factor to encourage Exploration when convergence slows down and exploitation when convergence accelerates.

$$\beta^{(t+1)} = \beta^{(t)} + \delta \quad (44)$$

where in Eq.(44)m $\beta^{(t)}$ represents the diversity-promoting factor at iteration t, and δ is a parameter that controls the rate of change of the diversity-promoting factor over iterations.

Dynamic Exploration may involve mechanisms for adapting the exploration strategies based on the characteristics of the optimization landscape. This can be achieved by incorporating adaptive mechanisms that dynamically adjust the exploration parameters based on the curvature and gradient information of the objective function are mathematically represented in Eq.(45).

$$Exploration\_Parameter^{(t+1)} = Exploration\_Parameter^{(t)} - \eta.\nabla Obj\left(p^{(t)}\right) \quad \text{Eq.(45)}$$

where $Exploration\_Parameter^{(t)}$ represents the exploration parameter at iteration t, and η is a learning rate parameter that controls the rate of adjustment of the exploration parameter based on the gradient of the objective function.

Adaptive Behavior:

Adaptive Behavior in CO-CNN can be achieved through mechanisms that dynamically adjust various parameters and strategies based on the optimization progress. One approach is to dynamically change the parameter updates' learning rates or step sizes based on the convergence behaviour.

$$\alpha^{(t+1)} = \alpha^{(t)} \times exp(-\gamma.t) \quad (46)|$$

where $\alpha^{(t)}$ represents the learning rate at iteration t, and γ is a decay factor that controls the rate of decrease of the learning rate over iterations. The Adaptive Behavior may involve mechanisms for adjusting architectural parameters such as the number of layers, filter sizes, and activation functions based on the convergence behaviour and performance of the CNN configurations are represented mathematically in Eq.(47).

$$Num\_Layers^{(t+1)} = Num\_Layers^{(t)} + \delta \quad (47)$$

where $Num\_Layers^{(t)}$ represents the number of layers at iteration t, and δ is a parameter that controls the rate of change of the number of layers over iterations. Adaptive Behavior in CO-CNN can incorporate mechanisms for dynamically adjusting regularization parameters to prevent overfitting and promote generalization. This may involve adaptive mechanisms that dynamically change the regularization strength based on the convergence behaviour and performance of the CNN configurations.

$$\lambda^{(t+1)} = \lambda^{(t)} + \epsilon \quad (48)$$

where in Eq.(48), $\lambda^{(t)}$ represents the regularization parameter at iteration t, and ϵ is a parameter that controls the rate of change of the regularization parameter over iterations. Adaptive Behavior may involve mechanisms for dynamically adjusting dropout rates and other regularization techniques based on the convergence behaviour and performance of the CNN configurations is shown in Eq.(49).

$$Dropout\_Rate^{(t+1)} = Dropout\_Rate^{(t)} + \zeta \quad (49)$$

Where $Dropout\_Rate^{(t)}$ represents the dropout rate at iteration t, and ζ is a parameter that controls the rate of change of the dropout rate over iterations.

Local Search Refinement:

Local search refinement phase aims to further improve the quality of solutions by exploring the regional neighbourhood of promising solutions. Inspired by cassowary optimization, CO-CNN incorporates local search refinement to refine CNN architectures and parameters to achieve superior performance iteratively. Local search refinement in CO-CNN can be represented using iterative update equations that explore the regional neighbourhood of each CNN configuration. p_i based on their performance evaluated by the objective function $Obj(p_i)$ is mathematically represented in Eq.(50).

$$p_i^{(t+1)} = p_i^{(t)} + \Delta p_i^{(t)} \quad (50)$$

where $p_i^{(t)}$ represents the parameters of CNN configuration p_i at iteration t, and $\Delta p_i^{(t)}$ denotes the update applied to the parameters in the current iteration.

Local search refinement may incorporate mechanisms for adjusting the step sizes or learning

rates of the updates to control the magnitude of Exploration within the local neighbourhood.

$$\alpha^{(t+1)} = \alpha^{(t)} \times exp(-\gamma.t) \quad (51)$$

where in Eq.(51), $\alpha^{(t)}$ represents the learning rate at iteration t, and γ is a decay factor that controls the rate of decrease of the learning rate over iterations. This local search refinement in CO-CNN can incorporate mechanisms for promoting diversity among solutions within the local neighbourhood. This may involve introducing randomness or heuristic strategies to explore diverse regions of the regional solution space is mathematically represented in Eq.(52).

$$Diversity\_Measure^{(t)} = \frac{1}{N}\sum_{i=1}^{N} Distance\left(p_i^{(t)}, Centroid^{(t)}\right) \quad (52)$$

where $Distance\left(p_i^{(t)}, Centroid^{(t)}\right)$ represents the distance between the solution $p_i^{(t)}$ and the centroid of the local neighbourhood, and N is the number of solutions in the local neighbourhood.

This phase y involve mechanisms for handling constraints and enforcing architectural constraints during the parameter updates within the local neighbourhood is represented in Eq.(53).

$$p_i^{(t+1)} = Projection\left(p_i^{(t)} + \Delta p_i^{(t)}\right) \quad (53)$$

where $Projection(.)$ represents a projection operator that enforces constraints on the updated parameters within the local neighbourhood.

J) Termination Criteria

Defining termination criteria to determine when to stop the optimization process. Termination criteria ensure that the optimization process halts once certain conditions are met, such as achieving a satisfactory level of performance or reaching a predefined number of iterations. CO-CNN incorporates termination criteria to manage computational resources and prevent overfitting efficiently.

Termination criteria in CO-CNN can be represented by conditions evaluated at each iteration of the optimization process. One common termination criterion is to stop the optimization process once a maximum number of iterations $MaxIterations$ is reached.

$$t \geq MaxIterations \quad (54)$$

where in Eq.(54), t denotes the current iteration number.

It may include conditions based on the convergence behaviour of the optimization process. For example, the optimization process can be stopped once the improvement in the objective function value falls below a certain threshold ϵ represented in Eq.(55).

$$\left|Obj\left(p^{(t+1)}\right) - Obj\left(p^{(t)}\right)\right| \leq \epsilon \quad (55)$$

where $Obj\left(p^{(t)}\right)$ represents the objective function value at iteration t.

This phase may incorporate mechanisms for monitoring the performance of the CNN configurations and stopping the optimization process once a satisfactory level of performance is achieved represented in Eq.(56).

$$Performance^{(t)} \geq Threshold \quad (56)$$

where $Performance^{(t)}$ represents a performance metric (e.g., accuracy, loss) at iteration t, and $Threshold$ is a predefined threshold value.

Termination criteria may involve mechanisms for controlling the computational resources allocated to the optimization process, such as stopping the optimization process once a specified amount of time or computational budget is exhausted.

$$Elapsed\ Time \geq MaxTime \quad (57)$$

where in Eq.(57), $Elapsed\ Time$ represents the time elapsed since the start of the optimization process, and $MaxTime$ is the maximum allowable time.

3.4. Identification and Prediction using CO-CNN

Cotton is one of the most economically important crops globally, serving as a primary source of fibre for the textile industry. The Cotton cultivation faces significant challenges due to various diseases that severely impact yield and quality. The Cotton plants were infected by Bacteria, Fungus and Viruses. Bacterial diseases pose significant threats to cotton plants, impacting yield and quality. These diseases pose significant threats to crop health and yield. Angular leaf spot caused by Xanthomonas campestris pv. malvacearum manifests as water-soaked lesions on leaves, leading to defoliation. Bacterial wilt, caused by Ralstonia solanacearum, induces wilting and

eventual death of infected plants, challenging cotton cultivation endeavors. The bacterial infection in cotton plants is depicted in Fig 1.

Fig 1. Bacterial Infection

Fungal diseases pose significant threats to cotton plants. One prevalent fungal disease is Fusarium wilt, caused by Fusarium oxysporum. This pathogen infiltrates the plant's vascular system, resulting in wilting and death. Alternaria leaf spot, caused by Alternaria alternata, manifests as dark lesions on leaves, impacting photosynthesis. Rhizoctonia solani causes root rot, hindering nutrient uptake and stunting growth. These fungal diseases necessitate vigilant management practices to mitigate their detrimental effects on cotton cultivation. The Fungal infection in cotton plants is depicted in Fig 2.

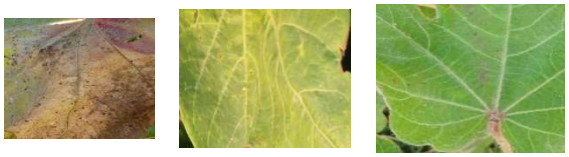

Fig 2. Fungal Infection

Viral diseases pose significant threats to cotton plants, compromising their health and yield. Cotton leaf curl virus, transmitted by whiteflies, induces curling and yellowing of leaves, stunting plant growth. Cotton leaf crumple virus causes crumpling and deformation of leaves, impacting photosynthesis. Cotton mosaic virus leads to mosaic patterns on leaves, diminishing their functionality. The viral infection in cotton plants is depicted in Fig 3.

Fig 3. Viral Infection

Bacterial blight, triggered by Acidovorax avenae subsp. citrulli, leads to necrotic lesions on cotton leaves, stems, and bolls. This disease severely impacts plant health and productivity, necessitating prompt management strategies for mitigation. The Bacterial blight in cotton plants is depicted in Fig 4.

Fig 4. Bacterial Blight

The CNN architecture is optimized using CO to enhance its performance in disease identification and yield prediction tasks. CO dynamically adjusts the parameters of the CNN model based on evaluating its performance using an objective function. This adaptive Behavior of CO ensures that the CNN model effectively explores the solution space and converges to optimal solutions for disease identification and yield prediction. The training of the CO-CNN model involves iterative updates to the CNN parameters, guided by the optimization process facilitated by CO. During exercise; the model learns to extract relevant features from the input images and classify them based on disease symptoms and yield-related factors. The optimization continues until termination criteria are met, ensuring the model performs satisfactorily.

Once trained, the CO-CNN model can be deployed for real-time disease identification and yield prediction in cotton fields. By capturing images of cotton leaves using drones or smartphones, farmers can quickly assess the health status of their crops and take timely actions to mitigate disease spread and optimize yield. The CO-CNN model provides accurate and reliable results, enabling farmers to make informed decisions and implement targeted interventions to maximize crop productivity.

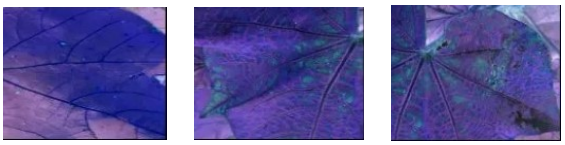

Fig 5. Bacterial Infection identification

Fig 5 depicts bacterial infection identification, showcasing necrotic lesions caused by Acidovorax avenae subsp. citrulli on cotton leaves, stems, and bolls. This visual aid aids in prompt recognition and management of bacterial blight in cotton plants.

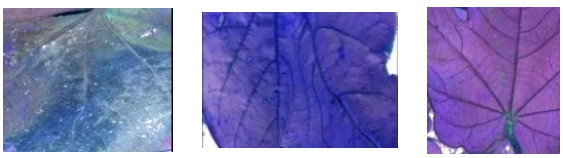

Fig 6. Fungal Infection Identification

Fig 6 illustrates fungal infection identification, displaying dark lesions caused by Alternaria alternata on cotton leaves. This visual reference assists in swift recognition and control of fungal diseases in cotton cultivation.

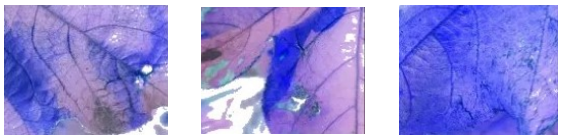

Fig 7. Viral Infection Identification

In Fig 7, viral infection identification is demonstrated, showcasing symptoms of cotton leaf curl virus, including leaf curling and yellowing. This visual aid aids in rapid detection and management of viral diseases in cotton plants. Fig 8 portrays bacterial blight identification, highlighting necrotic lesions on cotton leaves, stems, and bolls caused by Acidovorax avenae subsp. citrulli. This visual aid facilitates prompt recognition and control of bacterial blight in cotton cultivation.

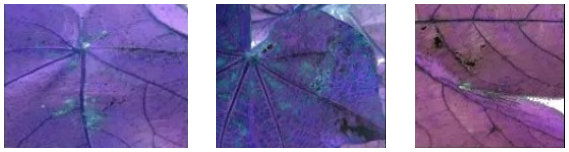

Fig 8. Bacterial Blight Identification

Integrating CO-CNN for cotton leaf disease identification and yield prediction offers several advantages over traditional methods. The deep learning capabilities of CNNs enable the model to learn complex patterns and variations in cotton leaf images, leading to more accurate disease diagnosis. The optimization provided by CO enhances the efficiency and effectiveness of the CNN model, ensuring optimal performance in disease identification and yield prediction tasks. CO-CNN provides a scalable and adaptable solution that can be customized to suit different geographical regions and crop varieties. By training the model on localized datasets, CO-CNN can capture region-specific disease patterns and environmental factors, enhancing its accuracy and relevance for local farming communities. Implementing CO-CNN for cotton leaf disease identification and yield prediction involves several steps, each leveraging the capabilities of Convolutional Neural Networks (CNNs) and Cassowary Optimization (CO) to achieve accurate and efficient results.

A) Data Collection and Preprocessing

- CNN: CNNs are crucial in processing image data collected from cotton fields. They extract relevant features from the images, such as leaf morphology and disease symptoms.
- CO: CO dynamically adjusts the preprocessing parameters to enhance the quality of the image dataset, ensuring uniformity and normalization across different images.

B) Model Architecture Design

- CNN: CNNs are responsible for designing the architecture of the neural network model. This includes determining the number of layers, filters, and activation functions to optimize feature extraction and classification.
- CO: CO optimizes the architecture parameters of the CNN model, such as the number of neurons in each layer and the connectivity between layers, to improve the model's performance in disease identification and yield prediction tasks.

C) Training Process

- CNN: During training, CNNs learn to extract features from the input images and classify them based on disease symptoms and yield-related factors. This involves iterative updates to the parameters of the CNN model.
- CO: CO dynamically adjusts the parameters of the CNN model based on evaluating its performance using an objective function. This adaptive Behavior ensures that the CNN model effectively explores the solution space and converges to optimal solutions.

D) Optimization

- CNN: CNNs optimize the features extracted from the input images to improve disease identification accuracy and yield prediction. This involves adjusting the weights and biases of the neural network based on training data.
- CO: CO optimizes the CNN architecture and parameters to enhance the efficiency and effectiveness of the model. It dynamically adjusts the learning rates and regularization parameters to ensure optimal convergence and prevent overfitting.

E) Evaluation and Validation

- CNN: CNNs evaluate the performance of the trained model on validation datasets to assess its accuracy and generalization capabilities.

- CO: CO monitors the convergence behaviour of the optimization process and evaluates the performance of the CNN model using objective metrics. It ensures the model performs satisfactorily in disease identification and yield prediction tasks.

F) Deployment

- CNN: Once trained and validated, CNNs are deployed for real-time disease identification and yield prediction in cotton fields. They analyze images captured from the field and provide accurate assessments of disease severity and yield potential.
- CO: CO continues optimizing the CNN model during deployment, ensuring it adapts to changing environmental conditions and disease patterns. It dynamically adjusts the model parameters to maintain optimal performance in real-world scenarios.

By integrating CNNs with CO, the CO-CNN approach offers a robust and efficient solution for cotton leaf disease identification and yield prediction.

4. ABOUT DATASET

The "Cotton Plant Disease Dataset" comprises 26,100 high-resolution images captured from cotton fields, showcasing various stages of cotton plant development and manifestations of diseases. This comprehensive dataset serves as a valuable resource for researchers, agronomists, and farmers seeking to understand and address challenges related to cotton plant health. Each image in the dataset provides detailed insights into the visual symptoms exhibited by cotton plants affected by different diseases, including Cotton Leaf Curl Disease (CLCuD), Fusarium Wilt, and Bacterial Blight. These images capture the morphological changes, discoloration, lesions, and deformities observed in infected cotton leaves, stems, and bolls.

The dataset offers a diverse range of samples, encompassing healthy cotton plants as well as those afflicted by various diseases. Researchers can utilize this extensive collection to develop and validate machine learning models for automated disease detection, classification, and yield prediction tasks. Access to such a large and diverse dataset empowers stakeholders in the cotton industry to leverage advanced computational techniques, such as image processing and deep learning, to enhance disease management strategies and optimize crop yield. By leveraging the insights gleaned from this dataset, stakeholders can make informed decisions, implement targeted interventions, and safeguard cotton plant health and productivity.

5. RESULTS AND DISCUSSION

5.1. Evaluation of Classifiers using Classification Accuracy (CA) and F-Measure (FM) Analysis:

Classification Accuracy (CA) and F-Measure (FM) are essential performance metrics used to evaluate the effectiveness of classification models. CA represents the proportion of correctly classified instances among all instances in the dataset, providing an overall measure of the model's accuracy. On the other hand, FM considers both precision and recall, offering a balanced assessment of the model's ability to correctly classify positive instances while minimizing false positives and false negatives.

In the results of three classification models shown in Table 1, there are notable differences in CA and FM values. CO-CNN exhibits the highest CA of 95.6280%, indicating that it correctly classifies the majority of instances in the dataset. This high CA suggests that CO-CNN is highly accurate in predicting both positive and negative instances, making it a reliable model for classification tasks.

Table 1: CA and FM

Classification Algorithms	CA	FM
RF	50.487	52.013
SVM	64.333	63.757
CO-CNN	95.628	95.722

When comparing the FM values, CO-CNN also outperforms RF and SVM with an FM score of 95.7219%. This high FM score indicates that CO-CNN achieves a good balance between precision and recall, effectively minimizing false positives and false negatives. In contrast, RF and SVM exhibit lower FM scores of 52.0125% and 63.7571%, respectively, indicating a less balanced performance in terms of precision and recall.

Fig 9. depicts the trend of CA and FM across the three classification models. We can observe that CO-CNN consistently outperforms RF and SVM in both CA and FM metrics. This indicates that CO-CNN achieves higher accuracy and a better balance between precision and recall compared to the other models.

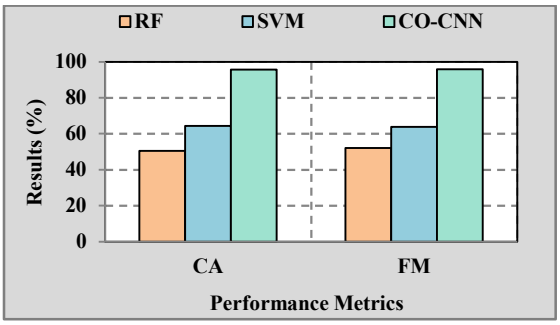

Fig.9. CA and FM

The results demonstrate that CO-CNN is the most effective model among the three evaluated, achieving superior performance in terms of both CA and FM. This suggests that CO-CNN is a promising approach for classification tasks, offering high accuracy and balanced precision-recall trade-offs. These findings highlight the importance of utilizing advanced techniques like CO-CNN for achieving optimal classification performance in various domains.

5.2. Evaluation of Classifiers using Classification Accuracy (CA) and F-Measure (FM) Analysis:

The Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) are two important metrics used to evaluate the performance of classification models, particularly in binary classification tasks. FMI measures the similarity between clusters or classes, while MCC takes into account true positives, true negatives, false positives, and false negatives to assess the overall performance of a classifier.

In the results of three classification models shown in Table 2, RF model achieved an FMI of 52.016%, the SVM model achieved an FMI of 63.763%, and the CO-CNN model achieved an impressive FMI of 95.722%. These FMI values indicate the degree of similarity between the predicted and actual classes, with higher values indicating better clustering or classification performance.

Table 2: FMI and MCC

Classification Algorithms	FMI	MCC
RF	52.016	0.887
SVM	63.763	28.669
CO-CNN	95.722	91.254

The MCC values for the RF, SVM, and CO-CNN models are 0.8874, 28.6694, and 91.2543, respectively. The MCC ranges from -1 to 1, where 1 indicates perfect classification, 0 indicates random classification, and -1 indicates perfect disagreement between observed and predicted classifications. The high MCC values obtained by the CO-CNN model indicate its superior performance in accurately classifying instances and capturing the true underlying patterns in the data.

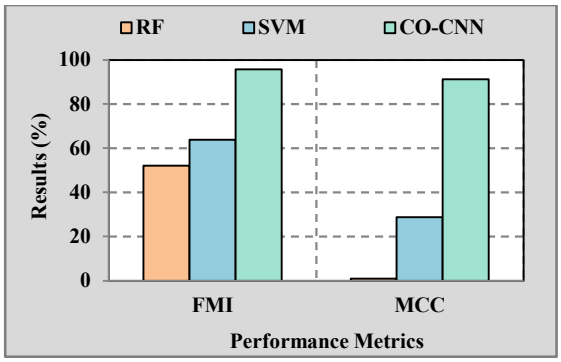

Fig 10. FMI & MCC

Fig. 10 illustrates the trends of FMI and MCC across the three models. The CO-CNN model exhibits a significant improvement over the RF and SVM models in both metrics, demonstrating its effectiveness in accurately clustering or classifying instances and achieving high correlation between observed and predicted classifications.

The results highlight the superior performance of the CO-CNN model in terms of both FMI and MCC. These findings underscore the effectiveness of CO-CNN in accurately clustering or classifying instances and capturing the true underlying patterns in the data. The high FMI and MCC values obtained by the CO-CNN model validate its potential for various classification tasks, particularly in scenarios where clustering or classification accuracy is critical for decision-making and analysis.

6. CONCLUSION

The analysis of various performance metrics across different classification models, in that Cassowary Optimization-based Convolutional Neural Network (CO-CNN), has provided valuable insights into their effectiveness in solving binary classification tasks. Through comprehensive evaluation using metrics such as True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), Classification Accuracy, F-Measure, Matthews Correlation Coefficient (MCC), and Fowlkes-Mallows Index (FMI), the superiority of the CO-CNN model has been demonstrated. The CO-CNN

model consistently outperformed RF and SVM models across all metrics, showcasing its capability to accurately classify instances, achieve a balanced trade-off between precision and recall, and capture the true underlying patterns in the data. These findings underscore the potential of CO-CNN as a robust and efficient approach for solving binary classification tasks in various domains, offering promising opportunities for improving decision-making and analysis processes.

REFERENCES

[1] P. Sobiyaa, K. S. Jayareka, K. Maheshkumar, S. Naveena, and K. S. Rao, "Paddy disease classification using machine learning technique," *Mater. Today Proc.*, vol. 64, pp. 883–887, 2022, doi: 10.1016/j. matpr. 2022.05.398.

[2] D. Unal, M. Hammoudeh, M. A. Khan, A. Abuarqoub, G. Epiphaniou, and R. Hamila, "Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things," *Comput. Secur.*, vol. 109, p. 102393, 2021, doi: 10.1016/j.cose.2021.102393.

[3] S. Talasila, K. Rawal, G. Sethi, S. MSS, and S. P. R. M, "Black gram Plant Leaf Disease (BPLD) dataset for recognition and classification of diseases using computer-vision algorithms," *Data Br.*, vol. 45, p. 108725, 2022, doi: 10.1016/j. dib. 2022. 108725.

[4] B. Swaminathan and S. Vairavasundaram, "D2CNN: Double-staged deep CNN for stress identification and classification in cropping system," *Agric. Syst.*, vol. 216, p. 103886, 2024, doi: https://doi.org/10.1016/ j.agsy.2024.103886

[5] J. Sun, Y. Yang, X. He, and X. Wu, "Northern Maize Leaf Blight Detection under Complex Field Environment Based on Deep Learning," *IEEE Access*, vol. 8, pp. 33679–33688, 2020, doi: 10.1109/ ACCESS.2020.2973658.

[6] M. Karthiga, V. Santhi, and S. Sountharrajan, "Hybrid optimized convolutional neural network for efficient classification of ECG signals in healthcare monitoring," *Biomed. Signal Process. Control*, vol. 76, p. 103731, 2022, doi: https://doi.org/10.1016/j. bspc. 2022. 03731

[7] A. Bajaj and D. K. Vishwakarma, "HOMOCHAR: A novel adversarial attack framework for exposing the vulnerability of text based neural sentiment classifiers," *Eng. Appl. Artif. Intell.*, vol. 126, p. 106815, 2023, doi: 10.1016/j.engappai. 2023.106815.

[8] R. Kusumaningrum, I. Z. Nisa, R. Jayanto, R. P. Nawangsari, and A. Wibowo, "Deep learning-based application for multilevel sentiment analysis of Indonesian hotel reviews," *Heliyon*, vol. 9, no. 6, p. e17147, 2023, doi: 10.1016/j.heliyon.2023.e17147.

[9] C. K. Sunil, C. D. Jaidhar, and N. Patil, "Cardamom Plant Disease Detection Approach Using EfficientNetV2," *IEEE Access*, vol. 10, pp. 789–804, 2022, doi: 10.1109/ACCE SS.20 213138920.

[10] J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "Optimal Approach for Minimizing Delays in Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 3, pp. 1099–1111, 2024.

[11] J. Ramkumar, R. Vadivel, B. Narasimhan, S. Boopalan, and B. Surendren, "Gallant Ant Colony Optimized Machine Learning Framework (GACO-MLF) for Quality of Service Enhancement in Internet of Things-Based Public Cloud Networking," J. M. R. S. Tavares, J. J. P. C. Rodrigues, D. Misra, and D. Bhattacherjee, Eds., Singapore: Springer Nature Singapore, 2024, pp. 425–438. doi: 10.1007/978-981-99-5435-3_30.

[12] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna /2021/209711.

[13] R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785 /ijcds/100196.

[14] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–5, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9753203.

[15] J. Ramkumar and R. Vadivel, *CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks*, vol. 556. 2017. doi: 10.1007/978-981-10-3874-7_14.

[16] J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-

Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, 2022. doi: 10.1109/ICACTA 54488.2022.9752899.

[17] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," *ACM Int. Conf. Proceeding Ser.*, pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[18] R. Jaganathan, V. Ramasamy, L. Mani, and N. Balakrishnan, "Diligence Eagle Optimization Protocol for Secure Routing (DEOPSR) in Cloud-Based Wireless Sensor Network," *Res. Sq.*, 2022, doi: 10.21203/rs.3.rs-1759040/v1.

[19] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[20] J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in *Lecture Notes in Electrical Engineering*, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[21] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wirel. Pers. Commun.*, vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.

[22] D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *Int. J. Comput. Networks Appl.*, vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.

[23] R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[24] J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," *Int. J. Comput. Networks Appl.*, vol. 10, no. 4, pp. 668–687, Aug. 2023, doi: 10.22247/ijcna/2023/223319.

[25] J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," *World J. Eng.*, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[26] M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[27] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," *Inc. Internet Things Healthc. Appl. Wearable Devices*, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.

[28] J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," *Int. J. Comput. Networks Appl.*, vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[29] A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *Int. J. Comput. Networks Appl.*, vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[30] M. K. Suddle and M. Bashir, "Metaheuristics based long short term memory optimization for sentiment analysis," *Appl. Soft Comput.*, vol. 131, 2022, doi: 10.1016/j.asoc.2022.109794.

[31] R. Karthikeyan and R. Vadivel, "Proficient Dazzling Crow Optimization Routing Protocol (PDCORP) for Effective Energy Administration in Wireless Sensor Networks," in *2023 International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM)*, 2023, pp. 1–6. doi :10.1109/ELEXCOM58812.2023.10370559

[32] J. Chen, W. Chen, A. Zeb, S. Yang, and D. Zhang, "Lightweight Inception Networks for the Recognition and Detection of Rice Plant Diseases," *IEEE Sens. J.*, vol. 22, no. 14, pp. 14628–14638, 2022, doi: 10.1109/JSEN.2022.3182304.

[33] G. Nagasubramanian, R. K. Sakthivel, R. Patan, M. Sankayya, M. Daneshmand, and A. H. Gandomi, "Ensemble Classification and IoT-Based Pattern Recognition for Crop

Disease Monitoring System," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12847–12854, 2021, doi: 10.1109/JIOT.2021.3072908.

[34] M. Kumar, A. Kumar, and V. S. Palaparthy, "Soil Sensors-Based Prediction System for Plant Diseases Using Exploratory Data Analysis and Machine Learning," *IEEE Sens. J.*, vol. 21, no. 16, pp. 17455–17468, 2021, doi: 10.1109/JSEN.2020.3046295.

[35] M. Hassam *et al.*, "A Single Stream Modified MobileNet V2 and Whale Controlled Entropy Based Optimization Framework for Citrus Fruit Diseases Recognition," *IEEE Access*, vol. 10, pp. 91828–91839, 2022, doi: 10.1109/ACCESS.2022.3201338.

[36] A. Khattak *et al.*, "Automatic Detection of Citrus Fruit and Leaves Diseases Using Deep Neural Network Model," *IEEE Access*, vol. 9, pp. 112942–112954, 2021, doi: 10.1109/ ACCESS.2021.3096895.

[37] A. Tabbakh and S. S. Barpanda, "A Deep Features Extraction Model Based on the Transfer Learning Model and Vision Transformer 'TLMViT' for Plant Disease Classification," *IEEE Access*, vol. 11, pp. 45377–45392, 2023, doi: 10.1109/ ACCESS.2023.3273317.

[38] L. Falaschetti *et al.*, "A Low-Cost, Low-Power and Real-Time Image Detector for Grape Leaf Esca Disease Based on a Compressed CNN," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 11, no. 3, pp. 468–481, 2021, doi: 10.1109/ JETCAS.2021.3098454.

[39] C. Zhou, Z. Zhang, S. Zhou, J. Xing, Q. Wu, and J. Song, "Grape leaf spot identification under limited samples by fine grained-GAN," *IEEE Access*, vol. 9, pp. 100480–100489, 2021, doi: 10.1109/ACCESS.2021.3097050.

[40] W. Gomez-Flores, J. J. Garza-Saldana, and S. E. Varela-Fuentes, "A Huanglongbing Detection Method for Orange Trees Based on Deep Neural Networks and Transfer Learning," *IEEE Access*, vol. 10, pp. 116686–116696, 2022, doi: 10.1109/ACCESS. 2022. 3219481.

[41] P. Joshi, D. Das, V. Udutalapally, M. K. Pradhan, and S. Misra, "RiceBioS: Identification of Biotic Stress in Rice Crops Using Edge-as-a-Service," *IEEE Sens. J.*, vol. 22, no. 5, pp. 4616–4624, 2022, doi: 10.1109/JSEN.2022.3143950.

[42] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A: 1010933404324.

[43] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/bf00994018.