Journal of Theoretical and Applied Information Technology
31st January 2025. Vol.103. No.2
© Little Lion Scientific

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

RELIABLE UNDERWATER IMAGE CLASSIFICATION WITH ENHANCED CNN MODELS USING QUOKKA OPTIMIZATION

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

ABSTRACT

This article presents an innovative deep-learning model, QuokkaNet, for underwater image classification tasks. By optimizing the Enhanced CNN (N-CNN) model using Quokka Optimization (QO), QuokkaNet demonstrates significant advancements in performance. The N-CNN model incorporates the layers such as convolutional, pooling, normalization, fully connected, and Nesterov Accelerated Gradient, creating a robust framework for image classification. Quokka Optimization further enhances this model through systematic exploration, exploitation, fitness evaluation, selection, adaptation, and migration. The study highlights QuokkaNet's superior capabilities and effectiveness in underwater image classification compared to DeepSeaNet and MCANet. These findings confirm QuokkaNet's potential as a reliable and accurate underwater research and exploration tool, offering significant improvements in the classification of complex underwater images. The innovative approach and optimization techniques employed in QuokkaNet set a new benchmark for performance in this domain.

Keywords: *Underwater Image Classification, CNN, QuokkaNet, Nesterov Accelerated Gradient, Quokka Optimization, Image Analysis, Classification Accuracy, N-CNN*

1. INTRODUCTION

The underwater environment poses unique challenges for capturing and analyzing images due to factors such as light absorption, scattering, and the presence of marine particles [1]. These challenges necessitate specialized underwater image identification and classification techniques, which are critical for various scientific, environmental, and industrial applications [2].

Images are visual representations of objects or scenes captured and stored in digital formats for analysis. Underwater images, however, are subject to the complexities of the underwater environment [3]. Water absorbs light, especially at longer wavelengths, leading to significant color distortion. Particles in the water scatter light, reducing image clarity and contrast. Capturing high-quality underwater images requires specialized equipment and techniques to mitigate these issues [4]. Image identification is recognizing and labeling objects or features within an image. This involves using algorithms to detect patterns and match them to known categories. The identification process is particularly challenging for underwater images due to the environmental factors that affect image quality [5]. Techniques such as color correction, dehazing, and noise reduction are essential for improving the visibility and accuracy of underwater image identification. Advanced machine learning algorithms play a crucial role in accurately processing these enhanced images to identify various objects and features [6].

Image classification involves categorizing images into predefined groups based on their content. This requires extracting significant features from the images and training machine learning models on these features. Underwater image classification presents additional challenges compared to terrestrial image classification [6]. The diversity of marine life and underwater environments and the varying conditions under which images are captured necessitate robust and adaptive classification techniques. Convolutional neural networks (CNNs) have shown significant success in this domain, providing reliable classification results for underwater images. Image identification and classification involve recognizing and categorizing objects or patterns within an image [7].

Machine learning and deep learning methods, especially CNNs, are usually used in these procedures. The steps include:

- *Feature Extraction:* Identifying essential features or attributes within an image.
- *Training:* Using labeled datasets to teach a model to recognize patterns and objects.
- *Prediction:* Applying the trained model to classify new images into predefined categories.

Underwater images differ fundamentally from those captured in terrestrial environments [8]. The underwater setting introduces unique factors that affect image quality, such as light attenuation, color shifting, and suspended particles. These factors necessitate preprocessing steps to enhance the quality of underwater images [9]. Methods such as histogram equalization, adaptive contrast enhancement, and color restoration are commonly used to improve the visibility and detail of underwater images, making them suitable for subsequent identification and classification tasks [10].

Underwater image identification is essential for recognizing and monitoring marine species, detecting underwater structures, and assessing environmental changes. Accurate identification is vital for applications in marine biology, underwater archaeology, and environmental monitoring [11]. Advanced image processing and machine learning techniques enable the identification of various objects and features in underwater images, overcoming the challenges posed by the underwater environment. Underwater image classification focuses on categorizing images into specific classes, such as different species of fish, types of coral reefs, or various underwater landscapes [12]. Developing accurate classification models requires comprehensive training datasets that capture the diversity of underwater scenes. Deep learning models, particularly CNNs, are trained on these datasets to develop the ability to classify new images accurately [13]. These classification systems are crucial for scientific research, conservation efforts, and industrial applications such as underwater exploration and resource management. Underwater image classification and identification present unique challenges due to poor visibility, color distortion, and light absorption [14]. Key steps in this process include:

- Preprocessing: Correcting distortions and enhancing image quality through color correction and noise reduction.
- Segmentation: Isolating objects or regions of interest from the background may be cluttered with marine life or sediments.
- Feature Extraction: Identifying unique features of underwater objects, which can be challenging due to varying lighting conditions and object appearances.
- Classification: Using trained models to categorize objects, such as different fish species, corals, or underwater structures.

The challenges associated with underwater imaging require innovative 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 precise and reliable results [16]. Researchers and engineers are developing effective methods for underwater image identification and classification by addressing the specific difficulties of underwater environments. These innovations are essential for advancing the capabilities of underwater research, conservation, and exploration technologies [17]

1.1. Problem Statement

Underwater image classification poses significant challenges due to the unique and harsh environment found beneath the surface. Variability in lighting conditions, presence of particulate matter, and limited visibility complicate the accurate classification of underwater images. These factors often lead to poor image quality, making extracting meaningful features necessary for effective classification difficult. Underwater environments are highly dynamic, with fluctuating conditions that can drastically change the appearance of objects, further complicating the task. Traditional image classification techniques struggle to adapt to these complexities, often resulting in high false positive and false negative rates. This affects the reliability of underwater imaging applications, critical in marine biology, underwater exploration, and environmental monitoring. The need for robust and accurate classification models is evident to overcome these issues and ensure reliable data interpretation.

Optimization of these models becomes imperative to enhance their performance and adapt to challenging underwater conditions. Without effective optimization strategies, models may fail to generalize well across different scenarios, leading to inconsistent results. Addressing these problems requires innovative approaches to optimize the performance of classification models, ensuring they can handle the inherent difficulties of underwater image data and deliver reliable and accurate classification outcomes.

2. LITERATURE REVIEW

"Hybrid Carbon" [18] focuses on developing hybrid carbon nanotube self-adhesive sensors specifically designed for underwater target detection. These sensors leverage carbon nanotubes' unique electrical properties and sensitivity, combined with adhesive capabilities, to improve detection efficiency in aquatic environments. The hybrid nature of the sensors allows for enhanced adhesion to various surfaces, ensuring stable and reliable placement in dynamic underwater conditions. "Quadratic Boost Routing" [19] introduces a novel quadratic ensemble weighted emphasis boosting technique aimed at achieving energy and bandwidth-efficient routing in underwater sensor networks (UWSNs). The method optimizes data transmission processes to reduce energy consumption and extend network lifespan.

"Enhanced Aggregation"[20] presents an innovative algorithm that combines feature enrichment with a dynamic accumulation strategy. The algorithm enhances detection accuracy by progressively refining features' representation through multiple processing stages. "Self-Attention Detection" [21] introduces a self-attention and long-range relationship capture network specifically designed for underwater object detection. This advanced network architecture utilizes self-attention mechanisms to capture detailed spatial relationships within underwater scenes. "Boosting R-CNN"[22] explores an innovative reweighting technique for R-CNN samples based on errors identified by the Region Proposal Network (RPN). The proposed Boosting R-CNN method focuses on improving underwater object detection by addressing sample weighting, which enhances the learning process of the detection model.

"Slugging Impact Modeling"[23] identifies and models the maximum impact force associated with critical slugging in underwater compressed gas energy storage systems. Slugging, characterized by the rapid movement of liquid and gas phases, poses significant challenges to these systems' structural integrity and efficiency. This study focuses on understanding the dynamics of slugging phenomena and developing a comprehensive model to predict the maximum impact force exerted during slugging events. "PlasPi TDM"[24] enhances the capabilities of the low-cost camera platform, PlasPi TDM, to facilitate advanced underwater physical-ecological observations. The research aims to improve the quality and efficiency of underwater data collection by augmenting the existing platform with improved imaging and data processing capabilities. "Deep Sea Debris"[25] discusses a deep neural network-based approach for the instant detection of deep-sea debris to support maneuverable underwater machines in maintaining sustainable ocean environments. The proposed method leverages deep learning techniques to quickly and accurately identify debris in deep-sea environments.

"YOLO-Fish"[26] focuses on YOLO-Fish, a robust fish detection model designed to operate in realistic underwater environments. The model builds upon the YOLO (You Only Look Once) architecture, renowned for its speed and accuracy in object detection. YOLO-Fish is tailored to address the unique challenges of underwater imaging, such as low visibility, variable lighting conditions, and diverse fish appearances. "Sea Cucumber"[27] involves developing a monitoring system for cage-cultured sea cucumbers using an underwater time-lapse camera combined with deep learning-based image analysis. This innovative approach aims to enhance aquaculture management by providing continuous and automated monitoring of sea cucumber populations.

"AVOA-LSTM Sunglass"[28] presents an innovative approach for segmenting and classifying the eye region in sunglass images using AVOA-LSTM with MRCNN. The method combines the strengths of adaptive variable optimization algorithm (AVOA), long short-term memory (LSTM), and Mask R-CNN (MRCNN) to handle the challenges of image-based identification effectively. The approach aims to segment and classify the eye region obscured by sunglasses accurately, a task often complicated by reflections and partial occlusions. "Underwater Acoustic Denoising"[29] proposes a novel denoising method for underwater acoustic signals using a combination of empirical mode decomposition (EEMD), correlation coefficient analysis, permutation entropy, and wavelet threshold denoising. This comprehensive approach addresses the unique challenges underwater acoustic environments pose, such as noise from water movement, marine life, and human activities.

"Backscatter Recognition"[30] introduces a deep fuzzy extreme convolutional neural network optimized via the hunger games search algorithm for underwater backscatter recognition. The proposed method enhances recognition accuracy by integrating fuzzy logic principles with deep learning techniques. The hunger games search algorithm optimizes the network's parameters, improving its ability to recognize and classify backscatter signals in underwater environments. "Striation Images"[31] employs deep learning techniques on striation images for classifying underwater and surface targets. Striation images, characterized by delicate linear patterns, provide valuable visual features for distinguishing between different types of targets. "Fish Disease Recognition"[32] presents a CNN-OSELM multi-layer fusion network with an attention mechanism for recognizing fish diseases in aquaculture. The proposed method combines convolutional neural networks (CNNs) with an online sequential extreme learning machine (OSELM) to enhance the accuracy of disease diagnosis. The attention mechanism further improves the model's ability to focus on relevant features, enabling precise identification of various fish diseases. Optimization is crucial in most research to achieve the expected results. Optimization is common to in all domain research [33]-[61].

"Image Retrieval" [62] discusses an adaptable image retrieval system that utilizes kernel machines and selective sampling with relevance feedback. The system is designed to improve the accuracy and efficiency of retrieving relevant images from large datasets. The system learns user preferences by incorporating relevant feedback and iteratively refines the search results. Kernel machines enhance the system's ability to handle complex and nonlinear relationships between images, while selective sampling focuses on the most informative samples to optimize retrieval. "Bubble-Forming Regime"[63] identifies bubble-forming regimes based on textural features of an image and the feature selection method of MCWA. The research aims to refine the analysis and understanding of bubble dynamics, which are critical in various industrial and scientific applications.

"DeepSeaNet"[64] a bio-detection network designed for species identification in deep-sea imagery. The network uses advanced deep learning techniques to analyze and classify images from the deep sea, ensuring precise species identification in difficult underwater conditions. "MCANet"[65] utilizes a multi-channel attention network combined with a multi-color space encoder. This method improves underwater image classification by using various color space information and concentrating on essential image regions, resolving issues like color distortion and poor visibility.

2.1. Intention and Goal

The unique challenges of underwater environments present significant hurdles for image classification, highlighting the need for advanced methodologies to improve accuracy and reliability. Poor lighting, particulate matter, and dynamic underwater conditions often lead to degraded image quality, complicating the feature extraction and classification processes. These issues necessitate robust models capable of adapting to and overcoming these environmental complexities. The Intention stems from the critical need for accurate underwater image classification in marine biology, underwater exploration, and environmental monitoring applications. Enhanced classification capabilities can lead to better monitoring of aquatic ecosystems, more efficient exploration missions, and improved ecological assessments. The aim is to develop a model that excels in performance and maintains consistency and reliability under the varied conditions found in underwater environments.

This article aims to explore the potential of Quokka Optimization (QO) in enhancing CNNs for underwater image classification. By integrating QO into the Enhanced CNN (N-CNN) model, this study seeks to improve classification accuracy, robustness, and overall performance significantly. The goal is to demonstrate that QuokkaNet can effectively address the inherent challenges of underwater image classification, providing a reliable and optimized solution for complex underwater imaging tasks.

3. ENHANCED CNN MODELS USING QUOKKA OPTIMIZATION (QuokkaNet)

A CNN consists of multiple layers designed to process and transform an input image into a structured output, typically a classification or identification. The primary components include convolutional, pooling, and fully connected layers. Each layer plays a specific role in feature extraction and image interpretation. Fig 1. Illustrates the unprocessed underwater images.

3.1. Enhanced CNN with NAG (N-CNN)

CNN is an artificial neural network that processes and analyzes visual data. It comprises convolutional, pooling, and fully connected layers that work together to extract and interpret features from images. Nesterov Accelerated Gradient (NAG) is an optimization technique that enhances the traditional gradient descent method by considering the gradient at a future position, leading to faster convergence and improved performance. They are integrating the Nesterov Accelerated Gradient into a Convolutional Neural Network, resulting in an enhanced version called N-CNN.

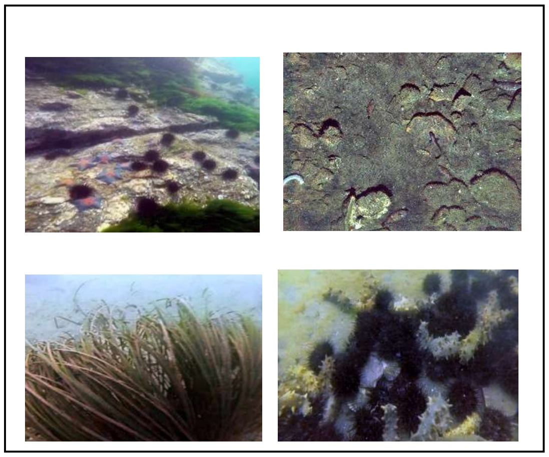
Fig 1. Unprocessed Underwater Images

3.1.1. Construct the Convolutional Layers of N-CNN

Constructing the convolutional layers in N-CNN involves defining the filters, applying convolution operations, and using activation functions. These steps are crucial for feature extraction from input images, forming the foundation for subsequent layers in the network. A filter is a small matrix used to scan through the input image. Assume that I stands for the input picture and K for the filter. A feature map F is generated by sliding the filter across the input picture and multiplying the results element-wise. The convolution procedure is represented as Eq.(1).

$$F(i,j) = \sum_{r=1}^{R} \sum_{t=1}^{T} I(i+r, j+t).K(r,t) \tag{1}$$

where R and T are the dimensions of the filter, and (i,j) denotes the position in the input image, this operation detects specific features within local regions of the image.

The filter's step size as it traverses the input picture is determined by Stride S. The filter shifts by a single pixel when S= 1, and with S=2, the filter shifts two pixels simultaneously. By enclosing the input image with a border of zeros, padding P controls the spatial dimensions of the output feature map. Stride S determines the step size by which the filter moves across the input image. If S=1, the filter moves one pixel at a time. If S=2, the filter moves two pixels at a time. Padding P involves adding a border of zeros around the input image to control the spatial dimensions of the output feature map. The dimensions of the output feature map F can be represented in Eq.(2).

$$F_{dim} = \left(\frac{I_{dim} - K_{dim} + 2P}{S}\right) + 1 \tag{2}$$

where I_{dim} and K_{dim} are the dimensions of the input image and filter, respectively. Proper selection of stride and padding ensures that important edge features are preserved.

The network is made non-linear by applying an activation function after convolution. A popular unit is the Rectified Linear Unit (ReLU) is expressed mathematically in Eq.(3).

$$ReLU(x) = \max(0, x) \tag{3}$$

ReLU activates only the positive values in the feature map, enhancing the network's ability to learn complex patterns.

The use of several filters captures various aspects of the input picture. Each filter produces a separate feature map, and stacking these maps along the depth dimension forms the complete output of the convolutional layer. If n filters are used, the output volume will have a depth of n. Let F_k represent the feature map from the k-th filter, then the output volume O is shown in Eq.(4).

$$O = [F_1, F_2, \dots, F_n] \tag{4}$$

The network can learn features with hierarchies when several convolutional layers are stacked. The earlier layers capture the lower-level elements, such as edges, while the later layers capture the higher-level features, such as forms and textures. Each subsequent convolutional layer takes the output volume O from the previous layer as its input, applies its own set of filters, and produces a new output volume. This iterative process allows the network to build a rich, multi-level representation of the input image.

3.1.2. Implement Pooling Layers of N-CNN

To control overfitting and minimize computational complexity, pooling layers decrease the spatial dimensions of the feature maps while keeping critical information. The majority of pooling operations employ max pooling. This technique involves defining a pooling window, typically of size $p \times p$. The pooling window slides over the feature map, and the maximum value is selected within each window. Let F represent the feature map and P the pooling window. The max pooling operation is expressed mathematically in Eq.(5).

$$P(i,j) = max\{F(m,n)|(m,n) \in W(i,j)\} \quad (5)$$

This operation significantly reduces the feature map's dimensions while preserving the most salient features, where $(i,)$ represents the pooling window centered at position $(i,)$.

Average pooling is another approach; it takes the whole window and averages it out. That which is known as the average pooling operation is expressed mathematically in Eq.(6).

$$P(i,j) = \frac{1}{p \times p} \sum_{(m,n)\in W(i,j)} F(m,n) \quad (6)$$

Average pooling provides a smoothed version of the feature map by averaging the values in each window, although max pooling is often preferred for retaining sharp features.

The dimensions of the output feature map P after pooling can be determined based on the size of the pooling window p, stride s, and the dimensions of the input feature map F. The mathematical expression for the output dimensions P_{dim} is Eq.(7).

$$P_{dim} = \left(\frac{F_{dim} - p}{s}\right) + 1 \quad (7)$$

where P_{dim} represents the dimensions of the input feature map. The stride s controls how much the pooling window moves across the feature map, influencing the output size.

In N-CNN, pooling layers are typically inserted after convolutional layers. It is possible to stack many pooling layers to extract hierarchical feature representations while reducing the spatial dimensions gradually. The network's generalizability is enhanced by each pooling layer, which processes the output of the one before it. The pooling operation maintains the spatial hierarchy of the features by retaining significant information across different scales. By reducing the feature map dimensions, pooling layers help manage the computational load and ensure efficient network training.

3.1.3. Normalization Layers of N-CNN

Normalization layers enhance the stability and efficiency of the network by standardizing the inputs to each layer, thereby accelerating training and improving performance. One standard normalizing method in N-CNN is batch normalization. By taking the batch standard deviation and dividing it by the batch mean, this approach normalizes the output of an earlier activation layer. Next, learnable parameters are used to scale and shift the normalized output. X denotes the batch normalization layer's input, the batch mean by μ, and the batch standard deviation by σ. The normalized output $\hat{X}$ is calculated as shown in Eq.(8).

$$\hat{X} = \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad (8)$$

where ϵ is a small constant added for numerical stability, this operation ensures that the input to each layer has a consistent distribution, which aids in training.

Learnable parameters γ (scale) and β (shift) are used to scale and shift the output after normalization is mathematically represented in Eq.(9).

$$Y = \gamma\hat{X} + \beta \quad (9)$$

The parameters γ and β allow the network to restore the representation power lost during normalization. These parameters are learned during training, enabling the network to adapt and improve performance

Layer normalization is another technique that independently normalizes the inputs across the features for each data point. This method is particularly useful for recurrent neural networks but can also be applied in N-CNN. Let μ_L and σ_L represent the mean and standard deviation for the features in a layer. The normalized output $\hat{X}_L$ for layer normalization is expressed mathematically in Eq.(10).

$$\hat{X}_L = \frac{X - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}} \quad (10)$$

Layer normalization helps stabilize the learning process and ensures that the activations within each layer remain well-behaved.

Normalization layers are usually added to N-CNN before the activation function and after each fully linked or convolutional layer. This integration ensures that the inputs to subsequent layers have consistent distributions, leading to improved convergence rates and model performance.

Normalization layers provide several benefits, including reduced internal covariate shift, accelerated training, and improved gradient flow. Normalization layers allow the network to learn more effectively by standardizing the inputs and making the training process more efficient. This step is essential for building deep and complex networks like N-CNN, where maintaining stability and efficiency is crucial. By integrating normalization layers, N-CNN achieves better training dynamics and enhanced performance, paving the way for constructing more advanced network architectures.

3.1.4. Fully Connected Layers of N-CNN

This phase is to provide accurate predictions, these layers also called dense layers integrate the characteristics obtained by the convolutional and pooling layers. The last pooling or convolutional layer output consists of multi-dimensional feature maps. Before feeding these into the fully connected layers, it is necessary to flatten them into a one-dimensional vector. Let F represent the 3D feature map output with dimensions $d \times h \times w$ (depth, height, width). The flattening operation transforms this 3D feature map into a 1D vector V represented mathematically in Eq.(11).

$$V = [f_1, f_2, \dots, f_{x \times h \times w}] \tag{11}$$

where f represents the individual elements of the feature map.

Neural connections between layers are quite thick in a completely linked layer. Each neuron processes its inputs in a fully connected layer in a weighted sum and then applies an activation function. Let W represent the weight matrix and b the bias vector for the fully connected layer. The output Z of a fully connected layer can be expressed as Eq.(12).

$$Z = W.V + b \tag{12}$$

where W has dimensions n×(d×h×w),V has dimensions (d×h×w)×1, and b has dimensions n×1. Here, n is the number of neurons in the fully connected layer.

By applying an activation function to the fully linked layer's output, non-linearity may be introduced into the network. Here is the definition of the Rectified Linear Unit (ReLU), an activation function often used mathematically expressed in Eq.(13).

$$ReLU(z) = \max(0, z) \tag{13}$$

N-CNN may contain multiple fully connected layers to transform the extracted features gradually into the desired output. After processing the output of the preceding layer, each fully connected layer applies a weighted sum and then sends the result via an activation function. Let $Z^{(i)}$ represent the output of the i-th fully connected layer, $W^{(i)}$ the weight matrix, and $b^{(i)}$ the bias vector. For the i-th fully connected layer, the operation can be expressed as Eq.(14).

$$Z^{(i)} = ReLU\left(W^{(i)}.Z^{(i-1)} + b^{(i)}\right) \tag{14}$$

where $Z^{(i-1)}$ is the output from the previous layer.

Depending on the job, the activation function used by the final fully linked layer is usually variable. If you're doing a classification job, the softmax activation function will turn your output into a probability distribution across the classes. The output of the final completely linked layer is denoted as $Z^{(L)}$, where L is the total number of layers. The softmax function is defined as shown in Eq.(15).

$$softmax(z_j) = \frac{e^{z_k}}{\sum_{k=1}^{n} e^{z_k}} \tag{15}$$

where z_j is the jth element of $Z^{(L)}$, and $n$is the number of classes. This function ensures that the output probabilities sum to one, making it suitable for classification tasks. Incorporating fully connected layers allows N-CNN to integrate and interpret the hierarchical features extracted by previous layers, enabling the network to make accurate predictions.

3.1.5. Loss Function of N-CNN

As a training optimization tool, the loss function measures how far off the actual objectives are from the projected outputs. The most popular loss function for classification tasks is the cross-entropy loss, which produces a probability value between 0 and 1 as an output and quantifies the effectiveness of the classification model. Let y denote the true label and $\hat{y}$ represent the predicted probability for each class. The cross-entropy loss L for a single example can be expressed mathematically in Eq.(16).

$$L = -\sum_{i=1}^{C} y_i log(\hat{y}_i) \tag{16}$$

The sentence states that for each observation i, y_i is either 0 or 1, indicating the proper classification for that observation and that $\hat{y}_i$ is the projected probability of that observation being in class i. Here, C is the number of classes. This loss function penalizes incorrect classifications more severely, thus encouraging the network to produce accurate predictions. A batch of N training examples, the average cross-entropy loss L_{batch} is shown in Eq.(17).

.

$$L_{batch} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C} y_{ij}(\hat{y}_{ij}) \qquad (17)$$

where y_{ij} and $\hat{y}_{ij}$ represent the true label and predicted probability for the j-th example and the i-th class, respectively. Averaging the loss over the batch ensures that the gradient updates consider the overall performance across multiple examples, promoting more stable and generalized learning.

In some cases, the mean squared error (MSE) is used as the loss function when dealing with regression tasks. Let y be the true value and $\hat{y}$ the predicted value. The mean squared error L_{MSE} is calculated as shown in Eq.(18)

$$L_{MSE} = -\frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2 \qquad (18)$$

To reduce the margin of error in continuous value prediction, this loss function calculates the average squared difference between the actual and forecast values. Incorporating regularization terms into the loss function can prevent overfitting by penalizing large weights. L2 regularization, also known as weight decay, adds a penalty proportional to the sum of the squared weights. The regularized lossL_{reg} can be expressed mathematically in Eq.(19).

$$L_{reg} = L + \lambda\sum_{k=1}^{K} w_k^2 \qquad (19)$$

where λ is the regularization parameter and w_k represents the weights of the network. This term discourages the network from assigning excessively high importance to any feature, promoting a more balanced model. By defining an appropriate loss function, N-CNN effectively measures the discrepancy between its predictions and the true labels, providing a basis for optimizing the network's parameters to minimize this discrepancy. This step is fundamental in training the network to achieve high accuracy and generalization on unseen data.

3.1.6. Nesterov Accelerated Gradient (NAG) of N-CNN

Nesterov Accelerated Gradient (NAG) is an optimization technique designed to improve the convergence speed and stability of the training process. Building on the momentum method, NAG anticipates the future position of the parameters to make more informed updates. In N-CNN, applying NAG involves several steps, beginning with initializing parameters and iteratively updating them based on the gradients of the loss function. Let θ represent the parameters of N-CNN, v the velocity vector, η the learning rate, and μ the momentum coefficient. The update rule for the velocity vector in the standard momentum method is expressed as Eq.(20).

$$v_{t+1} = \mu v_t - \eta\nabla L(\theta_t) \qquad (20)$$

where $\nabla L(\theta_t)$denotes the gradient of the loss function L concerning the parameters θ_t at iteration t. The parameters are then updated, as shown in Eq.(21).

$$\theta_{t+1} = \theta_t - v_{t+1} \qquad (21)$$

The key difference in NAG is that the gradient is not calculated at the current parameters. θ_t, but at a lookahead position $\theta_t + \mu v_t$.This anticipatory step helps to correct the course before making the actual update, thereby improving the convergence rate. The lookahead gradient is expressed mathematically in Eq.(22).

$$\nabla L(\theta_t + \mu v_t) \qquad (22)$$

The velocity update in NAG is then modified to incorporate this lookahead gradient, represented mathematically in Eq.(23).

$$\theta_{t+1} = \theta_t + v_{t+1} \qquad (23)$$

These equations collectively describe the NAG optimization process, which involves first computing the lookahead gradient, updating the velocity vector, and adjusting the parameters. This approach provides a more accurate direction for updates by considering future positions, leading to faster and more stable convergence. Applying NAG to N-CNN entails repeatedly executing these steps for each iteration of the training process; by consistently using the lookahead gradient, N-CNN benefits from more precise updates that help to avoid oscillations and overshooting, common issues in standard gradient descent methods. Consequently, NAG enables N-CNN to achieve better performance and efficiency during training, ultimately leading to a more robust and accurate model. This step is integral to the optimization strategy, ensuring that the network parameters converge to their optimal values effectively.

3.1.7. N-CNN Training

Forward and backward propagation is essential for training a convolutional neural network (CNN). To generate predictions, inputs must first travel through the network during forward propagation. To update the network parameters, the backward propagation phase determines the loss function's gradients concerning those parameters.

Forward propagation starts by passing input data through each network layer, from convolutional layers to fully connected layers, ultimately producing the output. Let$x$ represent the input and $W^{(l)}$ and $b^{(l)}$ denote the weights and biases of layer l, respectively. For a convolutional layer, the output feature map $Z^{(l)}$can be expressed as Eq.(24).

$$Z^{(l)} = f\left(W^{(l)} * A^{(l-1)} + b^{(l)}\right) \tag{24}$$

where $A^{(l-1)}$ is the activation from the previous layer, $*$ denotes the convolution operation, and f is the activation function. The output is mathematically represented in fully connected layers in Eq.(25).

$$Z^{(l)} = W^{(l)}A^{(l-1)}b^{(l)} \tag{25}$$

The final output of the network, after passing through all layers, is used to calculate the loss L concerning the true labels.

Optimization is made possible via backward propagation, which entails computing the gradients of the loss function L concerning each network parameter. The chain rule is applied to propagate gradients backward through the network. The gradient of the loss concerning the weights in the output layer $W^{(L)}$ is expressed mathematically in Eq.(26).

$$\frac{\partial L}{\partial W^{(L)}} = \frac{\partial L}{\partial A^{(L)}} \cdot \frac{\partial A^{(L)}}{\partial Z^{(L)}} \cdot \frac{\partial Z^{(L)}}{\partial W^{(L)}} \tag{26}$$

where $A^{(L)}$ is the activation of the last layer, and $Z^{(L)}$ is the pre-activation output. The gradients for the biases $b^{(L)}$ are shown in Eq.(27).

$$\frac{\partial L}{\partial b^{(L)}} = \frac{\partial L}{\partial A^{(L)}} \cdot \frac{\partial A^{(L)}}{\partial Z^{(L)}} \cdot \frac{\partial Z^{(L)}}{\partial b^{(L)}} \tag{27}$$

Gradients are calculated similarly for intermediate layers, considering the contributions from subsequent layers. For a hidden layer l, the gradient concerning the weights $W^{(l)}$ is expressed mathematically in Eq.(28).

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l+1)} . A^{(l)T} \tag{28}$$

where $\delta^{(l+1)}$ is the gradient propagated from the next layer, and $A^{(l)T}$ is the transpose of the activation of the current layer.

After computing the gradients, parameters are updated using an optimization algorithm such as Nesterov Accelerated Gradient (NAG). The velocity and parameter updates are performed in Eq.(29) and Eq.(30).

$$v_{t+1} = \mu v_t - \eta \nabla L(\theta_t + \mu v_t) \tag{29}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{30}$$

Training continues iteratively, with forward and backward propagation steps repeated for each batch of input data. This process adjusts the network parameters to minimize the loss function, ultimately improving the model's performance on the given task.

3.1.8. Evaluation and Fine-Tuning

Evaluation typically involves using a separate validation dataset to measure the network's performance and identify areas for improvement. Forward propagation is conducted on the validation dataset to obtain predictions for N-CNN. Let $\hat{y}_i$ represent the predicted output for the i-th sample and y_i the true label. Standard assessment measures include the F1 score, recall, accuracy, and precision. For a classification task, accuracy A represented mathematically in Eq.(31).

$$A = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i) \tag{31}$$

Assuming that there are N samples in the validation set, the indicator function 1 gives one if the predicted label is identical to the real label and 0 otherwise. The loss function used during training is also computed on the validation dataset to monitor overfitting. The cross-entropy loss L_{val} for the validation set can be expressed as Eq.(32).

$$L_{val} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} log(\hat{y}_{ij}) \tag{32}$$

where y_{ij} is the true label, and $\hat{y}_{ij}$ is the predicted probability for class j of the ith sample.

Fine-tuning involves adjusting hyperparameters, optimizing the network architecture, and applying techniques to reduce overfitting. One approach is to use regularization methods, such as dropout, randomly setting a fraction of the activations to zero during training. Let p represent the dropout probability. The dropout-modified activation A_{drop} for a layer can be expressed as Eq.(33).

$$A_{drop} = A.r \tag{33}$$

where A is the original activation, and r is a mask vector with each element independently set to 1 with probability $1 - p$ and 0 with probability p.

Adjusting the learning rate during training can also improve the model's performance. It is possible to use adaptive learning rates or learning rate scheduling methods. A common approach is to reduce the learning rate by a factor γ after a certain number of epochs T.

$$\eta_{new} = \eta . \gamma \quad (34)$$

where in Eq.(34), η is the current learning rate and η_{new} is the updated learning rate.

Early stopping is another technique to prevent overfitting, terminating training when the validation loss ceases to improve. Let $L_{val}(t)$ be the validation loss at epoch t as expressed in Eq.(35).

$$L_{val}(t) = L_{val}(t - p) \quad (35)$$

For p consecutive epochs, it was indicating no improvement. The model's performance on the validation set is optimized by evaluating and fine-tuning N-CNN, ensuring better generalization to unseen data. This iterative process involves continuous monitoring and adjustment, culminating in a robust and accurate neural network model.

3.2. N-CNN Model with Quokka Optimization (QuokkaNet)

Quokka Optimization (QO) is a metaheuristic optimization algorithm inspired by the foraging behavior of quokkas, small marsupials native to Australia. QO aims to explore the search space efficiently and converge to optimal solutions by mimicking the quokkas' adaptive foraging strategies. Applying QO to enhance the performance of the Enhanced CNN (N-CNN) model for underwater image identification and classification involves several steps.

3.2.1. QuokkaNet Initialization

In the initialization step of QuokkaNet, the population of quokkas representing potential solutions to the optimization problem is initialized. Each quokka corresponds to a unique set of hyperparameters or network configurations for the Enhanced CNN (N-CNN) model. Let Q denote the population of quokkas, q_i represent the i-th quokka and θ_i denote the parameters of the i-th quokka.

Parameter Initialization is to initialize the parameters θ_i of each quokka q_i randomly within predefined ranges. These parameters may include learning rates, activation functions, layer configurations, and other N-CNN-related hyperparameters expressed mathematically in Eq.(36).

$$\theta_i = random(a, b) \quad (36)$$

where a and b represent the lower and upper bounds of the parameter space, respectively.

Population Creation is to create the initial population Q by generating a specified number of quokkas q_i, each with its unique parameter configuration θ_i. The population size is determined based on the optimization requirements and computational resources, as shown mathematically in Eq.(37).

$$Q = \{q_1, q_2, \dots, q_N\} \quad (37)$$

where N is the number of quokkas in the population.

Evaluate the fitness of each quokka in the population based on its parameter configuration θ_i. This involves training and validating the corresponding N-CNN model using the hyperparameters of each quokka and measuring its performance on a validation dataset expressed as Eq.(38).

$$f(q_i) = evaluate\_fitness(\theta_i) \quad (38)$$

where $f(q_i)$ represents the fitness score of quokka q_i. Once all quokkas in the population have been initialized and their fitness evaluated, the initialization step is complete. QuokkaNet is now ready to proceed to the exploration and optimization phases.

3.2.2. QuokkaNet Exploration

In this phase of QuokkaNet, quokkas navigate the search space by adjusting their positions based on local and global information. This process aims to explore different combinations of hyperparameters and network configurations for the Enhanced CNN (N-CNN) model, facilitating the discovery of promising solutions. Let Q denote the population of quokkas, q_i represent the i-th quokka and θ_i denote the parameters of the i-th quokka.

Quokkas explore their local surroundings by adjusting their parameter configurations within a certain neighborhood. This encourages the exploitation of nearby solutions while maintaining diversity within the population. The local exploration can be mathematically expressed as Eq.(39).

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \epsilon_i \quad (39)$$

where ϵ_i represents the perturbation applied to the parameters of quokka q_i at iteration t.

Quokkas also engage in global exploration by exchanging information with others in the population. This allows them to learn from successful solutions discovered by their peers and explore regions of the search space that have not yet been thoroughly explored. The global exploration process can be formalized as Eq.(40).

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \sum_{j=1}^{N} \beta_{ij}.\left(\theta_j^{(t)} - \theta_i^{(t)}\right) \quad (40)$$

Quokkas employs strategies to explore different regions of the search space to prevent premature convergence and maintain diversity within the population. This includes introducing randomness in parameter adjustments and encouraging quokkas to explore unexplored areas. The diversity maintenance process can be represented mathematically in Eq.(41).

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \gamma_i.random\_vector \quad (41)$$

where γ_i represents the exploration rate of quokka q_i, and $random\_vector$ is a vector of random perturbations.

After exploring new parameter configurations, the fitness of each quokka is re-evaluated to assess the performance of the corresponding N-CNN model. This involves training and validating the model using the updated hyperparameters and measuring its performance on a validation dataset.

$$f\left(q_i^{(t+1)}\right) = evaluate\_fitness\left(\theta_i^{(t+1)}\right) \quad (42)$$

where in Eq.(42), $f\left(q_i^{(t+1)}\right)$ represents the fitness score of quokka q_i at iteration $t+1$.

Once all quokkas in the population have explored new parameter configurations and their fitness has been re-evaluated, the exploration step is complete. QuokkaNet is now ready to proceed to the exploitation phase, where it refines promising solutions discovered during exploration. The exploration step of QuokkaNet enables Quokkas to navigate the search space and discover potentially promising solutions for optimizing the performance of the Enhanced CNN (N-CNN) model.

3.2.3. QuokkaNet Exploitation

In the exploitation phase of QuokkaNet, quokkas focus on refining and improving promising solutions discovered during exploration. This process involves leveraging local and global information to exploit regions of the search space with high-quality solutions, aiming to enhance further the performance of the Enhanced CNN (N-CNN) model. Let Q denote the population of quokkas, q_i represent the i-th quokka and θ_i denote the parameters of the i-th quokka.

Quokkas refines their parameter configurations by focusing on local improvements that are near-promising solutions. This involves adjusting parameter values based on local information obtained during exploration. The local exploitation process can be formulated as Eq.(43).

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \delta_i.\nabla f\left(q_i^{(t)}\right) \quad (43)$$

where δ_i represents the step size for quokka q_i and $\nabla f\left(q_i^{(t)}\right)$ is the gradient of the fitness function concerning the parameters of quokka q_i at iteration t.

Quokkas also benefit from global information shared by other quokkas in the population. By leveraging successful solutions discovered by their peers, quokkas can exploit regions of the search space with higher-quality solutions. The global exploitation process can be mathematically expressed as Eq.(44).

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \sum_{j=1}^{N} \beta_{ij}.\left(\theta_i^{(t)} - \theta_i^{(t)}\right) \quad (44)$$

where β_{ij} represents the influence of quokka q_j on quokka q_i, determined based on their relative fitness scores.

Quokkas iteratively refine their parameter configurations to further optimize the N-CNN model's performance. This involves adjusting hyperparameters such as learning rates, activation functions, and network architectures to improve the model's accuracy and generalization. The parameter refinement process is represented mathematically in Eq.(45)

$$\theta_i^{(t+1)} = refine\_parameters\left(\theta_i^{(t)}\right) \quad (45)$$

where $refine\_parameters$ represents a function that refines the parameters of quokka q_i based on local and global information. The fitness of each quokka is re-evaluated to assess the performance of the updated N-CNN model. This involves training and validating the model using the refined hyperparameters and measuring its performance on a validation dataset. Once all quokkas in the population have refined their parameter configurations and their fitness has been re-evaluated, the exploitation step is complete. QuokkaNet is now ready to proceed to the next iteration, where it continues to refine and improve the solutions discovered during exploitation.

3.2.4. QuokkaNet Fitness Evaluation

The ensemble predictions of the AdamBoost In the fitness evaluation step of QuokkaNet, the performance of each quokka is assessed based on its parameter configuration and its corresponding N-CNN model's performance on a validation dataset. This step is crucial in guiding the optimization process by providing feedback on the quality of solutions discovered during exploration and exploitation. Let Q denote the population of quokkas, q_i represent the i-th quokka and θ_i denote the parameters of the i-th quokka. Quokkas train their corresponding N-CNN models using the parameter configurations. θ_i determined during exploration and exploitation. This involves feeding training data into the network, computing predictions, and comparing them with the ground truth labels to calculate a loss function. The training process can be represented mathematically in Eq.(46).

$$Train\left(q_i^{(t)}\right) \tag{46}$$

where Train represents the training process for quota q_i at iteration t.

After training, quokkas validate their N-CNN models using a separate validation dataset to assess their generalization performance. This involves passing validation data through the trained models, computing predictions, and evaluating them against the true labels expressed mathematically in Eq.(47).

$$Validate\left(q_i^{(t)}\right) \tag{47}$$

where $Validate$ represents the validation process for quokka q_i at iteration t.

Quokkas computes each individual's fitness based on the performance of its corresponding N-CNN model on the validation dataset. This fitness score measures the solution quality the quokka's parameter configuration represents. The fitness computation can be formalized as shown in Eq.(48).

$$f\left(q_i^{(t)}\right) = fitness\left(Validate\left(q_i^{(t)}\right)\right) \tag{48}$$

where $f\left(q_i^{(t)}\right)$ represents the fitness score of quokka q_i at iteration t.

Once the fitness of each quokka has been computed, the population of quokkas is updated based on their fitness scores. Quokkas with higher fitness values are more likely to survive and propagate their parameter configurations to the next generation. In comparison, those with lower fitness values may be replaced by new offspring generated during optimization.

$$Q^{(t+1)} = Select\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \tag{49}$$

where in Eq.(49), $Q^{(t)}$ and $Q^{(t+1)}$ represent the populations of quokkas at iterations t and $t+1$, respectively, and $Select$ is a selection process that determines which quokkas survive and reproduce based on their fitness scores.

QuokkaNet checks for convergence by monitoring the fitness scores of the population over multiple iterations. If the fitness scores stabilize or show no significant improvement over several iterations, the optimization process may be terminated, indicating that QuokkaNet has converged to a satisfactory solution.

$$Convergence\ Check\left(f\left(q_i^{(t)}\right)\right) \tag{50}$$

where in Eq.(50), $Convergence\,Check$ assesses whether the fitness scores have converged.

The fitness evaluation step of QuokkaNet plays a crucial role in guiding the optimization process by assessing the quality of solutions discovered during exploration and exploitation. By training N-CNN models and evaluating their performance on a validation dataset, QuokkaNet identifies promising solutions and continuously improves the model's accuracy and generalization capabilities.

3.2.5. QuokkaNet Selection

In the selection step of QuokkaNet, quokkas are chosen for propagation to the next generation based on their fitness scores. This process determines which quokkas survive and reproduce, guiding the evolution of the population towards higher-quality solutions. Let Q denote the population of quokkas, q_i represent the i-th quokka, $f\left(q_i^{(t)}\right)$ denote the fitness score of quokka q_i at iteration t, and $Q^{(t)}$ represent the population of quokkas at iteration t.

Quokkas are ranked based on their fitness scores, with higher fitness scores indicating better-performing individuals. This ranking allows QuokkaNet to prioritize individuals with superior solutions for propagation to the next generation. The ranking process can be mathematically represented as Eq.(51).

$$Rank\left(q^{(t)}, f\left(q_i^{(t)}\right)\right) \qquad (51)$$

where $Rank$ ranks the quokkas in $Q^{(t)}$ based on their fitness scores.

Quokkas are propagated using a selection mechanism that considers their ranking and fitness scores. Various selection mechanisms can be employed, such as roulette wheel selection, tournament selection, or elitism. The selection mechanism ensures that quokkas with higher fitness scores are more likely to be selected for propagation.

$$Select\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \qquad (52)$$

where in Eq.(52), $Select$ selects quokkas from $Q^{(t)}$ based on their fitness scores.

Quokkas selected for propagation contribute their parameter configurations to the next generation, ensuring that promising solutions are preserved and further evolved. This involves creating offspring by applying genetic operators such as crossover and mutation to the selected quokkas' parameter configurations, expressed mathematically as Eq.(53).

$$Propagate\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \qquad (53)$$

where $Propagate$ generates offspring based on the selected quokkas in $Q^{(t)}$.

After propagation, the population of quokkas is updated to include the selected individuals and their offspring. This ensures that the population remains diverse and continues to explore different regions of the search space. The population update process can be expressed as Eq.(54).

$$Q^{(t+1)} = Update\left(Q^{(t)}, Propagate\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right)\right) \qquad (54)$$

where $Q^{(t+1)}$ represents the population of quokkas at iteration $t+1$, and $Update$ updates the population based on the selected individuals and their offspring.

QuokkaNet may control the size of the population to manage computational resources efficiently and prevent overfitting. This involves maintaining a constant population size or dynamically adjusting it based on the performance of the optimization process is represented mathematically in Eq.(55).

$$Control\_Population\_Size\left(Q^{(t+1)}\right) \qquad (55)$$

where $Control\_Population\_Size$ manages the size of the population at iteration $t+1$. The selection step of QuokkaNet plays a crucial role in guiding the evolution of the population towards higher-quality solutions by selecting quokkas based on their fitness scores.

3.2.6. Adaptation of QuokkaNet

Quokkas adjust their behavior based on the success or failure of previous foraging attempts. This adaptive process ensures that QuokkaNet efficiently balances exploration and exploitation strategies to navigate the search space effectively. Let Q denote the population of quokkas, q_i represent the i-th quokka, $f\left(q_i^{(t)}\right)$ denote the fitness score of quokka q_i at iteration t, and $Q^{(t)}$ represent the population of quokkas at iteration t.

Quokkas update their exploration and exploitation strategies based on the success or failure of previous foraging attempts. This involves adjusting parameters or probabilities associated with exploration and exploitation mechanisms to optimize the search process, expressed mathematically in Eq.(56).

$$Update\_Strategy\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \qquad (56)$$

where $Update\_Strategy$ adjusts the exploration and exploitation strategies based on the performance of the population.

Quokkas employ adaptive mechanisms to dynamically adjust their behavior in response to changes in the search landscape. These mechanisms may include adaptive step sizes, mutation rates, or selection probabilities, allowing QuokkaNet to adapt its exploration and exploitation strategies over time.

$$Adapt\_Mechanisms\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \qquad (57)$$

where in Eq.(57), $Adapt\_Mechanisms$ updates adaptive parameters or probabilities based on the performance of the population.

Quokkas learn from their experiences by tracking the success or failure of previous foraging attempts and adjusting their behavior accordingly. This learning process enables QuokkaNet to recognize promising regions of the search space and focus its exploration and exploitation efforts on areas with higher potential for finding optimal solutions.

$$Learn\_from\_Experience\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \quad (58)$$

where in Eq.(58), $Learn\_from\_Experience$ updates quokkas' behavior based on the outcomes of previous iterations.

QuokkaNet maintains an adaptive balance between exploration and exploitation strategies to ensure effective navigation of the search space. This involves dynamically adjusting the trade-off between exploring new regions and exploiting known solutions based on the current state of the optimization process. The adaptive exploration-exploitation balance can be expressed mathematically in Eq.(59).

$$Adapt\_Exploration\_Exploitation\_Balance\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \quad (59)$$

where $Adapt\_Exploration - Exploitation\_Balance$ adjusts the exploration and exploitation balance based on the population's performance. The adaptation step of QuokkaNet ensures that the optimization process remains flexible and responsive to changes in the search landscape.

3.2.7. QuokkaNet Migration

In the migration phase of QuokkaNet, quokkas exchange information and genetic material to facilitate diversity and prevent premature convergence. This process involves the movement of individuals between different subpopulations or habitats, allowing for the exchange of beneficial traits and promoting the exploration of new regions in the search space. Let Q denote the population of quokkas, q_i represent the i-th quokka, $f\left(q_i^{(t)}\right)$ denote the fitness score of quokka q_i at iteration t, and $Q^{(t)}$ represent the population of quokkas at iteration t. QuokkaNet divides the population of quokkas into multiple subpopulations or habitats, each representing a distinct search space region. This division allows for localized exploration and exploitation within each subpopulation while maintaining diversity at the global level is shown in mathematical form as Eq.(60).

$$Form\_Subpopulations\left(Q^{(t)}\right) \quad (60)$$

where $Form\_Subpopulations$ partitions the population into subpopulations.

Quokkas migrate between subpopulations to exchange genetic material and information, promoting diversity and spreading beneficial traits. The migration mechanism can involve periodic exchange of individuals, directed migration towards regions with higher potential, or random movement to explore new areas, expressed mathematically in Eq.(61).

$$Migrate\left(Q^{(t)}\right) \quad (61)$$

where $Migrate$ facilitates the movement of quokkas between subpopulations.

During migration, quokkas exchange genetic material through mechanisms such as crossover and recombination, leading to the creation of offspring with diverse genetic characteristics. This genetic exchange promotes the spread of beneficial traits and facilitates adaptation to changing environmental conditions. The genetic exchange process can be represented mathematically as Eq.(62).

$$Genetic\_Exchange\left(Q^{(t)}\right) \quad (62)$$

where $Genetic\_Exchange$ facilitates the exchange of genetic material between quokkas.

Migration promotes diversity within the population by introducing new genetic material and facilitating the exploration of new regions in the search space. This diversity helps prevent premature convergence and ensures that the optimization process continues to explore a wide range of solutions.

$$Promote\_Diversity\left(Q^{(t)}\right) \quad (63)$$

where in Eq.(63), $Promote\_Diversity$ maintains diversity within the population through migration.

QuokkaNet adapts its migration strategy based on the current state of the optimization process and the characteristics of the search landscape. This adaptive strategy allows QuokkaNet to adjust the frequency and direction of migration to maximize exploration and exploitation efficiency.

$$Adapt\_Migration\_stragety\left(Q^{(t)}, f\left(q_i^{(t)}\right)\right) \quad (64)$$

where in Eq.(64), $Adapt\_Migration\_stragety$ adjusts the migration strategy based on the fitness scores of the population. The migration step of QuokkaNet facilitates the exchange of genetic material and information between quokkas, promoting diversity and exploration of new regions in the search space.

3.2.8. QuokkaNet Termination

This phase ensures that the optimization process halts when certain conditions are met, indicating that QuokkaNet has sufficiently explored the search space or achieved satisfactory performance. Let Q denote the population of quokkas, q_i represent the i-th quokka, $f\left(q_i^{(t)}\right)$ denote the fitness score of quokka q_i at iteration t, and $Q^{(t)}$ represent the population of quokkas at iteration t. QuokkaNet checks for convergence by monitoring the fitness scores of the population over multiple iterations. Suppose the fitness scores stabilize or show no significant improvement over several iterations. The optimization process may be terminated in that case, indicating that QuokkaNet has converged to a satisfactory solution, expressed as Eq.(65).

$$Convergence\_Check\left(f\left(q_i^{(t)}\right)\right) \tag{65}$$

where $Convergence\_Check$ assesses whether the fitness scores have converged.

QuokkaNet may be terminated after a predefined maximum number of iterations to limit computational resources and prevent overfitting. The optimization process is halted once the maximum number of iterations is reached and the best-performing solution discovered so far is returned. The termination based on maximum iterations can be expressed as Eq.(66).

$$Max\_Iterations\_Check(t) \tag{66}$$

where $Max\_Iterations\_Check$ checks if the current iteration exceeds the maximum allowable iterations.

The optimization process may be terminated if QuokkaNet performs satisfactorily on a validation dataset. This criterion ensures that the optimization process halts once the model's performance meets or exceeds predefined performance thresholds, as expressed mathematically in Eq.(67).

$$Performance\_Check\left(f\left(q_i^{(t)}\right)\right) \tag{67}$$

where $Performance\_Check$ evaluates whether the model's performance meets predefined thresholds.

QuokkaNet may be terminated if computational or memory resources are exhausted, preventing further optimization. This ensures the optimization process does not consume excessive resources beyond predefined limits. The termination based on resource constraints can be represented as Eq.(68).

$$Resource\_Constraints\_Check \tag{68}$$

The optimization process may be terminated based on user intervention, allowing users to manually halt the process if necessary. This criterion allows users to intervene and terminate the optimization process based on their judgment. The termination based on user intervention can be expressed as Eq.(69).

$$User\_Intervention\_Check \tag{69}$$

where $User\_Intervention\_Check$ allows users to halt the optimization process manually.

The termination step of QuokkaNet ensures that the optimization process concludes appropriately based on predefined stopping criteria. Fig 2. Illustrates the processed underwater images.

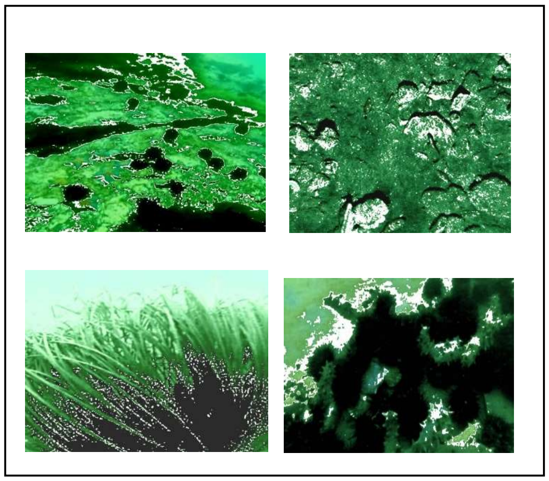
Fig 2. Processed Underwater Images

4. DATASET

The Large Scale Underwater Image Dataset (LSUI) is an essential dataset designed to support underwater image processing, identification, and classification research. With thousands of images captured in diverse marine environments, the LSUI dataset is valuable for developing and testing machine learning models. These images include a variety of scenes, such as different species of marine life, underwater landscapes, and man-made objects, making the dataset comprehensive and versatile. One of the key challenges in underwater imaging is the effect of environmental factors on image quality. The LSUI dataset addresses this by including images taken under various conditions, such as varying depths, different times of day, and various water clarity levels. This diversity in imaging conditions is critical for training machine learning models that can generalize well to underwater environments. The dataset helps overcome the difficulties associated with light absorption, scattering, and turbidity, which often degrade underwater image quality.

Each image in the LSUI dataset is accompanied by detailed annotations, providing labels for objects and features within the image. This metadata includes information about the capture conditions, such as location, depth, and environmental factors. These annotations are invaluable for supervised learning tasks, enabling precise training models for underwater image identification and classification. The comprehensive labeling ensures that the dataset can be effectively used to develop algorithms capable of accurate and reliable performance. The LSUI dataset is not only beneficial for machine learning but also for various scientific and conservation applications. In marine biology, researchers can use the dataset to study species distribution, monitor changes in aquatic ecosystems, and assess the health of coral reefs. Environmental monitoring efforts benefit from the ability to track underwater pollution, observe habitat changes, and evaluate the impact of human activities on marine environments. The underwater archaeologists can utilize the dataset to document and analyze submerged cultural heritage sites.

The LSUI dataset addresses the significant challenges of underwater imaging by providing a diverse and well-annotated collection of images. This resource is crucial for advancing the field of underwater image processing, enabling the development of more effective algorithms for identification and classification. Its wide-ranging applications in scientific research, conservation, and archaeology highlight the dataset's importance in enhancing the understanding and preservation of underwater environments.

5. RESULTS AND DISCUSSION:

The results from the evaluation of DeepSeaNet, MCANet, and QuokkaNet reveal significant differences in performance metrics, highlighting the efficacy of each model in underwater image classification tasks.

5.1. Classification Accuracy and F-Measure

The true positive (TP) counts were 1342.723 for DeepSeaNet, 1514.811 for MCANet, and 2001.750 for QuokkaNet. This progression indicates a clear improvement in correctly identifying positive instances, with QuokkaNet demonstrating superior performance. The true negative (TN) values followed a similar trend, with DeepSeaNet achieving 1383.806, MCANet 1549.288, and QuokkaNet 2057.295, suggesting QuokkaNet's enhanced capability to identify negative instances correctly. The false positive (FP) counts were 1067.653 for DeepSeaNet, 964.721 for MCANet, and 494.345 for QuokkaNet. Lower FP rates indicate fewer incorrect identifications, and QuokkaNet again shows its advantage. For false negatives (FN), DeepSeaNet had 1209.817, MCANet 975.180, and QuokkaNet 450.610, underscoring QuokkaNet's improved performance in minimizing missed positive instances.

.

Fig 3. Illustrates the outcome of Classification Accuracy and F-Measure. The True Positive Rate (TPR) or sensitivity, which measures the proportion of actual positives correctly identified, was 52.603% for DeepSeaNet, 60.836% for MCANet, and 81.625% for QuokkaNet. This metric illustrates QuokkaNet's effectiveness in capturing positive instances. The True Negative Rate (TNR), indicating the proportion of actual negatives correctly identified, was 56.448% for DeepSeaNet, 61.626% for MCANet, and 80.626% for QuokkaNet, further affirming QuokkaNet's superiority in correctly identifying negative instances. The False Positive Rate (FPR), reflecting the proportion of incorrect positive identifications among actual negatives, was 43.552% for DeepSeaNet, 38.374% for MCANet, and 19.374% for QuokkaNet. Lower FPR values are preferable, highlighting QuokkaNet's effectiveness in minimizing false positives. The False Negative Rate (FNR), indicating the proportion of missed positive instances, was 47.397% for DeepSeaNet, 39.164% for MCANet, and 18.375% for QuokkaNet, underscoring QuokkaNet's lower rate of missed positives.

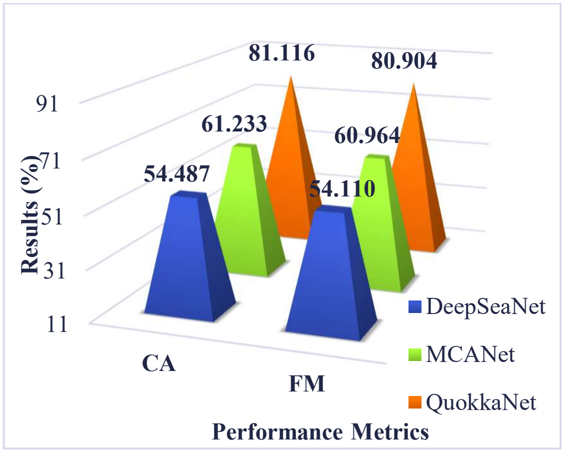

Fig 3. Classification Accuracy and F-Measure.

Precision, the proportion of correct positive identifications among all positive identifications, was 55.706% for DeepSeaNet, 61.093% for MCANet, and 80.195% for QuokkaNet. Higher precision indicates more reliable positive identifications, with QuokkaNet demonstrating superior precision. Classification Accuracy (CA), measuring the overall correctness of classifications, was 54.487% for DeepSeaNet, 61.233% for MCANet, and 81.116% for QuokkaNet, with QuokkaNet showing a significant accuracy advantage.

The F-Measure, combining precision and recall into a single metric, was 54.110 for DeepSeaNet, 60.964 for MCANet, and 80.904 for QuokkaNet. The F-Measure results highlight QuokkaNet's balanced performance in both precision and recall.

Table 1. Classification Accuracy and F-Measure.

Classification Algorithms	CA	FM
DeepSeaNet	54.487	54.110
MCANet	61.233	60.964
QuokkaNet	81.116	80.904

In Table 1, which tabulates the CA and FM values, DeepSeaNet had a CA of 54.487% and an FM of 54.110. MCANet improved with a CA of 61.233% and an FM of 60.964. QuokkaNet achieved the highest performance, with a CA of 81.116% and an FM of 80.904, demonstrating its superior classification capabilities. The results indicate that QuokkaNet outperforms DeepSeaNet and MCANet across all evaluated metrics. QuokkaNet's higher TP and TN values and lower FP and FN rates contribute to its superior TPR, TNR, FPR, FNR, precision, classification accuracy, and F-Measure. The analysis confirms that QuokkaNet significantly improves underwater image classification, making it the most effective model among the three.

5.2. Fowlkes-Mallows Index and Matthews Correlation Coefficient

DeepSeaNet achieved a Fowlkes-Mallows Index (FMI) of 54.132 and a Matthews Correlation Coefficient (MCC) of 9.056. The FMI, which evaluates the geometric mean of precision and recall, indicates moderate performance. The MCC, reflecting the quality of binary classifications, underscores the limited correlation between predicted and actual classes for DeepSeaNet. These results suggest that while DeepSeaNet provides reasonable accuracy, its predictive power remains modest. MCANet improved these results with an FMI of 60.964 and an MCC of 22.463. The higher FMI demonstrates a better balance between precision and recall than DeepSeaNet. The increased MCC signifies a stronger correlation between predictions and actual outcomes, highlighting MCANet's enhanced classification capabilities. This improvement suggests MCANet's greater effectiveness in correctly identifying underwater images.

Fig 4. Illustrates the outcome of the Fowlkes-Mallows Index and Matthews Correlation Coefficient. QuokkaNet surpassed DeepSeaNet and MCANet, achieving an FMI of 80.907 and an MCC of 62.240. The highest FMI among the three indicates QuokkaNet's superior precision-recall balance. The substantial MCC value signifies a robust correlation between predicted and actual classifications, reflecting QuokkaNet's exceptional performance in underwater image classification tasks. These metrics illustrate QuokkaNet's dominance in accurately classifying underwater images.

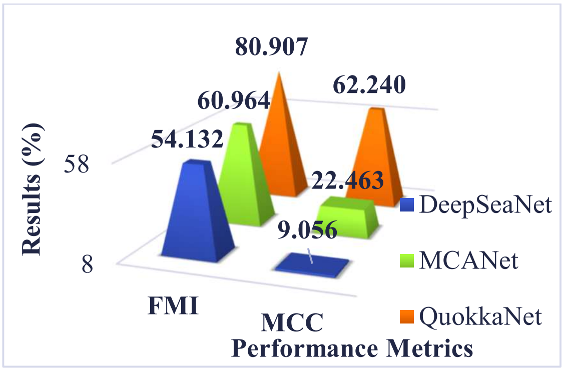

Fig 4. Fowlkes-Mallows Index and Matthews Correlation Coefficient.

The Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) provide critical insights into the models' performance. Analyzing the FMI values of 54.132 for DeepSeaNet, 60.964 for MCANet, and 80.907 for QuokkaNet reveals a progressive improvement in the balance of precision and recall, with QuokkaNet demonstrating the best results. Similarly, the MCC values of 9.056 for DeepSeaNet, 22.463 for MCANet, and 62.240 for QuokkaNet highlight a substantial enhancement in prediction quality, with QuokkaNet showing a significant advantage.

Classification Algorithms	FMI	MCC
DeepSeaNet	54.132	9.056
MCANet	60.964	22.463
QuokkaNet	80.907	62.240

Table 2. Fowlkes-Mallows Index and Matthews Correlation Coefficient

Table 2 illustrates the consolidated FMI and MCC of QuokkaNet. The classification algorithms' FMI and MCC values reiterate the superior performance of QuokkaNet, followed by MCANet, with DeepSeaNet lagging. These comparative metrics underscore QuokkaNet's exceptional ability to classify underwater images, affirming its robustness and reliability accurately. The evaluation metrics comprehensively compare DeepSeaNet, MCANet, and QuokkaNet. The results demonstrate QuokkaNet's superior performance across various metrics, including FMI and MCC. These findings confirm QuokkaNet's effectiveness and accuracy in underwater image classification, making it the most reliable model among the three.

6. CONCLUSION

The comparative analysis of DeepSeaNet, MCANet, and QuokkaNet for underwater image classification tasks reveals significant advancements in performance metrics with QuokkaNet. By leveraging enhanced convolutional neural networks and Quokka Optimization, QuokkaNet demonstrates superior capabilities regarding true positive and true negative rates, precision, classification accuracy, and F-measure. The improved precision-recall balance and strong correlation between predicted and actual outcomes, as indicated by higher Fowlkes-Mallows Index and Matthews Correlation Coefficient values, further affirm QuokkaNet's robustness. Integrating convolutional, pooling, normalization, and fully connected layers and applying Nesterov Accelerated Gradient in the N-CNN model form a solid foundation for effective image classification. Quokka Optimization enhances this model through systematic exploration, exploitation, fitness evaluation, selection, adaptation, and migration, ensuring optimal performance. This study highlights the significant advancements achieved through QuokkaNet in underwater image classification, demonstrating a clear performance edge over DeepSeaNet and MCANet. The results confirm that QuokkaNet's innovative approach and optimization techniques offer a reliable and accurate solution for complex image classification tasks, making it a valuable tool for underwater research and exploration applications.

REFERENCES:

[1] 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.

[2] N. K. Gupta, R. S. Yadav, R. K. Nagaria, D. Gupta, A. M. Tripathi, and O. J. Pandey, "Anchor-based void detouring routing protocol in three dimensional IoT networks," *Comput. Networks*, vol. 227, p. 109691, 2023, doi: 10.1016/j.comnet.2023.109691.

[3] 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.

[4] J. An and W. M. N. Wan Zainon, "Integrating color cues to improve multimodal sentiment analysis in social media," *Eng. Appl. Artif. Intell.*, vol. 126, p. 106874, 2023, doi: 10.1016/j.engappai.2023.106874.

[5] J. Pan, L. Xia, Q. Wu, Y. Guo, Y. Chen, and X. Tian, "Automatic strawberry leaf scorch severity estimation via faster R-CNN and few-shot learning," *Ecol. Inform.*, vol. 70, p. 101706, 2022, doi: 10.1016/j.ecoinf.2022.101706.

[6] D. liang Zhang, Z. Jiang, F. Mohammadzadeh, S. M. Hasani Azhdari, L. Abualigah, and T. M. Ghazal, "FUZ-SMO: A fuzzy slime mould optimizer for mitigating false alarm rates in the classification of underwater datasets using deep convolutional neural networks," *Heliyon*, vol. 10, no. 7, pp. e28681–e28681, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e28681.

[7] E. Essa and I. R. Abdelmaksoud, "Temporal-channel convolution with self-attention network for human activity recognition using wearable sensors," *Knowledge-Based Syst.*, vol. 278, p. 110867, 2023, doi: https://doi.org/10.1016/j.knosys.2023.110867.

[8] A. Mitra, B. Bera, A. K. Das, S. S. Jamal, and I. You, "Impact on blockchain-based AI/ML-enabled big data analytics for Cognitive Internet of Things environment," *Comput. Commun.*, vol. 197, pp. 173–185, 2023, doi: https://doi.org/10.1016/j.comcom.2022.10.010.

[9] Y. Zhou, B. Li, J. Wang, E. Rocco, and Q. Meng, "Discovering unknowns: Context-enhanced anomaly detection for curiosity-driven autonomous underwater exploration," *Pattern Recognit.*, vol. 131, p. 108860, 2022, doi:

https://doi.org/10.1016/j.patcog.2022.108860.

[10] J. Zhang, X. Liu, M. Chen, Q. Ye, and Z. Wang, "Image sentiment classification via multi-level sentiment region correlation analysis," *Neurocomputing*, vol. 469, pp. 221–233, 2022, doi: 10.1016/j.neucom.2021.10.062.

[11] M. Jahanbakht, M. Rahimi Azghadi, and N. J. Waltham, "Semi-supervised and weakly-supervised deep neural networks and dataset for fish detection in turbid underwater videos," *Ecol. Inform.*, vol. 78, p. 102303, 2023, doi: https://doi.org/10.1016/j.ecoinf.2023.102303.

[12] 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.

[13] T. B. N. Freitas, T. S. Leite, B. de Ramos, A. di Cosmo, and M. C. Proietti, "In an octopus's garden in the shade: Underwater image analysis of litter use by benthic octopuses," *Mar. Pollut. Bull.*, vol. 175, p. 113339, 2022, doi: https://doi.org/10.1016/j.marpolbul.2022.113339.

[14] C. Gupta, N. S. Gill, P. Gulia, S. Yadav, and J. M. Chatterjee, "A novel finetuned YOLOv8 model for real-time underwater trash detection," *J. Real-Time Image Process.*, vol. 21, no. 2, p. 48, 2024, doi: 10.1007/s11554-024-01439-3.

[15] V. karthick Perumal, T. Supriyaa, P. R. Santhosh, and D. S. Dhanasekaran, "CNN BASED PLANT DISEASE IDENTIFICATION USING PYNQ FPGA," *Syst. Soft Comput.*, p. 200088, 2024, doi: https://doi.org/10.1016/j.sasc.2024.200088.

[16] 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: 10.1016/j.neucom.2021.04.140.

[17] M. Wang, B. Fu, J. Fan, Y. Wang, L. Zhang, and C. Xia, "Sweet potato leaf detection in a natural scene based on faster R-CNN with a visual attention mechanism and DIoU-NMS," *Ecol. Inform.*, vol. 73, p. 101931, 2023, doi: 10.1016/j.ecoinf.2022.101931.

[18] H. Li, Y. Liu, Z. Ye, Q. Zhang, S. Yang, and M. Xu, "Underwater target detection using hybrid carbon nanotube self-adhesive sensors," *Device*, vol. 2, no. 1, p. 100223, 2024, doi: https://doi.org/10.1016/j.device.2023.100223.

[19] O. Vidhya and S. Ranjitha Kumari, "Quadratic ensemble weighted emphasis boosting based energy and bandwidth efficient routing in Underwater Sensor Network," *Int. J. Intell. Networks*, vol. 4, pp. 130–139, 2023, doi: 10.1016/j.ijin.2023.05.001.

[20] X. Hua *et al.*, "Underwater object detection algorithm based on feature enhancement and progressive dynamic aggregation strategy," *Pattern Recognit.*, vol. 139, p. 109511, 2023, doi: https://doi.org/10.1016/j.patcog.2023.109511.

[21] Z. Gao, Y. Shi, and S. Li, "Self-attention and long-range relationship capture network for underwater object detection," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 2, p. 101971, 2024, doi: https://doi.org/10.1016/j.jksuci.2024.101971.

[22] P. Song, P. Li, L. Dai, T. Wang, and Z. Chen, "Boosting R-CNN: Reweighting R-CNN samples by RPN's error for underwater object detection," *Neurocomputing*, vol. 530, pp. 150–164, 2023, doi: https://doi.org/10.1016/j.neucom.2023.01.088.

[23] C. Liang *et al.*, "Identification and maximum impact force modeling investigation for critical slugging in underwater compressed gas energy storage systems," *J. Energy Storage*, vol. 67, p. 107550, 2023, doi: https://doi.org/10.1016/j.est.2023.107550.

[24] C. G. F. Zinzindohoué, T. Schoening, E. B. Lima, and B. Fiedler, "PlasPi TDM: Augmentation of a low-cost camera platform for advanced underwater physical-ecological observations," *HardwareX*, vol. 15, pp. e00470–e00470, 2023, doi: https://doi.org/10.1016/j.ohx.2023.e00470.

[25] B. Huang, G. Chen, H. Zhang, G. Hou, and M. Radenkovic, "Instant deep sea debris detection for maneuverable underwater machines to build sustainable ocean using deep neural network," *Sci. Total Environ.*, vol. 878, p. 162826, 2023, doi: https://doi.org/10.1016/j.scitotenv.2023.162826.

[26] A. Al Muksit, F. Hasan, M. F. Hasan Bhuiyan Emon, M. R. Haque, A. R. Anwary, and S. Shatabda, "YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment," *Ecol. Inform.*, vol. 72, p. 101847, 2022, doi: https://doi.org/10.1016/j.ecoinf.2022.101847.

[27] T. Yoshida, J. Zhou, K. Terayama, and D.

Kitazawa, “Monitoring of cage-cultured sea cucumbers using an underwater time-lapse camera and deep learning-based image analysis,” *Smart Agric. Technol.*, vol. 3, p. 100087, 2023, doi: https://doi.org/10.1016/j.atech.2022.100087.

[28] D. A. K and N. Keshaveni, “An AVOA-LSTM with MRCNN for segmenting and classifying the sunglass image-based eye region identification,” *Multimed. Tools Appl.*, vol. 83, no. 12, pp. 35073–35095, 2024, doi: 10.1007/s11042-023-16800-0.

[29] Y. Zhang, Z. Yang, X. Du, and X. Luo, “A New Method for Denoising Underwater Acoustic Signals Based on EEMD, Correlation Coefficient, Permutation Entropy, and Wavelet Threshold Denoising,” *J. Mar. Sci. Appl.*, vol. 23, no. 1, pp. 222–237, 2024, doi: 10.1007/s11804-024-00386-6.

[30] M. Khishe, M. Mohammadi, and A. Ramezani Varkani, “Underwater Backscatter Recognition Using Deep Fuzzy Extreme Convolutional Neural Network Optimized via Hunger Games Search,” *Neural Process. Lett.*, vol. 55, no. 4, pp. 4843–4870, 2023, doi: 10.1007/s11063-022-11068-1.

[31] X. Zhou, K. Yang, and R. Duan, “Deep Learning Based on Striation Images for Underwater and Surface Target Classification,” *IEEE Signal Process. Lett.*, vol. 26, no. 9, pp. 1378–1382, 2019, doi: 10.1109/LSP.2019.2919102.

[32] Y.-P. Huang and S. P. Khabusi, “A CNN-OSELM Multi-Layer Fusion Network With Attention Mechanism for Fish Disease Recognition in Aquaculture,” *IEEE Access*, vol. 11, pp. 58729–58744, 2023, doi: 10.1109/ACCESS.2023.3280540.

[33] 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.

[34] 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.

[35] K. S. J. Marseline, J. Ramkumar, and D. R. Medhunhashini, “Sophisticated Kalman Filtering-Based Neural Network for Analyzing Sentiments in Online Courses,” in *Smart Innovation, Systems and Technologies*, A. K. Somani, A. Mundra, R. K. Gupta, S. Bhattacharya, and A. P. Mazumdar, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 345–358. doi: 10.1007/978-981-97-3690-4_26.

[36] 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.

[37] 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.

[38] 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.

[39] R. Jaganathan, S. Mehta, and R. Krishan, *Bio-Inspired intelligence for smart decision-making*, vol. i. 2024. doi: 10.4018/9798369352762.

[40] 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.

[41] J. Ramkumar, R. Karthikeyan, and M. Lingaraj, “Optimizing IoT-Based Quantum Wireless Sensor Networks Using NM-TEEN Fusion of Energy Efficiency and Systematic Governance,” in *Lecture Notes in Electrical Engineering*, V. Shrivastava, J. C. Bansal, and B. K. Panigrahi, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 141–153. doi: 10.1007/978-981-97-6710-6_12.

[42] 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.

[43] J. Ramkumar, R. Karthikeyan, and V. Valarmathi, “Alpine Swift Routing Protocol (ASRP) for Strategic Adaptive Connectivity Enhancement and Boosted Quality of Service in Drone Ad Hoc Network (DANET),” *Int. J.*

Comput. Networks Appl., vol. 11, no. 5, pp. 726–748, 2024, doi: 10.22247/ijcna/2024/45.

[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] 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.

[46] 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.

[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.

[48] R. Jaganathan, S. Mehta, and R. Krishan, *Intelligent Decision Making Through Bio-Inspired Optimization*. Sri Krishna Arts and Science College, India: IGI Global, 2024. doi: 10.4018/979-8-3693-2073-0.

[49] 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," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–6, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9752899.

[50] 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.

[51] 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.

[52] 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.

[53] 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

[54] 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.

[55] 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.

[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] 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.

[58] 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.

[59] M. P. Swapna, J. Ramkumar, and R. Karthikeyan, "Energy-Aware Reliable Routing with Blockchain Security for Heterogeneous Wireless Sensor Networks," in *Lecture Notes in Networks and Systems*, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., Springer Science and Business Media

Deutschland GmbH, 2025, pp. 713–723. doi: 10.1007/978-981-97-6106-7_43.

[60] 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.

[61] 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.

[62] M. R. Azimi-Sadjadi, J. Salazar, and S. Srinivasan, "An Adaptable Image Retrieval System With Relevance Feedback Using Kernel Machines and Selective Sampling," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1645–1659, 2009, doi: 10.1109/TIP.2009.2017825.

[63] H. Wang, F. Dong, and L. Song, "Bubble-Forming Regime Identification Based on Image Textural Features and the MCWA Feature Selection Method," *IEEE Access*, vol. 5, pp. 15820–15830, 2017, doi: 10.1109/ACCESS.2017.2716783.

[64] 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.

[65] 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.