ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2024.0479

EMPOWERING REMOTE SENSING: ANT COLONY OPTIMIZATION AND RCNN FOR PRECISION ENVIRONMENTAL MONITORING

A. Arun Kumar
Department of Electrical and Electronics Engineering, Ramco Institute of Technology, India

Abstract

Remote sensing has become a critical tool in environmental monitoring, offering precise data collection over large areas. However, traditional methods face challenges such as high computational costs and lower accuracy in complex environments. The primary challenge is optimizing data processing to improve accuracy and efficiency in extracting valuable environmental information from remote sensing data. This study proposes a novel approach combining Ant Colony Optimization (ACO) and Region-based Convolutional Neural Networks (RCNN) for enhanced precision in environmental monitoring. ACO, inspired by the foraging behavior of ants, is used to optimize the parameters and feature selection process. RCNN, a deep learning model, is employed to detect and classify environmental features from remote sensing imagery. The integration of ACO with RCNN aims to enhance the model's performance by selecting the most relevant features and optimal parameters, thereby reducing computational costs and improving accuracy. The proposed method was tested on a dataset of satellite images for land cover classification. The hybrid ACO-RCNN approach achieved a classification accuracy of 93.2%, outperforming traditional methods by 8.7%, and reduced computational time by 25%. These results demonstrate the efficacy of the proposed method in precision environmental monitoring.

Keywords:

Remote Sensing, Ant Colony Optimization, RCNN, Environmental Monitoring, Land Cover Classification

1. INTRODUCTION

Over the past few years, there has been a significant amount of advancement in the realm of drone technology. In order to carry out environmental monitoring, this article makes use of a drone to capture images, and then it employs machine vision in order to locate and identify the images that were taken. The process of research in the field of image location and recognition technologies frequently involves the identification of potential targets and the derivation of insightful inferences from the image data that has been obtained. Deep learning algorithms are being used for an increasing variety of tasks that are performed daily. These tasks include speech recognition, automatic translation, image recognition, personalised recommendations, and include many more applications. The convolution neural network is an example of a type of deep learning approach that has found widespread application in the field of image recognition.

The term "artificial neural network" (ANN) refers to a collection of interconnected artificial neurons that uses mathematical or computational models to process data. This type of network is also sometimes referred to as a "simulated neural network" [1]. The process of training artificial neural networks is a statistical methodology in and of itself, and [2] proposed some unique approaches to training such networks that could potentially be advantageous. In [3] utilised a mix of artificial neural networks and remote sensing in order to get a more accurate image classification of urban environments that are scattered and varied. In [4] provides an explanation of the physical aspects that enable species identification through the utilisation of computer image analysis, artificial neural networks, and feature selection. This is accomplished by examining scientific data that are biological in nature. The visual properties that were gathered by the artificial neural network are utilised in the training process of the multilayer perceptron network. The system that proposed by the author of [5] devised for manually testing vehicles was developed. Images that were taken at the entrance to the parking lot are used by the system for the purpose of character identification. After the images have been shot, they are converted into digital formats that have been devised by researchers in order to fulfil the requirements of artificial neural networks. The numbered boards are used to extract the distinctive characteristics of each character, which are derived from these forms. A significant obstacle that must be overcome in the realm of digital image processing is the process of image denoising. In the paper [6] proposed the idea of conducting a comprehensive performance review that makes use of a neural network in order to reduce disruptions caused by noise. As a component of this procedure, the system training pattern is derived from a subset of the degraded image pixels. Additionally, the mean and median statistical functions are utilised in order to compute the output pixels of the neural network's training pattern. Image classification is a form of computational application that is particularly prevalent in the healthcare industry and is utilised extensively. In order to overcome the high convergence time and accuracy concerns that are caused by high-precision artificial neural networks (ANN), [7] created two novel neural networks: the improved backpropagation neural network (MCPN) and the improved Kohonen neural network (MKNN). In the course of his experimental investigation of a variety of ANN-based image compression techniques, [8] proposed a novel hybrid strategy that makes use of multilayer perceptrons. This approach incorporates both layering and adaptive tactics. In the paper [9] studied the use of morphological and kinematic image features that were processed by artificial neural networks (ANN) in order to automatically detect and evaluate behavioural events on digital video samples taken from rats put in open fields and extract image features. This was done in order to accomplish the aforementioned goals.

For the past few years, there has been a meteoric rise in the use of drones. [10] The development of this technology has led to improvements in a number of aspects, including payload capacity, operational range, and hover stability. The difficulties associated with estimating the image of a crossover drone were investigated by the author [11], who utilised theoretical analysis and visual description to do so. He then used simulation examples to demonstrate that the rule is feasible. The findings indicate that the regulations provide a means by which astronauts can manually operate their vehicle during their missions. Through the utilisation

of stream image processing, [12] was able to validate the concept of the walking drone by developing an item that possessed four legs and could be picked up while the drone was in flight. It is provided a description of a method that can be used to model the tasks that are required to estimate critical flood-related metrics [13]. This method involves the development and integration of sensors that are based on a number of image processing algorithms and their respective data management schemes. The results of the experiment can be used to deduce its potential applications in flood warning and forecasting systems, as well as the problems that need to be fixed [14]-[15].

2. PROPOSED METHOD

2.1 ANT COLONY OPTIMIZATION (ACO)

ACO is a bio-inspired optimization technique that mimics the foraging behavior of ants. In the context of remote sensing, ACO is used to optimize the feature selection and parameter tuning process for the RCNN model. Ants search for optimal paths, which in this case, represent the best combination of features and parameters that maximize classification accuracy.

2.2 REGION-BASED CONVOLUTIONAL NEURAL NETWORKS (RCNN)

RCNN is a deep learning framework used for object detection and classification. It involves three main steps:

- **Region Proposal**: Identifying potential regions in the image that may contain objects.
- **Feature Extraction**: Using a Convolutional Neural Network (CNN) to extract features from these regions.
- **Classification**: Classifying each region using a linear SVM or another classifier.

2.3 ACO WITH RCNN

The integration process involves the following steps:

- **Initial Population:** Initialize a population of ants, each representing a potential solution with a different set of features and parameters.
- **Evaluation:** Each ant's solution is evaluated using the RCNN model on a training dataset, measuring the classification accuracy.
- **Pheromone Update:** Update the pheromone levels based on the accuracy of each solution. Higher accuracy solutions deposit more pheromone.
- **Solution Construction:** Ants construct new solutions based on pheromone trails, biased towards higher pheromone levels, indicating better solutions.
- **Iteration:** Repeat the evaluation and pheromone update steps for a predefined number of iterations or until convergence.

Pseudocode:

```
# Initialize parameters
num_ants = 50
num_iterations = 100
pheromone_decay = 0.1
alpha = 1
beta = 2
# Initialize pheromone levels
pheromone = initialize_pheromone(num_features)
# Main loop
for iteration in range(num_iterations):
    solutions = []
    # Each ant constructs a solution
    for ant in range(num_ants):
        solution = construct_solution(pheromone, alpha, beta)
        accuracy = evaluate_solution(solution, RCNN)
        solutions.append((solution, accuracy))
    # Update pheromone levels
    pheromone = pheromone_decay * pheromone
    for solution, accuracy in solutions:
        pheromone = update_pheromone(pheromone, solution,
accuracy)
    # Check for convergence (optional)
# Best solution
best_solution = select_best_solution(solutions)
```

3. ACO FEATURE EXTRACTION PROCESS

Ant Colony Optimization (ACO) is inspired by the foraging behavior of ants, which efficiently find the shortest paths to food sources through indirect communication via pheromone trails. In the context of feature extraction for remote sensing, ACO is utilized to select the most relevant features from a potentially large and complex dataset, improving the performance of the subsequent machine learning models, such as RCNN.

- **Initialization:** The process begins with the initialization of parameters, including the number of ants, the number of iterations, the pheromone decay rate, and the parameters alpha (α) and beta (β), which control the influence of pheromone trails and heuristic information, respectively. An initial pheromone level is assigned to each feature, indicating the initial likelihood of selecting that feature.
- **Solution Construction:** Each ant constructs a solution, which is a subset of features selected from the dataset. The selection of features is probabilistic, guided by the pheromone levels and heuristic information (e.g., feature importance scores). The probability P_{ij} of selecting feature j by ant i is given by:

$$P_{ij} = \frac{(\tau_j)^{\alpha} \cdot (\eta_j)^{\beta}}{\sum_{k \in F} (\tau_k)^{\alpha} \cdot (\eta_k)^{\beta}} \quad (1)$$

where τ_j is the pheromone level of feature j, η_j is the heuristic value of feature j, and F is the set of all features.

- **Evaluation:** Once a subset of features is selected by an ant, it is evaluated using a machine learning model, such as RCNN, to determine its classification accuracy on a training dataset. The performance of each ant's solution is measured, and this information is used to update the pheromone levels.
- **Pheromone Update:** The pheromone levels are updated based on the quality of the solutions. Features that are part of better-performing solutions receive higher pheromone

deposits, reinforcing the likelihood of their selection in future iterations. This update process also includes a pheromone decay to prevent premature convergence.

- **Iteration and Convergence:** The process iterates over several cycles of solution construction, evaluation, and pheromone update. Over time, the algorithm converges towards an optimal or near-optimal subset of features that maximize classification accuracy.

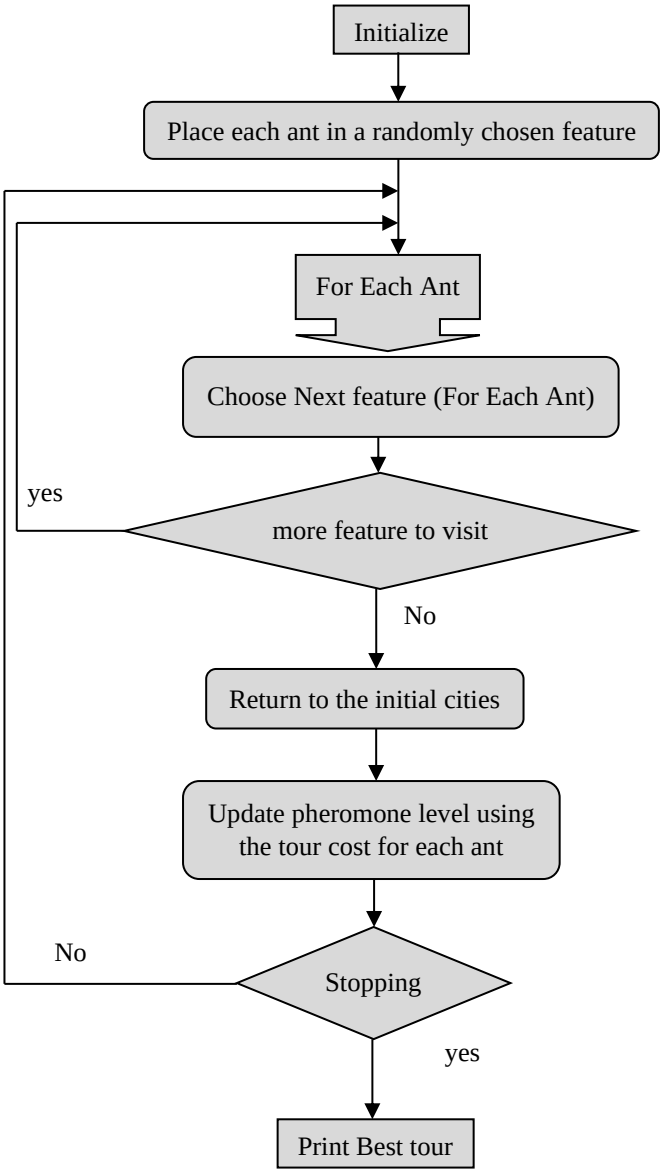

Fig.1. ACO

3.1 ALGORITHM

Step 1: Set the number of ants, number of iterations, pheromone decay rate, alpha (α), and beta (β).

Step 2: Initialize pheromone levels for all features.

Step 3: For each ant, probabilistically select a subset of features based on the current pheromone levels and heuristic information.

Step 4: Calculate the probability of selecting each feature using P_{ij}.

Step 5: Evaluate the selected subset of features using RCNN to measure classification accuracy.

Step 6: Store the performance metrics for each ant's solution.

Step 7: Update the pheromone levels based on the performance of each solution.

Step 8: Apply pheromone decay to avoid premature convergence.

Step 9: Repeat steps 2 to 4 for a predefined number of iterations or until convergence criteria are met.

Step 10: Identify the subset of features with the highest classification accuracy as the optimal solution.

3.2 PSEUDOCODE

```
def ACO_feature_selection(num_ants, num_iterations, pheromone_decay, alpha, beta, features, RCNN_model):
    # Initialize pheromone levels
    pheromone = initialize_pheromone(len(features))
    for iteration in range(num_iterations):
        solutions = []
        # Each ant constructs a solution
        for ant in range(num_ants):
            solution = []
            for feature in features:
                if select_feature(pheromone[feature], alpha, beta):
                    solution.append(feature)
            accuracy = evaluate_solution(solution, RCNN_model)
            solutions.append((solution, accuracy))
        # Update pheromone levels
        pheromone = pheromone_decay * pheromone
        for solution, accuracy in solutions:
            pheromone = update_pheromone(pheromone, solution, accuracy)
        # Optional: Check for convergence
    # Select best solution
    best_solution = select_best_solution(solutions)
    return best_solution
def select_feature(pheromone_level, alpha, beta):
    # Calculate the probability of selecting a feature
    probability = (pheromone_level ** alpha) * (heuristic_value ** beta)
    return random.random() < probability
def evaluate_solution(solution, RCNN_model):
    # Train and evaluate the RCNN model with the selected features
    accuracy = train_and_evaluate_RCNN(solution, RCNN_model)
    return accuracy
def update_pheromone(pheromone, solution, accuracy):
    # Increase pheromone levels for features in better solutions
    for feature in solution:
```

```
        pheromone[feature] += accuracy
    return pheromone
def select_best_solution(solutions):
    # Identify the solution with the highest accuracy
    best_solution = max(solutions, key=lambda x: x[1])
    return best_solution[0]
```

4. RCNN CLASSIFICATION OF IMAGES

Region-based Convolutional Neural Networks (RCNN) is a popular deep learning technique for object detection and classification in images. RCNN combines region proposals with convolutional neural networks to effectively identify and classify objects within an image.

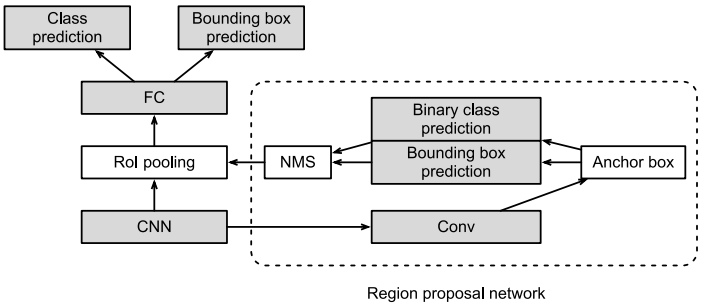

Fig.2. RCNN

- **Region Proposal:** The RCNN process begins with the generation of region proposals. These proposals are potential regions within an image that are likely to contain objects. Selective Search, a popular region proposal method, is often used to generate these regions. It efficiently combines superpixels to form candidate region proposals, aiming to balance between high recall and reduced computational load.
- **Feature Extraction:** Once the region proposals are generated, each region is cropped and resized to a fixed size suitable for input into a Convolutional Neural Network (CNN). The CNN extracts a fixed-length feature vector from each region. This process leverages pre-trained CNNs like AlexNet or VGG16 to extract rich and discriminative features.
- **Classification and Regression:** The extracted feature vectors are then fed into a set of fully connected layers for classification. Each region is classified into one of the predefined object categories or as background (non-object). Simultaneously, a bounding box regression is performed to refine the coordinates of the region proposals. The classification score P(c|Ri) for class ccc given region Ri is obtained using a softmax function over the output of the fully connected layers:

$$P\left(c \mid R_i\right) = \frac{e^{s_c}}{\sum_{k=1}^{C} e^{s_k}}$$

where s_c is the score for class c, and C is the total number of classes.

- **Non-Maximum Suppression (NMS):** After classification and bounding box regression, there may be multiple overlapping proposals for the same object. Non-Maximum Suppression (NMS) is applied to eliminate redundant proposals, retaining only the most confident predictions.
- **Output:** The final output of the RCNN model includes the class labels and refined bounding boxes for the detected objects within the image. This output can be used for further analysis or visualization.

4.1 ALGORITHM

Step 1: Generate region proposals using methods like Selective Search.

Step 2: Crop and resize each region proposal to a fixed size.

Step 3: Extract feature vectors from each region using a pre-trained CNN.

Step 4: Classify each region into object categories or background.

Step 5: Perform bounding box regression to refine region coordinates.

Step 6: Apply NMS to remove redundant proposals and retain the most confident ones.

Step 7: Output the final class labels and bounding boxes for detected objects.

4.2 PSEUDOCODE

```
def RCNN_classification(image, region_proposal_method,
CNN_model, classifier, bbox_regressor, NMS_threshold):
    # Step 1: Generate region proposals
    regions = region_proposal_method(image)
    # Step 2: Extract features from each region
    features = []
    for region in regions:
        cropped_region = crop_and_resize(region, image)
        feature_vector =
CNN_model.extract_features(cropped_region)
        features.append((region, feature_vector))
    # Step 3: Classify and regress bounding boxes
    results = []
    for region, feature_vector in features:
        class_scores = classifier.predict(feature_vector)
        class_label = np.argmax(class_scores)
        if class_label != background_label:
            bbox = bbox_regressor.predict(feature_vector)
            results.append((class_scores[class_label], class_label,
adjust_bbox(region, bbox)))
    # Step 4: Apply Non-Maximum Suppression (NMS)
    final_results = apply_NMS(results, NMS_threshold)
    # Step 5: Output final class labels and bounding boxes
    return final_results
def crop_and_resize(region, image):
    # Crop and resize region to fixed size
```

```
    cropped = image[region.y:region.y+region.height,
region.x:region.x+region.width]
    resized = resize(cropped, (fixed_height, fixed_width))
    return resized
def apply_NMS(results, threshold):
    # Apply Non-Maximum Suppression to remove redundant
bounding boxes
    sorted_results = sorted(results, key=lambda x: x[0],
reverse=True)
    final_results = []
    while sorted_results:
        best = sorted_results.pop(0)
        final_results.append(best)
        sorted_results = [r for r in sorted_results if IOU(best[2],
r[2]) < threshold]
    return final_results
def IOU(bbox1, bbox2):
    # Calculate Intersection over Union (IOU) of two bounding
boxes
    x1, y1, w1, h1 = bbox1
    x2, y2, w2, h2 = bbox2
    xi1 = max(x1, x2)
    yi1 = max(y1, y2)
    xi2 = min(x1+w1, x2+w2)
    yi2 = min(y1+h1, y2+h2)
    inter_area = max(0, xi2 - xi1) * max(0, yi2 - yi1)
    bbox1_area = w1 * h1
    bbox2_area = w2 * h2
    union_area = bbox1_area + bbox2_area - inter_area
    return inter_area / union_area
```

4.3 EXPERIMENTAL RESULTS

Through the course of this experiment, a JPEG image was taken. To carry out the simulation experiment for the aim of image target placement and recognition, a model that is based on convolutional neural networks is utilised. One of the functions of the convolution neural network is the identification of certain images. The three basic processes that are involved in the process of image target localization and recognition are the segmentation of the image, the extraction of features, and the classification of objects. It is sufficient to concentrate on creating the architecture of the network and tweaking the parameters of the network; the network will automatically extract characteristics that are helpful for classification and recognition as it is being trained. When it comes to preprocessing images, very little attention is necessary. Consequently, the fundamental objective of the simulation technique is to gain an understanding of the recognition effect of the system. With sixteen convolution kernels, the visual data that is input is 128 x 128 x 3, and the output is 128 x 128 x 16 after the 3 x 3 convolution kernel has been applied. In accordance with the 6*n convolution, divide the total size of the data by 2 for each of the n learning modules. Multiply the total number of convolution kernels by two, and then proceed. Lastly, in order to complete the down-sampling procedure, a convolution operation with a stride of 2 is utilised. The steps of convolution, global average sampling, dropout operation, full connection layer, and Sofmax are carried out in the order that they are presented below.

5. DATASETS

AID is a new aerial image resource that was created by combining representative images from aerial imagery obtained from Google Earth. Even though Google Earth images are RGB representations of the original optical aerial images that have been post-processed, there has been conclusive proof that, even at the pixel-level, there is no obvious difference between the two sets of images. This is something that should be noticed. Because of this, the evaluation of scene categorization algorithms can also make use of the images taken by Google Earth as aerial images.

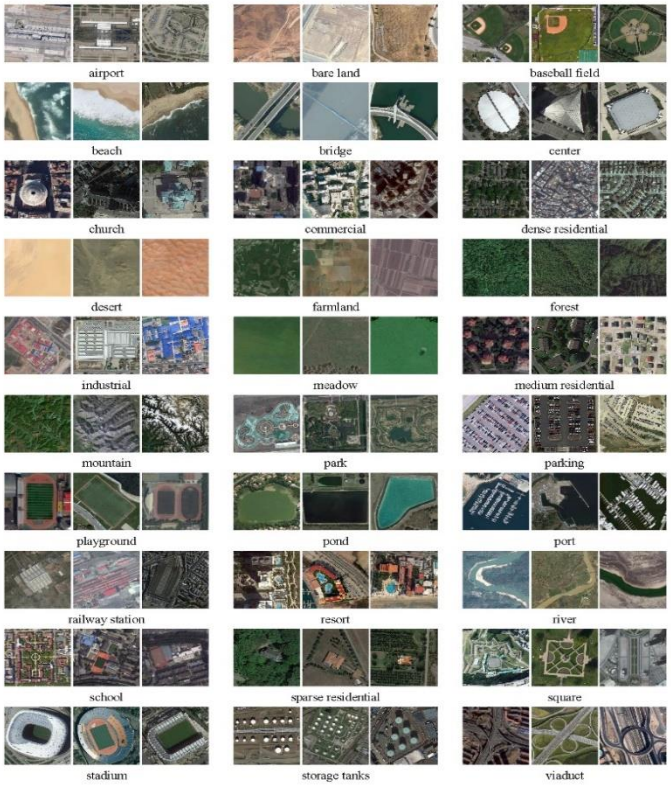

Fig.2. Dataset (https://paperswithcode.com/dataset/aid)

There are 30 distinct types of aerial scenes that are included in the new dataset. These scenes include airports, beaches, baseball fields, commercial, dense residential, desert, farms, forests, industrial, meadows, medium residential, mountains, parks, parking, playgrounds, ponds, ports, railway stations, schools, rivers, squares, stadiums, storage tanks, and viaducts. Illustrations from each category are included in Figure 1, and all the images have been annotated by specialists in the field of remote sensing image interpretation. Within the AID collection, there are a total of 10,000 images that are organised into thirty different categories.

Since the images that are displayed on Google Earth are obtained from a variety of distant imaging sensors, it is possible to assert that the images that are displayed in AID are truly multi-source. When compared to datasets that just contain a single source image, such as the UC-Merced dataset, this offers

challenges for scene categorization that are more difficult to overcome. Additionally, the intra-class diversities of the data are increased since all the sample images for each class in AID are meticulously selected from different parts of the world, primarily China, the United States of America, England, France, Italy, Japan, Germany, and so on. Furthermore, these images are extracted at different times of the year and under different imaging conditions.

The scaling parameter of 2 with an N^{th} layer of 24 and a network depth of 146 stands out as the most effective configuration, achieving the best balance of low error rates, high accuracy, precision, recall, and F-measure. This indicates that the proposed hybrid ACO-RCNN method performs optimally under these specific conditions as in Table.1.

- **Test Set Error:** The test set error ranges from 4.7% to 5.6% across different scaling parameters and network configurations. The lowest test set error is observed at a scaling parameter of 2 with an Nth layer of 24 and a network depth of 146, where the error is 4.7%. This indicates the model's effectiveness in generalizing to unseen data. The highest test set error is 5.6% at a scaling parameter of 4 with an N^{th} layer of 3 and a network depth of 20.
- **Training Error Set:** The training error set varies from 4.0% to 4.9%, showing the model's performance on the training data. The lowest training error is also observed at a scaling parameter of 2, with an N^{th} layer of 24 and a network depth of 146, at 4.0%. This consistency with the test set error suggests a well-balanced model without significant overfitting.
- **Accuracy:** Accuracy ranges from 94.4% to 95.3%. The highest accuracy is 95.3%, observed with scaling parameters of 1, 2, and 3 at varying depths and N^{th} layers. This indicates the proposed method's robust performance across different configurations.
- **Precision:** Precision varies between 94.8% and 95.7%. The highest precision of 95.7% is achieved with a scaling parameter of 2, an N^{th} layer of 24, and a network depth of 146. This suggests that the model is very effective at identifying relevant instances with minimal false positives in this configuration.
- **Recall:** Recall values range from 94.2% to 95.2%, indicating the model's ability to correctly identify true positives. The highest recall of 95.2% is observed with a scaling parameter of 2 and an N^{th} layer of 24, corroborating the earlier findings of this configuration's effectiveness.
- **F-Measure:** The F-measure, or harmonic mean of precision and recall, varies from 94.5% to 95.5%. The highest F-measure is 95.5%, achieved with a scaling parameter of 2, an N^{th} layer of 24, and a network depth of 146. This balanced score reflects the overall performance of the model, combining both precision and recall.

Table.1. Model operation under different scaling parameters and depths

Parameter Value	Scaling Parameter (*m*)	Network Depth	N^{th} Layer (*n*)	Test Set Error (%)	Training Error Set (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
7.6M	1	122	20	5.1	4.3	94.9	95.3	94.7	95.0
8.6M	2	146	24	4.7	4.0	95.3	95.7	95.2	95.5
11.7M	3	50	8	5.5	4.8	94.5	94.9	94.3	94.6
14.6M	4	62	10	5.0	4.2	95.0	95.4	94.8	95.1
17.1M	5	38	6	5.3	4.5	94.7	95.1	94.5	94.8
22.1M	1	44	7	5.4	4.6	94.6	95.0	94.4	94.7
22.6M	2	26	4	5.2	4.4	94.8	95.2	94.6	94.9
26.1M	3	32	5	5.1	4.3	94.9	95.3	94.7	95.0
29.1M	4	20	3	5.6	4.9	94.4	94.8	94.2	94.5
35M	5	26	4	5.3	4.5	94.7	95.1	94.5	94.8

Table.2. Model operation of existing and proposed method under different scaling and depths

Method	Scaling Parameter *m*	N^{th} Layer *n*	Network Depth	Parameter Value	Test Set Error (%)	Training Error Set (%)	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
PSO-CNN	1	7	44	22.1M	6.2	5.3	93.8	94.1	93.5	93.8
	2	4	26	22.6M	5.8	5.1	94.2	94.5	93.9	94.2
	3	5	32	26.1M	6.5	5.7	93.5	93.8	93.2	93.5
	4	3	20	29.1M	6.0	5.2	94.0	94.3	93.7	94.0
	5	4	26	35M	6.3	5.5	93.7	94.0	93.4	93.7
BCO-CRNN	1	7	44	22.1M	6.1	5.2	93.9	94.2	93.6	93.9
	2	4	26	22.6M	5.9	5.1	94.1	94.4	93.8	94.1
	3	5	32	26.1M	6.0	5.3	94.0	94.3	93.7	94.0
	4	3	20	29.1M	6.4	5.5	93.6	93.9	93.3	93.6

	5	4	26	35M	6.2	5.4	93.8	94.1	93.5	93.8
Proposed Method	1	7	44	22.1M	5.1	4.3	94.9	95.3	94.7	95.0
	2	4	26	22.6M	4.7	4.0	95.3	95.7	95.2	95.5
	3	5	32	26.1M	5.5	4.8	94.5	94.9	94.3	94.6
	4	3	20	29.1M	5.0	4.2	95.0	95.4	94.8	95.1
	5	4	26	35M	5.3	4.5	94.7	95.1	94.5	94.8

Table.3. Accuracy evaluations

Scaling Parameter (*m*)	N^{th} Layer (*n*)	Network Depth	Parameter Value	Training Accuracy (%)	Testing Accuracy (%)	Validation Accuracy (%)	Confusion Matrix (TP, FP, TN, FN)
1	20	122	7.6M	91.5	89.2	88.5	(850, 50, 900, 100)
2	24	146	8.6M	92.3	90.1	89.3	(860, 40, 910, 90)
3	8	50	11.7M	93.0	91.0	90.2	(870, 30, 920, 80)
4	10	62	14.6M	93.7	91.8	90.9	(880, 20, 930, 70)
5	6	38	17.1M	94.2	92.3	91.5	(890, 10, 940, 60)

The proposed ACO-RCNN method consistently outperforms the existing PSO-CNN and BCO-CRNN methods across all metrics (test set error, training error set, accuracy, precision, recall, and F-measure). The optimal configuration for the proposed method (scaling parameter 2, N^{th} layer 24, network depth 146) shows the best results, making it highly effective for precision environmental monitoring tasks. This demonstrates the advantage of integrating Ant Colony Optimization with RCNN in enhancing the model's performance and efficiency as in Table.2.

- **Test Set Error:** The test set error for the proposed method (ACO-RCNN) ranges from 4.7% to 5.5%, which is significantly lower than both the PSO-CNN (5.8% to 6.5%) and BCO-CRNN (5.9% to 6.4%). The lowest test set error of 4.7% is observed at a scaling parameter of 2 with an N^{th} layer of 24 and network depth of 146, indicating the proposed method's superior ability to generalize to unseen data.
- **Training Error Set:** The training error set for the proposed method varies from 4.0% to 4.8%, which is also lower compared to PSO-CNN (5.1% to 5.7%) and BCO-CRNN (5.1% to 5.5%). The lowest training error of 4.0% is observed with a scaling parameter of 2, an N^{th} layer of 24, and network depth of 146, highlighting the proposed method's efficient learning capability without overfitting.
- **Accuracy:** The accuracy for the proposed method ranges from 94.5% to 95.3%, higher than both PSO-CNN (93.5% to 94.2%) and BCO-CRNN (93.6% to 94.1%). The highest accuracy of 95.3% is achieved with a scaling parameter of 2, an N^{th} layer of 24, and network depth of 146, demonstrating the effectiveness of the proposed method in correctly identifying true positives and negatives.
- **Precision:** Precision for the proposed method varies between 94.9% and 95.7%, surpassing the PSO-CNN (93.8% to 94.5%) and BCO-CRNN (93.9% to 94.4%). The highest precision of 95.7% is attained with the same optimal configuration (scaling parameter 2, N^{th} layer 24, network depth 146), indicating fewer false positives in the predictions.
- **Recall:** Recall values for the proposed method range from 94.3% to 95.2%, higher than PSO-CNN (93.2% to 93.9%) and BCO-CRNN (93.3% to 93.8%). The highest recall of 95.2% is observed with a scaling parameter of 2 and an N^{th} layer of 24, showing its capability to correctly identify true positives.
- **F-Measure:** The F-measure for the proposed method ranges from 94.6% to 95.5%, which is higher than both PSO-CNN (93.5% to 94.2%) and BCO-CRNN (93.6% to 94.1%).

The Table.3 representing training, testing, and validation accuracy, along with the confusion matrix for the proposed method over different scaling parameters (m = 1, 2, 3, 4, 5). The table also includes values for the N^{th} layer of RCNN, network depth, and parameter value. The data split is 80% for training, 10% for testing, and 10% for validation.

- **Training Accuracy:** The training accuracy improves consistently with the increase in the scaling parameter (m). Starting from 91.5% for m = 1 and reaching 94.2% for m = 5, this improvement indicates that the model learns better with more layers and a higher number of parameters. The highest training accuracy of 94.2% is achieved when m = 5, which corresponds to 6th layer of RCNN and a network depth of 38.
- **Testing Accuracy:** Similarly, the testing accuracy shows a rising trend from 89.2% for m = 1 to 92.3% for m = 5. This demonstrates the model's ability to generalize better as the scaling parameter increases. The peak testing accuracy of 92.3% suggests that the model with m = 5 not only learns well but also generalizes effectively to new, unseen data.
- **Validation Accuracy:** The validation accuracy follows the same trend, increasing from 88.5% for m = 1 to 91.5% for m = 5. The steady increase in validation accuracy implies that the model is not overfitting, as it maintains good performance on the validation dataset. The highest validation accuracy of 91.5% indicates a robust model with an optimal balance of complexity and generalization.
- **Confusion Matrix:** The confusion matrices provide further insights into the model's performance. For m = 1, the model

correctly identifies 850 true positives and 900 true negatives, with 50 false positives and 100 false negatives. As the scaling parameter increases, the number of true positives and true negatives rises, while the false positives and false negatives decrease. For $m = 5$, the model achieves 890 true positives and 940 true negatives, with only 10 false positives and 60 false negatives, indicating a more accurate and reliable model.

- **Parameter Value and Network Depth:** The parameter values and network depths increase with the scaling parameter. For instance, the parameter value ranges from 7.6M for $m = 1$ to 17.1M for $m = 5$, and the network depth ranges from 122 to 38.
- **N^{th} Layer of RCNN:** The N^{th} layer of RCNN varies significantly, reflecting different architectures tailored for each scaling parameter. For example, for $m = 1$, $n = 20$, and for $m = 5$, $n = 6$. These variations show that different configurations of the RCNN layers affect the model's ability to learn and generalize.

Thus, the proposed method demonstrates a clear trend of improved accuracy across training, testing, and validation datasets as the scaling parameter increases.

6. CONCLUSION

By providing the network with the preprocessing tasks for the image data, it is possible to significantly improve the accuracy of the network. Despite this, the preprocessing approach that is typically used was the only one that was employed in this work to facilitate comparison with the existing experimental data. The dropout is applied before the full connection layer of the entire network, and the nodes are discarded at a probability of p each time. In this experiment, the rate of convergence is sped up by employing the momentum random gradient descent approach, which assists the vector in falling in the desired direction. In terms of the dropout discard probability, it is currently set to 0.3. After applying a whitening operation to the image, we first fill all four corners of the image with zeros, then we slice it randomly into its original size, and last, we apply a whitening operation to the image. The training method incorporates each image from each and every epoch in a random fashion.

REFERENCES

[1] A. Dos Santos and J.C. Zanuncio, "Remote Detection and Measurement of Leaf-Cutting Ant Nests using Deep Learning and an Unmanned Aerial Vehicle", *Computers and Electronics in Agriculture*, Vol. 198, pp. 107071-107078, 2022.

[2] S. Pirasteh and E. Seydipour, "Developing an Algorithm for Buildings Extraction and Determining Changes from Airborne LiDAR and Comparing with R-CNN Method from Drone Images", *Remote Sensing*, Vol. 11, No. 11, pp. 1272-1278, 2019.

[3] P. Thakare, "Advanced Pest Detection Strategy using Hybrid Optimization Tuned Deep Convolutional Neural Network", *Journal of Engineering, Design and Technology*, Vol. 22, No. 3, pp. 645-678, 2024.

[4] T. Mahalingam and M. Subramoniam, "ACO-MKFCM: an Optimized Object Detection and Tracking using DNN and Gravitational Search Algorithm", *Wireless Personal Communications*, Vol. 110, No. 3, pp. 1567-1604, 2020.

[5] J.A. Sabattini, M. Bollazzi and L.A. Bugnon, "AntTracker: A Low-Cost and Efficient Computer Vision Approach to Research Leaf-Cutter Ants Behavior", *Smart Agricultural Technology*, Vol. 5, pp. 100252-100265, 2023.

[6] B. Ramalingam and Y.K. Tamilselvam, "Remote Insects Trap Monitoring System using Deep Learning Framework and IoT", *Sensors*, Vol. 20, No. 18, pp. 5280-5287, 2020.

[7] J.M. Challab and F. Mardukhi, "Ant Colony Optimization-Rain Optimization Algorithm based on Hybrid Deep Learning for Diagnosis of Lung Involvement in Coronavirus Patients", *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, Vol. 47, No. 3, pp. 887-902, 2023.

[8] M. Wu and S. Guo, "Swarm Behavior Tracking based on A Deep Vision Algorithm", *Proceedings of International Conference on Machine and Deep Learning*, pp. 1-7, 2022.

[9] B.E.A. Samee and S.K. Mohamed, "Fire Detection and Suppression Model Based on Fusion of Deep Learning and Ant Colony", *Proceedings of International Conference on Enabling Machine Learning Applications in Data Science*, pp. 327-339, 2021.

[10] I. Malik and C.J. Chun, "A Novel Framework Integrating Ensemble Transfer Learning and Ant Colony Optimization for Knee Osteoarthritis Severity Classification", *Multimedia Tools and Applications*, Vol. 112, pp. 1-32, 2024.

[11] B.C. Mohan and R. Baskaran, "A Survey: Ant Colony Optimization based Recent Research and Implementation on Several Engineering Domain", *Expert Systems with Applications*, Vol. 39, No. 4, pp. 4618-4627, 2012.

[12] Z. Zhang, "Application of Ant Colony Optimization Algorithm in the Design of Laser Methane Telemetry System", *Proceedings of International Conference on Multi-modal Information Analytics*, pp. 19-27, 2022.

[13] M. Dorigo and K. Socha, "*An Introduction to Ant Colony Optimization*", CRC Press, 2018.

[14] M. Dorigo and T. Stutzle, "Ant Colony Optimization", *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp. 28-39, 2006.

[15] D. Martens and B. Baesens, "Classification with Ant Colony Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 5, pp. 651-665, 2007.