Journal of Theoretical and Applied Information Technology
15th October 2024. Vol.102. No. 19

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

EMUBOOST OPTIMIZED ENSEMBLE MODEL FOR ROBUST UNDERWATER IMAGE CLASSIFICATION

SARAVANAN P[1], VADIVAZHAGAN K[2]

[1]Research Scholar & Assistant Professor, Department of Computer and Information Science
Annamalai University, Chidambaram, Tamilnadu, India
[2]Assistant Professor, Department of Computer and Information Science,
Annamalai University, Chidambaram, Tamilnadu, India
E-mail: [1]aucse.saran@gmail.com, [2]vadivazhagan.k@gmail.com

ID 55578 Submission	Editorial Screening	Conditional Acceptance	Final Revision Acceptance
12-09-2024	14-09-2024	27-09-2024	29-09-2024

ABSTRACT

Underwater image classification faces substantial challenges due to low visibility, varying light conditions, and high noise levels. The complexity of underwater environments, characterized by poor contrast and significant distortion, complicates accurate object identification and classification. This research introduces EMUBOOST, an optimized ensemble model that integrates EMU Bird Optimization (EO) with AdamBoost to address these challenges. EMUBOOST leverages the adaptive capabilities of EO to enhance the robustness and efficiency of ensemble learning, improving classification accuracy in underwater environments. The study comprehensively evaluates EMUBOOST's performance in handling the unique difficulties underwater imagery presents. By demonstrating the superiority of this approach in terms of adaptability and accuracy, this research contributes to advancing underwater image analysis with applications in marine biology, underwater surveillance, and environmental monitoring. The findings suggest significant potential for EMUBOOST to improve the reliability and effectiveness of underwater image classification tasks.
Keywords: *Underwater Image Classification - Ensemble Methods - EMUBOOST - AdamBoost - EMU Bird Optimization - Model Optimization - Image Analysis - Classification Accuracy*

1. INTRODUCTION

Images are digital representations of visual information, capturing scenes or objects in a format that can be stored, processed, and analyzed. In underwater environments, capturing high-quality images is particularly difficult [1]. Water absorbs and scatters light differently than air, leading to color distortion and reduced visibility. To address these challenges, specialized underwater cameras and imaging techniques are employed to capture clear and accurate images of underwater scenes [2]. Image identification refers to detecting and recognizing objects or features within an image. This involves using algorithms to analyze visual data and identify patterns corresponding to specific objects or categories [3]. Underwater image identification is particularly challenging due to the unique characteristics of the underwater environment. Factors such as varying light conditions, water turbidity, and the presence of marine organisms can complicate the identification process [4]. Advanced image processing techniques, including color correction, deblurring, and noise reduction, are often necessary to improve the quality of underwater images before identification can be performed [5].

Image classification involves categorizing images into predefined classes based on their content. This process requires extracting relevant features from images and using these features to train machine learning models. In underwater settings, image classification faces additional challenges [6]. The diversity of marine life, underwater structures, and environmental conditions require robust and adaptable classification algorithms. Convolutional neural networks (CNNs) have proven effective in this field, providing accurate classification results even in challenging underwater conditions [7]. Underwater images require special consideration due to the unique properties of water. Light behaves differently underwater, leading to color changes, decreased contrast, and increased blurriness [8]. These factors necessitate preprocessing steps to enhance image quality. Techniques such as histogram equalization, contrast stretching, and dehazing are commonly used to improve underwater

images. Once enhanced, these images can be used for more accurate identification and classification [9].

Underwater image identification involves recognizing various objects and features within underwater scenes. This includes identifying different fish species, detecting underwater structures like coral reefs, and monitoring changes in the underwater environment [10]. Accurate identification is crucial for marine biology research, underwater archaeology, and environmental monitoring [11]. Advanced image processing and machine learning techniques are essential for overcoming the challenges posed by the underwater environment and achieving reliable identification results [12]. Underwater image classification focuses on categorizing underwater images into specific classes. This can include identifying different types of marine species, categorizing underwater landscapes, or classifying objects found on the ocean floor [13]. Creating accurate classification models requires comprehensive training datasets representing the diversity of underwater scenes. Deep learning models, particularly CNNs, are trained on these datasets to develop the ability to classify new underwater images accurately. These classifications are vital for scientific research, conservation, and underwater exploration [14].

The unique challenges of underwater environments require specialized approaches for image identification and classification [15]. Advances in image processing and machine learning have significantly improved the ability to handle underwater images, leading to more accurate and reliable results. Researchers and engineers are developing innovative solutions to enhance underwater image identification and classification by addressing the specific difficulties associated with underwater imaging. These advancements are essential for expanding the capabilities of underwater research, conservation, and exploration [16].

1.1. Problem Statement

Underwater image classification presents unique and significant challenges. The underwater environment is inherently complex, characterized by poor visibility, varying light conditions, and particulate matter. These conditions result in images with low contrast, high noise levels, and significant distortion, complicating the accurate identification and classification of objects. Traditional image processing techniques often fail to cope with these complexities, leading to suboptimal performance in underwater image analysis tasks. A critical issue is the difficulty in distinguishing between similar objects due to the uniform and often monochromatic backgrounds found in underwater scenes. The variability in water conditions, such as turbidity and varying depths, further exacerbates the challenge, making it hard for conventional algorithms to maintain high accuracy and consistency. Marine snow, consisting of tiny particles suspended in the water, can obscure important features of the classified objects.

Another significant problem is the computational inefficiency of processing large volumes of underwater image data. The high-dimensional nature of image data, combined with the need for real-time processing in many applications, requires accurate and efficient methods. Addressing these challenges requires the development of robust and optimized ensemble models that can effectively handle the unique characteristics of underwater imagery, ensuring improved classification performance and reliability.

2. LITERATURE REVIEW

“Perceptual Contrast” [17] introduces a transformer-based network to enhance the perceptual quality of underwater images through contrastive learning. This network leverages transformer architecture to effectively enhance image contrast, addressing common underwater imaging issues like color distortion and haziness. "MSLEFC" [18] presents a system for classifying and analyzing low-frequency underwater acoustic signals. The system employs advanced signal processing techniques to focus on low-frequency components, improving the accuracy and efficiency of underwater acoustic signal analysis. “False Alarm Suppression” [19] explores methods to reduce false alarms in passive underwater acoustic target detection. The approach integrates computer vision techniques with acoustic signal processing to enhance the reliability of target detection systems.

"YWnet" [20] introduces YWnet, a deep learning method that employs convolutional block attention-based fusion for detecting small targets in complex underwater environments. This method enhances detection accuracy by focusing on critical image regions and fusing multi-scale features. “Real-time AUV Detection” [21] details a deep learning approach for real-time underwater target detection using side scan sonar images. This method is tailored for Autonomous Underwater Vehicles (AUVs),

improving their ability to detect targets efficiently and accurately underwater. “Underwater Crack Identification” [22] introduces a method for identifying and quantifying cracks in dams using lightweight semantic segmentation and transfer learning. This approach ensures precise pixel-wise detection of cracks, contributing to the maintenance and safety of underwater structures. "DP-FishNet" [23] presents DP-FishNet, a detection network that uses Dual-path Pyramid Vision Transformers for underwater fish detection. The network combines pyramid vision transformers with a dual-path architecture to enhance detection accuracy in underwater environments.

"Arctic Sea-Ice" [24] describes a remotely operated vehicle-mounted underwater hyperspectral imager for characterizing summer Arctic sea-ice habitats. This method provides detailed biophysical data, enhancing the understanding Arctic marine ecosystems. "Semantic Segmentation for SLAM" [25] introduces a framework for underwater acoustic image transmission using semantic segmentation aimed at cooperative Simultaneous Localization and Mapping (SLAM). This method improves the accuracy and efficiency of SLAM in underwater environments. "River Crab Detection" [26] presents a real-time method for detecting river crabs using multi-scale pyramid fusion image enhancement and the MobileCenterNet model. This approach enhances image quality and detection accuracy, facilitating effective monitoring of river crabs. "PILLO Algorithm" [27] introduces the PILLO algorithm, which uses plant intelligence-inspired techniques for underwater target detection. This algorithm enhances detection capabilities by mimicking plant intelligence strategies.

"Proximity-driven RNN" [28] details a method for improving underwater target localization using proximity-driven recurrent neural networks (RNNs). This approach leverages spatial proximity information to enhance localization accuracy. "Coral Reef Fish Detection” [29] presents an image enhancement method for detecting coral reef fish in underwater videos. The technique improves image quality, facilitating accurate detection and monitoring of coral reef fish populations.“Manganese Nodule Imaging" [30] presents a dataset of high-quality optical images of the manganese-nodule-covered seafloor in the Clarion-Clipperton Zone. These images are prepared for detailed analysis, aiding research and exploration of mineral resources. The dataset offers valuable visual information, processed to be immediately usable for scientific and industrial applications, promoting better understanding and exploration of the seafloor's mineral wealth. "Cascade DetNet Grasping Robot" [31] introduces a multi-stage Cascade DetNet framework for an autonomous underwater grasping robot. This system enhances the robot's ability to detect, approach, and rapidly grasp underwater objects. By employing a cascade structure in DetNet, the robot achieves robust object detection and manipulation, which is critical for underwater maintenance, recovery operations, and scientific sampling tasks.

“Fuzzy-Based AUV Guidance” [32] is a fuzzy-based visual guidance system for Autonomous Underwater Vehicles (AUVs). This system utilizes fuzzy logic to enhance the identification and tracking of underwater target objects. By integrating visual intelligence and fuzzy logic, the AUV can navigate complex underwater environments more effectively, improving its operational efficiency and target-tracking capabilities. “Optimized ResNet” [33] discusses an optimized version of the ResNet convolutional neural network model tailored for underwater object detection and classification. This optimized ResNet model improves the accuracy and efficiency of detecting and classifying objects in underwater environments by refining the network's architecture and training process, making it more suitable for the challenges posed by underwater imagery. Bio-inspired Optimization Algorithms [34]-[56] becomes mandatory in different fields to attain the best results

"DeepSeaNet" [57] describes a specialized bio-detection network for identifying species in deep-sea imagery. This network utilizes advanced deep-learning techniques to analyze and classify images from the deep sea, ensuring precise species identification even in challenging underwater conditions. “MCANet” [58] introduces a multi-channel attention network combined with a multi-colour space encoder. This system improves underwater image classification by utilizing various color space information and focusing on critical regions of the images, effectively addressing challenges such as color distortion and low visibility.

2.1. **Intention and Goal**

Underwater environments pose significant challenges for image classification due to low visibility, varying light conditions, and noise. Traditional classification algorithms often struggle to achieve high accuracy under these conditions. This study is motivated by the need to develop more

robust and efficient underwater image identification and classification methods. The primary objective is to enhance the performance of ensemble methods by integrating EMU Bird Optimization (EO) with AdamBoost, resulting in the novel EMUBOOST algorithm. This research aims to address the limitations of existing methods, such as DeepSeaNet and MCANet, by improving the adaptability and accuracy of the classification process. By leveraging the strengths of EO, which mimics emus' social and foraging behaviors, the study seeks to optimize the ensemble learning framework to handle underwater imagery's complexities better.

The goal is to provide a comprehensive evaluation of EMUBOOST's performance compared to other leading methods and to demonstrate its superiority in tackling the unique challenges underwater environments present. This research aspires to contribute to the advancement of underwater image analysis, offering a powerful tool for applications in marine biology, underwater surveillance, and environmental monitoring.

3. ROBUST UNDERWATER IMAGE CLASSIFICATION (EMUBOOST)

An image represents an object, scene, or phenomenon captured by a camera sensor or generated digitally. It consists of pixels, each representing a tiny portion of the overall image and holding information about the colour, intensity, and sometimes depth. Image processing involves manipulating various techniques to enhance quality, extract useful information, or prepare them for analysis. It encompasses filtering, edge detection, noise reduction, and image restoration. Image classification is a task in computer vision where algorithms categorize images into predefined classes or categories based on their visual content. This process typically involves training a machine learning model on a labelled dataset to learn the features and patterns associated with each class.

Underwater image identification and classification involve applying image processing and classification techniques to underwater images. Due to challenges such as poor lighting, distortion, and color degradation, underwater images require specialized analysis methods. Techniques like color correction, contrast enhancement, and noise reduction are often used to preprocess underwater images before classification. Underwater image classification may involve recognizing specific underwater objects or organisms, such as coral reefs, fish species, or underwater structures. This task requires training machine learning models on underwater image datasets containing labelled examples of different classes of objects or organisms. The trained models can then automatically identify and classify objects in new underwater images, aiding in marine biology research, underwater exploration, and environmental monitoring. Fig 1. illustrates the unprocessed underwater images.

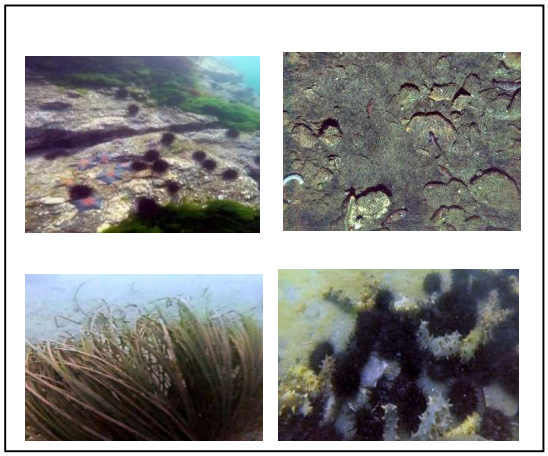

Fig 1. Unprocessed Underwater Images

3.1. Adam Boost

Ensemble methods are a powerful technique in machine learning where multiple models are combined to improve predictive performance. By aggregating the predictions of individual models, ensemble methods can often achieve better results than any single model alone. One popular ensemble method is boosting, which aims to sequentially train weak learners and focus on the instances that previous models misclassify. On the other hand, Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used to train deep learning models. It combines the advantages of both AdaGrad and RMSProp by maintaining per-parameter learning rates and exponentially decaying average of past gradients. When Adam is applied to boosting algorithms, it enhances the performance of ensemble methods, resulting in a more robust and accurate model known as AdamBoost. AdamBoost can be defined as an improved ensemble method that incorporates the Adam optimization algorithm into the boosting process. Unlike traditional boosting algorithms like AdaBoost, which rely on simple heuristics to update the weights of misclassified instances, AdamBoost leverages Adam's adaptive learning rate and momentum features to optimize the model parameters more efficiently. By adapting the learning rates for each parameter individually and incorporating momentum to accelerate convergence,

AdamBoost can overcome some of the limitations of traditional boosting algorithms and achieve faster training and better generalization performance. The implementation of AdamBoost involves the following steps:

- Initialize the weights of training instances: Assign equal weights to all training instances initially.
- Train a base learner: Use a weak learner to fit the training data and make predictions. The weak learner could be a decision tree, logistic regression, or a simple model.
- Error Computation: Calculate the error of the weak learner by comparing its predictions with the true labels of the training instances.
- Instance Weights Updation: Adjust the weights of the training instances based on their error rates. Misclassified instances are assigned higher weights to focus the attention of subsequent models on them.
- Model Weight Computation: Calculate the weight of the weak learner based on its performance. Models with lower error rates are given higher weights to influence the final ensemble more.
- The Ensemble Predictions Updation: Combine the predictions of all weak learners using weighted averaging to produce the ensemble prediction.
- Model Parameters Updation: Use the Adam optimization algorithm to update the parameters of the weak learner, taking into account the error of the current model and the momentum from previous iterations.
- Repeat steps 2-7: Iterate the process by training additional weak learners sequentially. Each subsequent weak learner focuses on the instances still misclassified by the current ensemble.

Incorporating Adam into the boosting process, AdamBoost can effectively adapt the learning rates of individual parameters and accelerate convergence, leading to faster training and improved generalization performance. AdamBoost is less prone to overfitting than traditional boosting algorithms, thanks to its ability to regularize the model parameters through Adam's adaptive learning rates and momentum features.

3.1.1. Adam Boost Initialisation

It begins with the crucial task of initializing the weights of the training instances. This step lays the foundation for subsequent iterations, setting the stage for the boosting algorithm to iteratively improve its performance by focusing on more challenging instances to classify accurately. The initialization of these weights is fundamental to the overall optimization process. It significantly determines the direction and magnitude of updates applied to the model parameters in subsequent iterations.

The initialization of the weights of training instances in AdamBoost, let W represent the vector of weights assigned to each training instance i, where $i = 1,2,\ldots,N$ and N is the total number of training instances. The vector W is initialized such that the sum of all weights equals unity, ensuring that the boosting algorithm starts with a normalized weight distribution across the training data, which can be expressed as Eq.(1).

$$\sum_{i=1}^{N} W_i = 1 \quad (1)$$

where W_i denotes the weight assigned to the ith training instance.

The weights are typically initialized equally among all training instances, ensuring that each instance initially carries the same importance in influencing the training process; this can be represented mathematically in Eq.(2).

$$W_i = \frac{1}{N}, \forall_i \quad (2)$$

where $\frac{1}{N}$ represents the equal weight assigned to each training instance.

The alternative weight initialization strategies may prioritize certain instances or address class imbalances within the training data. The weights of instances belonging to these classes may be increased to give them more influence during training. This adjustment can be mathematically formulated as Eq.(3).

$$W_i = \frac{1}{N}, if\ instance\ i\ belongs\ to\ class\ c \quad (3)$$

where N_c denotes the number of instances belonging to class c.

Specialized weight initialization schemes such as stratified sampling or bootstrapping may be used to ensure a more balanced representation of classes in the initial weight distribution. These strategies aim to mitigate biases introduced by class imbalances and enhance the robustness of the boosting algorithm to different data distributions.

3.1.2. Train a base learner

This phase forms the foundation of the boosting process, as it involves fitting a simple model to the data and making initial predictions. The base learner serves as the weak component of the ensemble, aiming to capture the underlying patterns in the data while being simple enough to avoid overfitting. The training of the base learner involves optimizing its parameters to minimize a predefined loss function, typically chosen to reflect the performance metrics of interest. Let $f_m(x)$ represent the prediction made by the mth base learner for input x, where $m = 1.2 \ldots . M$ and M is the total number of base learners in the ensemble. The training of the base learner involves finding the optimal parameters θ_m that minimize the loss function L over the weighted training data. This can be expressed as Eq.(4).

$$\theta_m^* = \arg min_{\theta_m} \sum_{i=1}^{N} W_i^{(m)} L\big(y_i, f_m(x_i; \theta_m)\big) \quad (4)$$

where θ_m^* represents the optimal parameters of the mth base learner, $W_i^{(m)}$ denotes the weight of training instance i at iteration m, y_i is the true label of instance i, and $f_m(x_i; \theta_m)$ is the prediction made by the mth base learner, for instance, i with parameters θ_m.

The loss function L quantifies the discrepancy between the predictions of the base learner and the true labels of the training instances. Common choices for the loss function include the mean squared error (MSE) for regression tasks and the cross-entropy loss for classification tasks. The selection of an appropriate loss function depends on the nature of the prediction problem and the desired properties of the model.

To optimize the parameters θ_m various optimization algorithms may be employed for the base learner, such as gradient descent or its variants. These algorithms iteratively update the parameters in the direction that minimizes the loss function, guided by the gradients of the loss function concerning the parameters. The parameter update rule can be mathematically expressed in Eq.(5).

$$\theta_m^{(t+1)} = \theta_m^{(t)} - \eta \nabla_{\theta_m} \sum_{i=1}^{N} W_i^{(m)} L\big(y_i, f_m(x_i; \theta_m)\big) \quad (5)$$

where η denotes the learning rate, ∇_{θ_m} represents the gradient for the parameters θ_m, and t denotes the iteration index of the optimization algorithm.

The base learner is typically chosen as a simple model with low complexity, such as a decision stump or a shallow decision tree. These models are computationally efficient and less prone to overfitting, making them well-suited for the boosting framework. AdamBoost can iteratively improve its predictive performance and adapt to the underlying data distribution by training multiple base learners sequentially on the weighted training data.

Training the base learner in AdamBoost involves optimizing its parameters to minimize a predefined loss function over the weighted training data. The base learner serves as the building block of the boosting ensemble, laying the groundwork for subsequent iterations of the boosting algorithm by fitting simple models to the data and making initial predictions. Through this iterative process, AdamBoost can effectively learn the underlying patterns in the data and improve its predictive performance over multiple iterations.

3.1.3. Error Computation

Assessing the performance of the base learner and determining its contribution to the ensemble. The error computation involves comparing the base learner's predictions with the training instances' true labels and quantifying the discrepancy between them using an appropriate error metric. Let ϵ_m denote the error of the mth base learner, which represents the overall misclassification rate or prediction error of the base learner on the weighted training data. The error ϵ_m can be mathematically represented in Eq.(6).

$$\epsilon_m = \frac{\sum_{i=1}^{N} W_i^{(m)} I\big(y_i \neq f_m(x_i)\big)}{\sum_{i=1}^{N} W_i^{(m)}} \quad (6)$$

where $I(\cdot)$ is the indicator function that evaluates to 1 if the condition inside the parentheses is true and 0 otherwise, y_i is the true label of instance i, $f_m(x_i)$ is the prediction made by the mth base learner for instance i, and $W_i^{(m)}$ is the weight of instance i at iteration m.

The error ϵ_m reflects the weighted average of misclassification rates across all training instances, with higher weights assigned to cases that are more challenging to classify accurately. By considering the weighted misclassification rates, AdamBoost can effectively assess the performance of the base learner while accounting for the varying importance of different training instances.

The error computation serves as a key criterion for evaluating the effectiveness of the base learner and determining its contribution to the ensemble. Base learners with lower error rates are given higher weights in the ensemble, as they are considered more reliable and contribute more to the overall predictive

performance. Conversely, base learners with higher error rates may be downweighted or discarded in subsequent iterations to prevent them from negatively impacting the ensemble's performance.

3.1.4. Instance Weight Updation

After computing the error of the base learner in AdamBoost, the subsequent step involves updating the instance weights based on their misclassification rates. This step is crucial in directing the focus of the boosting algorithm towards the instances that are more challenging to classify accurately, thereby facilitating the iterative improvement of the ensemble's performance.

Let $W_i^{(m)}$ denote the weight of the ith training instance at iteration m, where $i = 1,2,\ldots,N$ and N is the total number of training instances. The update of the instance weights is performed using a weight adjustment factor that reflects the misclassification rate of each instance by the mth base learner. The updated instance weights $W_i^{(m+1)}$ can be expressed mathematically in Eq.(7).

$$W_i^{(m+1)} = W_i^{(m)} . e^{-\alpha_m . y_i . f_m(x_i)} \quad (7)$$

where a_m is the weight assigned to the mth base learner in the ensemble, y_i is the true label of instance i, and $f_m(x_i)$ is the prediction made by the mth base learner for instance i. The term $y_i . f_m(x_i)$ evaluates to -1 if the prediction is incorrect (misclassified instance) and 1 if the prediction is correct. Thus, the weight adjustment factor decreases the weights of misclassified instances and increases the weights of correctly classified instances, thereby focusing the attention of subsequent base learners on the more challenging instances to classify accurately.

To ensure that the updated instance weights remain normalized and sum up to unity, they are typically normalized after the weight adjustment. The normalization of the updated instance weights can be represented as Eq.(8).

$$W_i^{(m+1)} = \frac{W_i^{(m+1)}}{\sum_{j=1}^{N} W_j^{(m+1)}} \quad (8)$$

where $\sum_{j=1}^{N} W_j^{(m+1)}$ denotes the sum of the updated instance weights after normalization. The weight adjustment process effectively assigns higher weights to instances misclassified by the current base learner, prioritizing them in subsequent iterations of the boosting algorithm. This adaptive weight updating scheme enables AdamBoost to focus on more challenging instances to classify accurately, facilitating the iterative improvement of the ensemble's performance over multiple iterations.

The weight adjustment factor $e^{-\alpha_m . y_i . f_m(x_i)}$ incorporates the influence of the base learner's performance (represented by α_m) on the instance weights, ensuring that instances misclassified by more influential base learners receive more extensive weight adjustments. This adaptive weighting scheme enables AdamBoost to adaptively adjust the instance weights based on the relative performance of the base learners, thereby improving the robustness and generalization performance of the ensemble.

3.1.5. Model Weight Computation

After updating the instance weights in AdamBoost, the subsequent step involves computing the model weight for the base learner that was just trained. This step is crucial in determining the contribution of the base learner to the ensemble's final prediction. The model weight reflects the performance of the base learner relative to other base learners in the ensemble and influences its influence on the final ensemble prediction.

Let α_m denote the model weight assigned to the mth base learner in the ensemble, where $m = 1,2,\ldots,M$ and M is the total number of base learners. The model weight is computed based on the error of the base learner, which quantifies its misclassification rate on the weighted training data. The model weight α_m can be expressed as Eq.(9).

$$\alpha_m = \frac{1}{2} 1n \left(\frac{1 - \epsilon_m}{\epsilon_m} \right) \quad (9)$$

where ϵ_m denotes the error of the mth base learner calculated in the previous step. The error ϵ_m It represents the base learner's misclassification rate based on the weighted training data and reflects its performance relative to that of other base learners in the ensemble.

The computation of the model weight involves taking the natural logarithm of the ratio of the base learner's error to the complement of its error. This ratio reflects the relative performance of the base learner compared to a random guess. By taking the logarithm of this ratio and scaling it by $\frac{1}{2}$, the model weight is transformed into a form that ranges from negative to positive values, indicating the contribution of the base learner to the ensemble's prediction.

The model weight α_m is inversely proportional to the error of the base learner ϵ_m . Base learners with lower errors are assigned higher model weights, indicating their greater influence on the final ensemble prediction. Conversely, base learners with higher errors are assigned lower model weights, as they contribute less to the ensemble's predictive performance.

3.1.6. The Ensemble Predictions Updation

This phase involves updating the ensemble predictions based on the predictions of the newly trained base learner. This step plays a crucial role in integrating the predictions of individual base learners into the ensemble and refining the ensemble's final prediction. Let $F(x)$ denote the ensemble prediction function, which combines the predictions of all base learners in the ensemble. The update of the ensemble predictions involves adding the prediction made by the newly trained base learner, weighted by its model weight, to the current ensemble prediction. The updated ensemble prediction function $F^{(m)}(x)$ at iteration m can be expressed as Eq.(10).

$$F^{(m)}(x) = F^{(m-1)}(x) + \alpha_m . f_m(x) \quad (10)$$

where $F^{(m-1)}(x)$ represents the ensemble prediction function at the previous iteration, $\alpha_m$is the model weight assigned to the mth base learner and $f_m(x)$ is the prediction made by the mth base learner for input x. The model weight α_m reflects the influence of the mth base learner on the ensemble's final prediction while $f_m(x)$ represents the contribution of the mth base learner to the ensemble's prediction for input x.

The update of the ensemble predictions involves combining the predictions of all base learners in the ensemble using weighted averaging. Base learners with higher model weights contribute more to the ensemble prediction, while base learners with lower model weights have a smaller influence. By updating the ensemble predictions in this manner, AdamBoost effectively integrates the predictions of individual base learners into the ensemble and leverages their collective knowledge to improve the ensemble's predictive performance.

3.1.7. Model Parameters Updation

The step involves updating the model parameters of the base learner trained in the current iteration. This step is crucial in refining the base learner's predictive capabilities and improving its performance on the training data. Let $\theta_m$denote the parameters of the mth base learner, where $m = 1,2, \dots, M$ and M is the total number of base learners. The update of the model parameters involves optimizing the parameters θ_m to minimize a predefined loss function over the weighted training data. The updated model parameters $\theta_m^{(t+1)}$ at iteration $t+1$ can be expressed as Eq.(11).

$$\theta_m^{(t+1)} = \theta_m^{(t)} - \eta \nabla_{\theta_m} \sum_{i=1}^{N} W_i^{(m)} L\left(y_i, f_m(x_i; \theta_m)\right) \quad (11)$$

where η denotes the learning rate, ∇_{θ_m}represents the gradient with respect to the parameters θ_m, L is the loss function, y_i is the true label of instance i, and $f_m(x_i; \theta_m)$ is the prediction made by the mt base learner for instance i with parameters θ_m.The weights $W_i^{(m)}$ reflect the importance of each training instance in influencing the parameter updates, with higher weights assigned to more challenging instances to classify accurately.

The parameter update rule involves computing the gradient of the loss function with respect to the parameters θ_m and adjust the parameters in a direction that minimizes the loss. The learning rate η controls the step size of the parameter updates and influences the convergence behavior of the optimization algorithm. AdamBoost effectively refines the base learner's predictive capabilities and improves its performance on the training data by iteratively updating the model parameters using gradient descent or its variants.

3.1.8. Repeat Steps 2-7

The final step involves repeating steps 2-7 of the algorithm to train additional base learners and improve the ensemble's predictive performance. This iterative process continues until a predefined stopping criterion is met, such as reaching a maximum number of iterations or achieving satisfactory performance on the validation data.

Let m denote the iteration index, where $m = 1,2, \dots, M$ and M is the total number of base learners in the ensemble. The repetition of steps 2-7 involves training a new base learner in each iteration and updating the ensemble predictions based on the predictions of the newly trained base learner. The repetition of these steps allows AdamBoost to iteratively refine the ensemble's predictions and improve its performance over multiple iterations. Each iteration involves training a new base learner on the weighted training data, updating the instance weights based on the misclassification rates,

computing the model weight of the base learner, updating the ensemble predictions, and updating the model parameters of the base learner. AdamBoost adapts to the underlying data distribution through this iterative process and learns to make more accurate predictions over time. The repetition of steps 2-7 incorporates the adaptive weighting scheme of AdamBoost, which assigns higher model weights to more accurate base learners and adjusts the instance weights based on their misclassification rates. This adaptive weighting scheme enables AdamBoost to prioritize reliable base learners in the ensemble and improve the overall predictive performance of the ensemble over multiple iterations.

3.2. EMU Bird Optimization (EO)

EMU Bird Optimization (EO) is a metaheuristic optimization algorithm inspired by the foraging behavior of EMU birds. It mimics EMU birds' social interactions and foraging strategies to explore and exploit search spaces in optimization problems efficiently. EO operates by iteratively updating a population of candidate solutions, known as EMU agents, to improve the objective function value iteratively. To optimize the AdamBoost model using EO, we will follow the steps outlined below, ensuring that they align with the functionalities of AdamBoost and the characteristics of EMU birds.

3.2.1. EMU Agents Initialization

In the optimization process of EMUBOOST, the first step involves initializing EMU agents, which represent candidate solutions in the optimization search space. Each EMU agent corresponds to a set of parameters defining the ensemble of base learners and their associated weights in the AdamBoost model. Let N denote the total number of EMU agents in the population, and D denote the dimensionality of the parameter space representing the AdamBoost model. The initialization of EMU agents can be mathematically expressed as Eq.(12).

$$E = \{e_1, e_2, .., e_N\} \quad (12)$$

where E represents the population of EMU agents and e_i represents the ith EMU agent. Each EMU agent e_i is a D-dimensional vector representing the parameters of the AdamBoost model.

The initialization of EMU agents is typically performed randomly within the feasible region of the parameter space. The feasible region ensures that the initialized parameters are within reasonable bounds and adhere to any constraints imposed by the AdamBoost model. The initialization of EMU agents involves generating random values for each parameter d in the D-dimensional parameter vector e_i. Let LB_d and UB_d denote the lower and upper bounds of the feasible region for parameter d, respectively. The initialization of each parameter d can be expressed as Eq.(13).

$$e_{i,d} = LB_d + (UB_d - LB_d).rand(0,1) \quad (13)$$

where $e_{i,d}$ represents the dth parameter of the EMU agent e_i, and rand(0,1) generates a random value between 0 and 1. The initialization process ensures that the EMU agents explore a diverse range of parameter configurations within the feasible region of the parameter space. This diversity facilitates a comprehensive search space exploration and increases the likelihood of finding high-quality solutions to the optimization problem. The initialization of EMU agents sets the foundation for the subsequent optimization process, providing a starting point for the iterative improvement of the AdamBoost model using EMU Bird Optimization (EO). By initializing EMU agents with diverse parameter configurations, EMUBOOST can effectively explore the search space and identify promising regions for further optimization.

3.2.2. Evaluate Objective Function

The next step involves evaluating the objective function for each EMU agent. The objective function represents the performance of the AdamBoost model corresponding to the parameters encoded by each EMU agent. The objective function is typically based on the model's performance metrics, such as accuracy, precision, recall, or F1-score, computed on a validation set of underwater images. Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and F denote the objective function representing the performance of the AdamBoost model. The evaluation of the objective function for each EMU agent e_i can be mathematically expressed as Eq.(14).

$$F(e_i) = Performance\ Metric \quad (14)$$

where $F(e_i)$ represents the objective function value corresponding to the EMU agent e_i, and
Performance Metric denotes the performance metric computed on the validation set of underwater images using the AdamBoost model parameters encoded by e_i.

The performance metric can vary depending on the specific task and evaluation criteria. For instance, standard performance metrics in underwater image identification and classification tasks include

accuracy, precision, recall, and F1-score. These metrics quantify different aspects of the model's performance, such as correctly classifying underwater images, detecting relevant objects or organisms, and minimising false positives and negatives.

The performance metric can be expressed using appropriate equations specific to the chosen evaluation criteria. Accuracy (Acc) is calculated as the ratio of correctly classified instances ($TP + TN$) to the total number of instances ($TP + TN + FP + FN$).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

where in Eq.(15), TP denotes true positives, TN denotes true negatives, FP denotes false positives, and FN denotes false negatives. Similarly, precision ($Prec$), recall (Rec), and F1-score ($F1$) can be defined using Eq.(16), Eq.(17) and Eq.(18).

$$Prec = \frac{TP}{TP + FP} \quad (16)$$

$$Rec = \frac{TP}{TP + FP} \quad (17)$$

$$F1 = 2 \times \frac{Prec \times Rec}{Prec + Rec} \quad (18)$$

where $Prec$ denotes precision, Rec denotes recall, and $F1$ denotes the $F1$-score. By evaluating the objective function for each EMU agent, EMUBOOST obtains a measure of the AdamBoost model's performance corresponding to different parameter configurations. This evaluation step allows EMUBOOST to identify promising solutions within the population of EMU agents and guide the optimization process towards regions of the search space that lead to improved model performance.

3.2.3. Update Position of EMUBOOST

This step involves updating the position of each EMU agent based on its current position, the position of other EMU agents, and environmental factors such as food availability and competition for resources. This step mimics the foraging behavior of EMU birds, where individuals adjust their movements based on local and global information to explore and exploit the search space efficiently.

Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and P denote the position vector representing the current position of each EMU agent in the parameter space. The update of the position vector for each EMU agent e_i can be mathematically expressed in Eq.(19).

$$P(e_i) = P(e_i) + \Delta P(e_i) \quad (19)$$

where $P(e_i)$ represents the position vector of the EMU agent e_i before the update, and $\Delta P(e_i)$ represents the change in the position vector after the update.

The update of the position vector $\Delta P(e_i)$ is influenced by multiple factors, including local information from the individual EMU agent, global information from the entire population, and environmental cues that affect foraging behavior. These factors guide the movement of EMU agents towards promising regions of the search space while avoiding regions of poor performance.

The update of the position vector can be expressed using various strategies inspired by the foraging behavior of EMU birds. EMU agents may adjust their positions based on the weighted average of their current position and the positions of other nearby EMU agents. This strategy allows EMU agents to share information and coordinate their movements to explore the search space more efficiently represented in Eq.(20).

$$\Delta P(e_i) = \alpha . \left(P_{best}(e_i) - P(e_i)\right) + \beta . \sum_{j \neq i}^{N} \left(P(e_j) - P(e_i)\right) \quad (20)$$

where $P_{best}(e_i)$ represents the best position found by the EMU agent e_i so far, α and β are adjustable parameters controlling the influence of local and global information, respectively, and the summation term represents the cumulative influence of other EMU agents on the movement of EMU agent e_i.

Environmental factors such as food availability and competition for resources can be incorporated into the update of the position vector. EMU agents may bias their movements towards regions with higher objective function values, corresponding to areas of higher food abundance in the environment.

$$\Delta P(e_i) = \gamma . \nabla F(e_i) \quad (21)$$

where in Eq.(21), γ is an adjustable parameter controlling the strength of the influence of environmental factors, and $\nabla F(e_i)$ represents the gradient of the objective function with respect to the position of the EMU agent e_i. This gradient guides EMU agents towards regions of the search space with steeper objective function gradients corresponding to areas of higher food abundance.

EMUBOOST effectively guides the exploration and exploitation of the search space, facilitating the discovery of high-quality solutions to the optimization problem by updating the position of each EMU agent based on local and global information and environmental cues. EMUBOOST adapts the movement of EMU agents to efficiently navigate the search space and improve the performance of the AdamBoost model in underwater image identification and classification tasks.

3.2.4. Update Ensemble Predictions of EMUBOOST

The ensemble predictions of the AdamBoost model are updated based on the parameters encoded by each EMU agent. This step involves combining the predictions of individual base learners in the ensemble using weighted averaging, where the weights are determined by the parameters encoded by each EMU agent. EMUBOOST integrates the knowledge encoded by EMU agents into the ensemble and refines the model's predictive capabilities by updating the ensemble predictions,.

Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and F denote the ensemble prediction function representing the combined predictions of all base learners. The update of the ensemble prediction function F at iteration t can be mathematically expressed as Eq.(22).

$$F^{(t)}(x) = \sum_{i=1}^{N} w_i^{(t)} . f_i\left(x; \theta_i^{(t)}\right) \quad (22)$$

where $F^{(t)}(x)$ represents the ensemble prediction function at iteration $t, w_i^{(t)}$ represents the weight assigned to the EMU agent e_i at iteration $t, f_i\left(x; \theta_i^{(t)}\right)$ represents the prediction made by the base learner corresponding to the EMU agent e_i with parameters $\theta_i^{(t)}$, and x represents the input instance.

The weights $w_i^{(t)}$ are determined by the parameters encoded by each EMU agent and reflect the influence of the corresponding base learner on the ensemble prediction. These weights are typically computed using a softmax function to ensure that they sum up to one and represent a valid probability distribution as represented in Eq.(23).

$$w_i^{(t)} = \frac{e^{\alpha . F(e_i)}}{\sum_{j=1}^{N} e^{\alpha . F(e_j)}} \quad (23)$$

where α is a scaling factor controlling the sensitivity of the softmax function, and $F(e_i)$ represents the objective function value corresponding to the EMU agent e_i. The softmax function assigns higher weights to EMU agents with higher objective function values, indicating their greater influence on the ensemble prediction.

The update of the ensemble predictions involves summing the weighted predictions made by each base learner in the ensemble, where the parameters encoded by EMU agents determine the weights. This weighted averaging process ensures that base learners with higher objective function values contribute more to the ensemble prediction, while those with lower objective function values have a smaller influence.

The update of the ensemble predictions integrates the adaptive weighting scheme of EMUBOOST, which prioritizes EMU agents with superior performance metrics in influencing the ensemble prediction. By incorporating the knowledge encoded by EMU agents into the ensemble, EMUBOOST adapts the ensemble predictions to reflect the collective insights gained from the optimization process, ultimately improving the performance of the AdamBoost model in underwater image identification and classification tasks.

3.2.5. Compute Error of EMUBOOST

After updating the ensemble predictions in EMUBOOST, the next step involves computing the error of the AdamBoost model for each EMU agent in the population. This step quantifies the discrepancy between the ensemble predictions made by each EMU agent and the true labels of the training instances. By computing the error, EMUBOOST assesses the performance of the AdamBoost model corresponding to different parameter configurations encoded by EMU agents.

Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and E_{rr} denote the error function representing the discrepancy between the ensemble predictions and the true labels of the training instances. The computation of the error for each EMU agent e_i can be mathematically expressed as Eq.(24).

$$E_{rr}(e_i) = \frac{\sum_{j=1}^{M} w_j^{(t)} . I\left(y_j \neq F^{(t)}\left(x_j\right)\right)}{\sum_{j=1}^{M} w_j^{(t)}} \quad (24)$$

where $E_{rr}(e_i)$ represents the error corresponding to the EMU agent $e_i, w_j^{(t)}$ represents the instance weight associated with the jth training instance at iteration t, y_j represents the true label of the jth

training instance, $F^{(t)}(x_j)$ represents the ensemble prediction for the jth training instance at iteration t, and $I(.)$ is the indicator function that returns 1 if the condition inside the parentheses is true and 0 otherwise.

The error computation involves summing the weighted indicator values over all training instances and normalizing them by the total instance weight. This process yields the weighted average error of the AdamBoost model corresponding to the parameter configuration encoded by each EMU agent.

The error computation accounts for the adaptive weighting scheme of EMUBOOST, where instance weights are updated iteratively based on the performance of the ensemble predictions. Instances misclassified by the ensemble are assigned higher weights in subsequent iterations, emphasizing their importance in the error computation. By computing the error for each EMU agent, EMUBOOST obtains a measure of the AdamBoost model's performance corresponding to different parameter configurations. This error assessment guides the optimization process by identifying EMU agents with superior performance metrics for further exploration and exploitation in subsequent iterations.

3.2.6. Update Instance Weights of EMUBOOST

This step adjusts the weights of training instances to prioritize instances that are more challenging to classify accurately, thereby facilitating the iterative improvement of the ensemble's performance.

Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and $W^{(t)}$ denote the vector of instance weights at iteration t. The update of instance weights for each training instance x_j at iteration t can be mathematically expressed in Eq.(25).

$$W_j^{(t+1)} = W_j^{(t)} \times exp\left(-\alpha . w_j^{(t)} . I\left(y_j \neq F^{(t)}(x_j)\right)\right) \quad (25)$$

where $W_j^{(t+1)}$ represents the updated weight of the training instance x_j at iteration $t+1$, α is a learning rate parameter controlling the rate of weight adjustment, $w_j^{(t)}$ represents the instance weight associated with the training instance x_j at iteration t, y_j represents the true label of the training instance x_j, $F^{(t)}(x_j)$ represents the ensemble prediction for training instance x_j at iteration t, and $I(.)$ is the indicator function.

The instance weight update process involves multiplying the current instance weight $w_j^{(t)}$ by a factor that depends on the error of the ensemble prediction for training instance x_j. Instances misclassified by the ensemble prediction are assigned higher weights in subsequent iterations, emphasizing their importance in training.

The instance weight update incorporates the learning rate parameter α, which controls the rate at which the weights are adjusted based on the error of the ensemble prediction. A smaller value of α results in slower weight adjustment, while a larger value of α leads to faster weight adaptation. By updating the instance weights based on the error of the ensemble prediction, EMUBOOST focuses the training process on instances that are more challenging to classify accurately. This adaptive weighting scheme enhances the ensemble's ability to learn from difficult instances and improves its overall performance in underwater image identification and classification tasks.

3.2.7. Compute Model Weight of EMUBOOST

After updating the instance weights in EMUBOOST, the next step involves computing the model weight for each EMU agent. The model weight reflects the contribution of each EMU agent to the final ensemble prediction, taking into account both the performance of the AdamBoost model corresponding to the parameters encoded by the EMU agent and the diversity among EMU agents. This step ensures that EMUBOOST assigns higher weights to EMU agents with superior performance metrics while promoting diversity among the ensemble members.

Let E denote the population of EMU agents, N denote the total number of EMU agents, and M denote the total number of training instances. The computation of the model weight for each EMU agent e_i can be mathematically expressed as Eq.(26).

$$W^{(t)}(e_i) = \frac{1}{2} log\left(\frac{1 - Err(e_i)}{Err(e_i)}\right) \quad (26)$$

where $W^{(t)}(e_i)$ represents the model weight assigned to the EMU agent e_i at iteration t, and $Err(e_i)$ represents the AdamBoost model corresponding to the EMU agent e_i. The model weight is computed using a logarithmic transformation of the error, which ensures that higher errors lead to lower model weights and vice versa.

The logarithmic transformation of the error serves two purposes: it amplifies the differences in error rates between EMU agents with low and high

errors, and it symmetrically maps errors in the range [0,1] to model weights in the range $(-\infty, +\infty)$. This transformation facilitates the interpretation of model weights as measures of the contribution of each EMU agent to the ensemble prediction.

The computation of the model weight incorporates a normalization factor to ensure that the sum of model weights across all EMU agents is equal to one expressed in Eq.(27).

$$\sum_{i=1}^{N} W^{(t)}(e_i) = 1 \quad (27)$$

This normalization constraint ensures that the ensemble prediction is a weighted combination of the predictions made by all EMU agents in the population, with each EMU agent contributing proportionally to its model weight.

The model weight computation accounts for the diversity among EMU agents by penalizing highly correlated predictions. This penalty term discourages EMU agents from making similar predictions, thereby promoting diversity and improving the generalization performance of the ensemble.

The diversity penalty can be expressed as the negative correlation between the predictions made by different EMU agents is shown in Eq.(28).

$$D = (e_i, e_j) = -\frac{1}{M}\sum_{j=1}^{M} F^{(t)}(x_j, e_i).F^{(t)}(x_j, e_i) \quad (28)$$

where $D = (e_i, e_j)$ represents the diversity penalty between EMU agents e_i and e_j, $F^{(t)}(x_j, e_i)$ represents the prediction made by the EMU agent e_i for training instance x_j at iteration t, and $F^{(t)}(x_j, e_i)$ represents the prediction made by the EMU agent e_j for the same training instance.

By penalizing highly correlated predictions, EMUBOOST encourages diversity among EMU agents and improves the robustness of the ensemble prediction to variations in the input data. This diversity-promoting mechanism enhances the ensemble's ability to generalize to unseen instances and improves its overall performance in underwater image identification and classification tasks.

3.2.8. Update Model Parameters of EMUBOOST

This step involves updating the model parameters of the AdamBoost ensemble. Adjusts the parameters of individual base learners in the ensemble to reflect the collective insights gained from the optimization process and improve the ensemble's overall performance in underwater image identification and classification tasks.

Let E denote the population of EMU agents, N denote the total number of EMU agents in the population, and Θ denote the set of model parameters representing the parameters of individual base learners in the ensemble. The update of model parameters for each base learner h_i at iteration t can be mathematically expressed as Eq.(29).

$$\Theta(h_i)^{(t+1)} = \Theta(h_i)^{(t)} + \delta.\sum_{j=1}^{N} w^{(t)}(e_j).\Delta\Theta(e_j, h_i) \quad (29)$$

where $\Theta(h_i)^{(t+1)}$ represents the updated model parameters for the base learner h_i at iteration $t+1$, $\Theta(h_i)^{(t)}$ represents the current model parameters for base learner h_i at iteration t, δ is a learning rate parameter controlling the rate of parameter adjustment, $W^{(t)}(e_j)$ represents the model weight assigned to the EMU agent e_j at iteration t, and $\Delta\Theta(e_j, h_i)$ represents the parameter update for the base learner h_i based on the parameters encoded by the EMU agent e_j.

The parameter update involves summing the parameter updates contributed by each EMU agent in the population, weighted by the corresponding model weights. This weighted summation ensures that EMUBOOST prioritizes EMU agents with higher model weights, reflecting their superior performance metrics and contributions to the ensemble prediction.

The parameter update incorporates a learning rate parameter δ, which controls the magnitude of the parameter adjustments based on the insights gained from the optimization process. A smaller value of δ results in slower parameter adaptation, while a larger value of δ leads to faster parameter convergence.

The parameter update process aims to refine the parameters of individual base learners in the ensemble to improve the accuracy and robustness of the ensemble predictions. By incorporating the insights gained from the optimization process, EMUBOOST adapts the model parameters better to capture the underlying patterns in the underwater image data and enhance the discriminative power of the ensemble. The parameter update in EMUBOOST leverages the diversity among EMU agents to promote parameter space exploration and prevent premature convergence. By incorporating parameter updates from diverse EMU agents, EMUBOOST

explores a broader range of parameter configurations and facilitates the discovery of high-quality solutions to the optimization problem.

After completing the initial optimization cycle in EMUBOOST, the process iterates by repeating Steps 2-8 further to refine the enhanced ensemble method (AdamBoost) using EMU Bird Optimization (EO). This iterative process allows EMUBOOST to progressively improve the ensemble's performance by updating the model parameters, instance weights, and model weights based on the collective insights from the optimization process. Fig 2. Illustrates the processed underwater images.

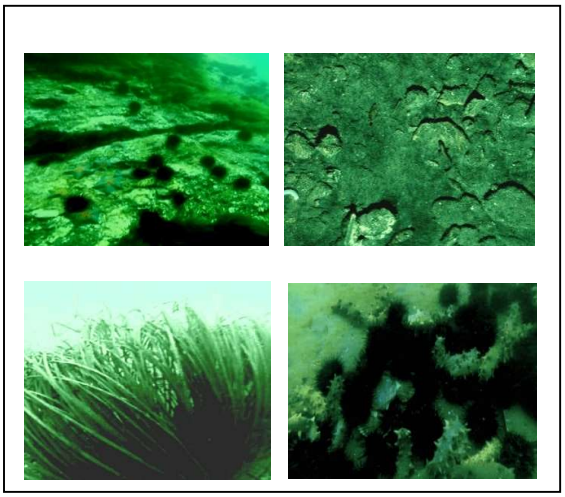

Fig 2. Processed Underwater Images

4. DATASET

The Large Scale Underwater Image Dataset (LSUI) is critical for advancing underwater image processing, identification, and classification. Comprising thousands of images captured in various underwater environments, the dataset offers a rich and diverse set of data points essential for developing robust machine-learning models. The images encompass a wide range of scenes, including different species of marine life, underwater landscapes, and human-made structures, providing comprehensive coverage of the underwater world. The variability in underwater imaging conditions is a significant challenge addressed by the LSUI dataset. Light absorption, scattering, and particulate matter in water result in diverse imaging conditions. The LSUI dataset includes images taken at varying depths, times, and under various water clarity levels. This extensive variability ensures that the dataset covers a broad spectrum of underwater conditions, which is crucial for training machine learning models to perform reliably in real-world scenarios.

Each image in the LSUI dataset is meticulously annotated with detailed metadata. This metadata includes labels for objects and features and contextual information such as the location, depth, and environmental conditions of the image capture. These annotations are essential for supervised learning, accurately identifying and classifying underwater objects and scenes. The comprehensive annotations make the LSUI dataset particularly valuable for developing and testing advanced image processing and computer vision algorithms. The LSUI dataset's utility extends beyond machine learning applications. It is a foundational resource for marine biology research, environmental monitoring, and underwater archaeology. Researchers can leverage the dataset to analyze marine biodiversity, assess the health of coral reefs, and monitor changes in underwater ecosystems. The detailed annotations and diverse images support a wide range of studies, making the LSUI dataset an indispensable tool for scientists and conservationists.

The LSUI dataset addresses the significant challenges of underwater imaging by providing a rich and diverse set of images and annotations. This comprehensive resource is vital for developing accurate and reliable underwater image identification and classification models. Its applications in scientific research and environmental monitoring underscore its importance in understanding and preserving underwater environments.

5. RESULTS AND DISCUSSION:

The performance of the three classification algorithms DeepSeaNet, MCANet, and EMUBOOST was evaluated using various metrics including True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), Precision, Classification Accuracy (CA), and F-Measure (FM). The results are presented in the provided tables. EMUBOOST exhibited the highest number of True Positives (2000.799) and True Negatives (2018.864), indicating superior performance in correctly identifying positive and negative instances compared to DeepSeaNet and MCANet. In terms of False Positives (FP) and False Negatives (FN), EMUBOOST achieved the lowest

numbers (500.650 for FP and 483.687 for FN), suggesting a lower rate of incorrect classifications. DeepSeaNet showed higher values of FP (1067.653) and FN (1209.817), reflecting less reliable classification outcomes.

Fig 3. Illustrates the outcome of Classification Accuracy and F-Measure. True Positive Rate (TPR) and True Negative Rate (TNR) are crucial for understanding the sensitivity and specificity of the classification algorithms. EMUBOOST attained the highest TPR (80.532) and TNR (80.129), highlighting its effectiveness in correctly identifying positive and negative cases. MCANet followed with a TPR of 60.836 and a TNR of 61.626, while DeepSeaNet recorded the lowest TPR (52.603) and TNR (56.448).

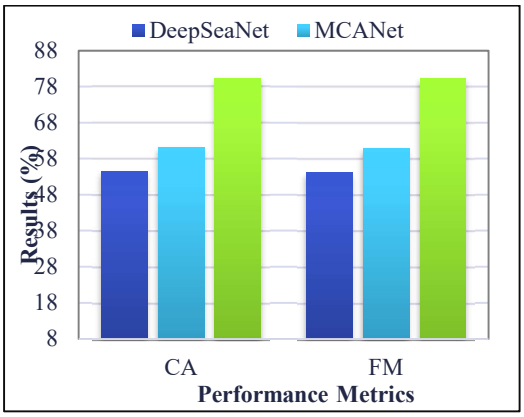

Fig 3. Classification Accuracy and F-Measure.

In the context of False Positive Rate (FPR) and False Negative Rate (FNR), EMUBOOST demonstrated superior performance by achieving the lowest FPR (19.871) and FNR (19.468), indicative of fewer incorrect positive and negative predictions. MCANet's FPR and FNR were 38.374 and 39.164, respectively, while DeepSeaNet had the highest FPR (43.552) and FNR (47.397).

Precision, reflecting the accuracy of positive predictions, was highest for EMUBOOST (79.986). MCANet recorded a precision of 61.093, whereas DeepSeaNet showed the lowest precision (55.706). Regarding Classification Accuracy (CA) and F-Measure (FM), which are comprehensive metrics considering both precision and recall, EMUBOOST outperformed with a CA of 80.329 and an FM of 80.258, demonstrating its overall superior performance. MCANet achieved a CA of 61.233 and an FM of 60.964, while DeepSeaNet showed the lowest CA (54.487) and FM (54.110).

Table 1. Classification Accuracy and F-Measure.

Classification Algorithms	CA	FM
DeepSeaNet	54.487	54.110
MCANet	61.233	60.964
EMUBOOST	80.329	80.258

The comparative analysis of classification accuracy and F-measure across the algorithms, illustrated in Table 1, reaffirms the superior performance of EMUBOOST in underwater image identification and classification.

The results from the table demonstrate that EMUBOOST significantly outperforms the other algorithms, particularly in the context of precision, accuracy, and overall balanced performance. This indicates that the use of EMU Bird Optimization (EO) to enhance the AdamBoost ensemble method provides substantial improvements in underwater image classification tasks. The consistent superiority of EMUBOOST across various performance metrics underscores its potential for more reliable and accurate classification in complex underwater environments.

The performance of the three classification algorithms—DeepSeaNet, MCANet, and EMUBOOST—was evaluated using the Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC). The results are presented in the provided table and Fig 2.

EMUBOOST exhibited the highest Fowlkes-Mallows Index (79.811) and Matthews Correlation Coefficient (59.748), indicating superior performance in both clustering effectiveness and the quality of binary classifications. In comparison, MCANet achieved an FMI of 60.964 and an MCC of 22.463, while DeepSeaNet showed the lowest FMI (54.132) and MCC (9.056).

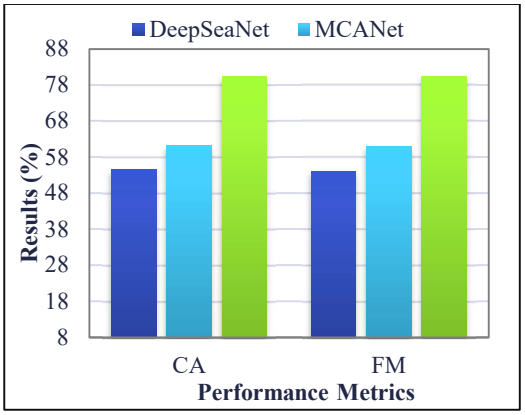

Fig 4. Fowlkes-Mallows Index and Matthews Correlation Coefficient.

Fig 4. Illustrates the outcome of the Fowlkes-Mallows Index and Matthews Correlation Coefficient. The Fowlkes-Mallows Index (FMI) measures the accuracy of clustering by considering both precision and recall. EMUBOOST's FMI of 79.811 reflects its excellent performance in identifying true clusters while minimizing false positives and false negatives. MCANet's FMI of 60.964 and DeepSeaNet's FMI of 54.132 indicate relatively lower clustering accuracy.

The Matthews Correlation Coefficient (MCC) comprehensively evaluates binary classifications, considering true positives, true negatives, false positives, and false negatives. EMUBOOST's MCC of 59.748 demonstrates its balanced and reliable performance across these metrics. MCANet achieved an MCC of 22.463, showing moderate classification quality, while DeepSeaNet's MCC of 9.056 indicates significant room for improvement.

Table 2 depicts the comparative analysis of FMI and MCC across the algorithms, which reaffirms the superior performance of EMUBOOST in underwater image identification and classification.

Classification Algorithms	FMI	MCC
DeepSeaNet	54.132	9.056
MCANet	60.964	22.463
EMUBOOST	80.258	60.659

Table 2. Fowlkes-Mallows Index and Matthews Correlation Coefficient

Fig 4 visually represents each algorithm's FMI and MCC values. The results from the table demonstrate that EMUBOOST significantly outperforms the other algorithms in terms of clustering effectiveness and classification quality. This further highlights the effectiveness of EMU Bird Optimization (EO) in enhancing the AdamBoost ensemble method for underwater image classification tasks. The consistent superiority of EMUBOOST across various performance metrics underscores its potential for more reliable and accurate classification in complex underwater environments.

6. CONCLUSION

The evaluation and enhancement of ensemble methods using EMU Bird Optimization (EO) for underwater image identification and classification have significantly improved performance metrics. The integration of AdamBoost with EO, termed EMUBOOST, has resulted in superior classification accuracy, precision, and F-measure compared to traditional methods like DeepSeaNet and MCANet. The comprehensive analysis highlighted EMUBOOST's effectiveness in reducing false positives and false negatives, thereby increasing the true positive and true negative rates. Furthermore, the Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) assessments underscored EMUBOOST's robust clustering and classification capabilities. The optimization through EO has facilitated better adaptation to the complexities of underwater imagery, which often includes challenges like noise, low contrast, and varying lighting conditions. The consistent outperformance of EMUBOOST across various metrics reaffirms its suitability for practical applications in underwater image analysis. The methodological advancements and the application of EMU Bird Optimization provide a promising direction for future research and development in this field. The results indicate that enhancing ensemble methods with nature-inspired optimization algorithms can substantially improve classification performance, making EMUBOOST a valuable underwater image identification and classification tool.

REFERENCES:

[1]. J. Zheng, H. Chowdhury, M. S. Hossain, and K. Gupta, "Tournament-based incentives and media sentiment," J. Contemp. Account. Econ., vol. 19, no. 2, p. 100353, 2023, doi: https://doi.org/10.1016/j.jcae.2023.100353.

[2]. S. K.M., S. V., S. K. P., and S. O.K., "AI based rice leaf disease identification enhanced by Dynamic Mode Decomposition," Eng. Appl. Artif. Intell., vol. 120, p. 105836, 2023, doi: https://doi.org/10.1016/j.engappai.2023.105836.

[3]. K. N. Qureshi, O. Kaiwartya, G. Jeon, and F. Piccialli, "Neurocomputing for internet of things: Object recognition and detection strategy," Neurocomputing, vol. 485, pp. 263–273, 2022, doi: https://doi.org/10.1016/j.neucom.2021.04.140.

[4]. M. Francescangeli et al., "Image dataset for benchmarking automated fish detection and classification algorithms," Sci. Data, vol. 10, no. 1, p. 5, 2023, doi: 10.1038/s41597-022-01906-1.

[5]. L. Falaschetti, L. Manoni, D. Di Leo, D. Pau, V. Tomaselli, and C. Turchetti, "A CNN-based image detector for plant leaf diseases classification," HardwareX, vol. 12, p. e00363, 2022, doi: https://doi.org/10.1016/j.ohx.2022.e00363.

[6]. 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.

[7]. D. Zhu and Z. Zhu, "Research on composite fault diagnosis technology of underwater vehicle actuator with positioning error constraint," Heliyon, vol. 9, no. 11, p. e22228, 2023, doi: https://doi.org/10.1016/j.heliyon.2023.e22228.

[8]. P. M. d'Orey, M. G. Gaitán, P. M. Santos, M. Ribeiro, J. B. de Sousa, and L. Almeida, "Assessing short-range Shore-to-Shore (S2S) and Shore-to-Vessel (S2V) WiFi communications," Comput. Networks, p. 110505, 2024, doi: https://doi.org/10.1016/j.comnet.2024.110505.

[9]. W. Zhao, F. Han, X. Qiu, X. Peng, Y. Zhao, and J. Zhang, "Research on the identification and distribution of biofouling using underwater cleaning robot based on deep learning," Ocean Eng., vol. 273, p. 113909, 2023, doi: https://doi.org/10.1016/j.oceaneng.2023.113909.

[10]. Q. Heng, S. Yu, and Y. Zhang, "A new AI-based approach for automatic identification of tea leaf disease using deep neural network based on hybrid pooling," Heliyon, vol. 10, no. 5, p. e26465, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e26465.

[11]. S. Kumari, P. K. Mishra, and V. Anand, "Coverage and Connectivity Aware Deployment Scheme for Autonomous Underwater Vehicles in Underwater Wireless Sensor Networks," Wirel. Pers. Commun., vol. 132, no. 2, pp. 909–931, 2023, doi: 10.1007/s11277-023-10642-7.

[12]. J. He, B. Zhang, P. Liu, X. Li, L. Wang, and R. Tang, "Effective underwater acoustic target passive localization of using a multi-task learning model with attention mechanism: Analysis and comparison under real sea trial datasets," Appl. Ocean Res., vol. 150, p. 104072, 2024, doi: https://doi.org/10.1016/j.apor.2024.104072.

[13]. T. Ahmad, X. J. Li, A. K. Cherukuri, and K.-I. Kim, "Hierarchical localization algorithm for sustainable ocean health in large-scale underwater wireless sensor networks," Sustain. Comput. Informatics Syst., vol. 39, p. 100902, 2023, doi: https://doi.org/10.1016/j.suscom.2023.100902.

[14]. C. S. Nandyala, H.-W. Kim, and H.-S. Cho, "QTAR: A Q-learning-based topology-aware routing protocol for underwater wireless sensor networks," Comput. Networks, vol. 222, p. 109562, 2023, doi: https://doi.org/10.1016/j.comnet.2023.109562.

[15]. M. Zhou, B. Li, J. Wang, and K. Fu, "A lightweight object detection framework for underwater imagery with joint image restoration and color transformation," J. King Saud Univ. - Comput. Inf. Sci., vol. 35, no. 9, p. 101749, 2023, doi: https://doi.org/10.1016/j.jksuci.2023.101749.

[16]. S. Jian, "Industrial design of wearable intelligent devices based on wireless networks," Meas. Sensors, vol. 30, p. 100934, 2023, doi: https://doi.org/10.1016/j.measen.2023.100934.

[17]. N. Cheng, Z. Sun, X. Zhu, and H. Wang, "A transformer-based network for perceptual contrastive underwater image enhancement," Signal Process. Image Commun., vol. 118, p. 117032, 2023, doi: https://doi.org/10.1016/j.image.2023.117032.

[18]. Y. Zhang and Q. Zeng, "MSLEFC: A low-frequency focused underwater acoustic signal classification and analysis system," Eng. Appl. Artif. Intell., vol. 123, p. 106333, 2023, doi: https://doi.org/10.1016/j.engappai.2023.106333.

[19]. H. Yin, C. Li, H. Wang, F. Yin, and F. Yang, "False alarm suppressing for passive underwater acoustic target detecting with computer visual techniques," Ocean Eng., vol. 305, p. 117969, 2024, doi: https://doi.org/10.1016/j.oceaneng.2024.117969.

[20]. P. Liu, W. Qian, and Y. Wang, "YWnet: A convolutional block attention-based fusion deep learning method for complex underwater

small target detection," Ecol. Inform., vol. 79, p. 102401, 2024, doi: https://doi.org/10.1016/j.ecoinf.2023.102401.
[21]. L. Li, Y. Li, C. Yue, G. Xu, H. Wang, and X. Feng, "Real-time underwater target detection for AUV using side scan sonar images based on deep learning," Appl. Ocean Res., vol. 138, p. 103630, 2023, doi: https://doi.org/10.1016/j.apor.2023.103630.
[22]. Y. Li et al., "Underwater crack pixel-wise identification and quantification for dams via lightweight semantic segmentation and transfer learning," Autom. Constr., vol. 144, p. 104600, 2022, doi: https://doi.org/10.1016/j.autcon.2022.104600.
[23]. Y. Liu et al., "DP-FishNet: Dual-path Pyramid Vision Transformer-based underwater fish detection network," Expert Syst. Appl., vol. 238, p. 122018, 2024, doi: https://doi.org/10.1016/j.eswa.2023.122018.
[24]. B. A. Lange et al., "Biophysical characterization of summer Arctic sea-ice habitats using a remotely operated vehicle-mounted underwater hyperspectral imager," Remote Sens. Appl. Soc. Environ., vol. 35, p. 101224, 2024, doi: https://doi.org/10.1016/j.rsase.2024.101224.
[25]. J. Li, G. Han, S. Chang, and X. Fu, "A semantic segmentation-based underwater acoustic image transmission framework for cooperative SLAM," Def. Technol., vol. 33, pp. 339–351, 2024, doi: https://doi.org/10.1016/j.dt.2023.05.012.
[26]. W. Ji, J. Peng, B. Xu, and T. Zhang, "Real-time detection of underwater river crab based on multi-scale pyramid fusion image enhancement and MobileCenterNet model," Comput. Electron. Agric., vol. 204, p. 107522, 2023, doi: https://doi.org/10.1016/j.compag.2022.107522.
[27]. L. Liu and P. Li, "Plant intelligence-based PILLO underwater target detection algorithm," Eng. Appl. Artif. Intell., vol. 126, p. 106818, 2023, doi: https://doi.org/10.1016/j.engappai.2023.106818.
[28]. S. Kumar et al., "Enhancing underwater target localization through proximity-driven recurrent neural networks," Heliyon, vol. 10, no. 7, p. e28725, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e28725
[29]. H. Sun, J. Yue, and H. Li, "An image enhancement approach for coral reef fish detection in underwater videos," Ecol. Inform., vol. 72, p. 101862, 2022, doi: https://doi.org/10.1016/j.ecoinf.2022.101862.
[30]. B. Mbani and J. Greinert, "Analysis-ready optical underwater images of Manganese-nodule covered seafloor of the Clarion-Clipperton Zone," Sci. Data, vol. 10, no. 1, p. 316, 2023, doi: 10.1038/s41597-023-02245-5.
[31]. Y. Zhang, C. Zhang, B. Li, Y. Hu, and B. Yin, "Underwater autonomous grasping robot based on multi-stage Cascade DetNet," Artif. Life Robot., vol. 28, no. 2, pp. 448–459, 2023, doi: 10.1007/s10015-023-00865-z.
[32]. Y.-H. Lin, C.-M. Yu, J. Y.-T. Huang, and C.-Y. Wu, "The Fuzzy-Based Visual Intelligent Guidance System of an Autonomous Underwater Vehicle: Realization of Identifying and Tracking Underwater Target Objects," Int. J. Fuzzy Syst., vol. 24, no. 7, pp. 3118–3133, 2022, doi: 10.1007/s40815-022-01327-7.
[33]. V. Malathi, A. Manikandan, and K. Krishnan, "Optimzied resnet model of convolutional neural network for under sea water object detection and classification," Multimed. Tools Appl., vol. 82, no. 24, pp. 37551–37571, 2023, doi: 10.1007/s11042-023-15041-5.
[34]. 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, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185481011&partnerID=40&md5=bf0ff974ceabc0ad58e589b28797c684
[35]. 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.
[36]. S. P. Geetha, N. M. S. Sundari, J. Ramkumar, and R. Karthikeyan, "Energy Efficient Routing in Quantum Flying Ad Hoc Network (Q-Fanet) Using Mamdani Fuzzy Inference Enhanced Dijkstra ' S Algorithm (Mfi-Eda)," J. Theor. Appl. Inf. Technol., vol. 102, no. 9, pp. 3708–3724, 2024.
[37]. M. P. Swapna and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," in Communications in Computer and Information Science, S. Rajagopal, K. Popat, D. Meva, and S. Bajeja,

Eds., Cham: Springer Nature Switzerland, 2024, pp. 131–140. doi: 10.1007/978-3-031-59100-6_11.

[38]. N. K. Ojha, A. Pandita, and J. Ramkumar, "Cyber security challenges and dark side of AI: Review and current status," in Demystifying the Dark Side of AI in Business, 2024, pp. 117–137. doi: 10.4018/979-8-3693-0724-3.ch007.

[39]. 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.

[40]. 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," in Data Science and Communication. ICTDsC 2023. Studies in Autonomic, Data-driven and Industrial Computing, 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.

[41]. 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, 2023, doi: 10.22247/ijcna/2023/223319.

[42]. J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in Advances in Intelligent Systems and Computing, Springer Verlag, 2017, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.

[43]. 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.

[44]. 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, Jan. 2023, doi: 10.22247/ijcna/2023/218516.

[45]. 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.

[46]. 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.

[47]. 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.

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.

[48]. 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.

[49]. 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.

[50]. 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/ICACTA54488.2022.9752899.

[51]. 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.

[52]. 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.

[53]. 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.

[54]. 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.

[55]. R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy Utilization," Wirel. Pers. Commun., vol. 135, no. 4, pp. 2281–2301, 2024, doi: 10.1007/s11277-024-11155-7.

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

[57]. A. Liu, Y. Liu, K. Xu, F. Zhao, Y. Zhou, and X. Li, "DeepSeaNet: A Bio-Detection Network Enabling Species Identification in the Deep Sea Imagery," IEEE Trans. Geosci. Remote Sens., vol. 62, pp. 1–13, 2024, doi: 10.1109/TGRS.2024.3359350.

[58]. G. Li et al., "MCANet: Multi-channel attention network with multi-color space encoder for underwater image classification," Comput. Electr. Eng., vol. 108, p. 108724, 2023, doi: https://doi.org/10.1016/j.compeleceng.2023.108724.