International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I6P119

Volume 72 Issue 6, 178-194, June 2024

Original Article

Persistent Fish School-Inspired Deep Belief Network for Object Detection in Varied Weather Traffic Surveillance

V. Valarmathi[1], S. Dhanalakshmi[2]

[1]*Department of Information Technology, Sri Krishna Arts and Science College, Tamil Nadu, India.*
[2]*Department of Software Systems, Sri Krishna Arts and Science College, Tamil Nadu, India.*

[1]*Corresponding Author : valarskasc1@gmail.com*

Received: 06 February 2024 Revised: 14 May 2024 Accepted: 24 May 2024 Published: 29 June 2024

Abstract *- Traffic surveillance is pivotal in ensuring public safety and efficient urban mobility. With the continuous improvements in computer vision, surveillance systems can now identify things automatically in real-time, greatly expanding their possibilities. However, the challenges associated with object detection, particularly in diverse weather conditions, pose a considerable obstacle. Adverse weather elements, such as rain and snow, can impede the accuracy of detection algorithms, impacting the overall effectiveness of traffic surveillance systems. This research addresses these challenges by introducing the Persistent Fish School Search-Inspired Deep Belief Network (PFSS-DBN), a novel algorithm designed to bolster object detection in varying weather climates. Inspired by fish schools' persistent and adaptive nature, PFSS-DBN leverages deep belief networks to navigate complex visual data. The algorithm dynamically adapts its parameters, optimizing its performance for weather scenarios. This adaptability enhances detection accuracy and ensures reliable surveillance outcomes even in challenging conditions. The study employs the AAU RainSnow Traffic Surveillance Dataset to evaluate the proposed PFSS-DBN algorithm. Through comprehensive experimentation, the results demonstrate the superior performance of PFSS-DBN compared to traditional methods, showcasing its efficacy in mitigating the impact of adverse weather on object detection. The findings underscore the potential of PFSS-DBN as a valuable solution for improving the reliability of traffic surveillance systems, particularly in regions prone to diverse weather conditions.*

Keywords *- Adaptive parameter optimization, Nature-Inspired computing, Object detection, PFSS-DBN, Traffic surveillance, Weather-Adaptive algorithms.*

1. Introduction

Traffic surveillance is a cornerstone of modern urban infrastructure, playing a pivotal role in ensuring road safety, optimizing traffic flow, and responding swiftly to incidents. Leveraging a spectrum of technologies, including advanced cameras, sensors, and data analytics, these systems provide real-time insights into traffic conditions [1]. Its primary goal is improving road safety by detecting and resolving key situations such as accidents, obstacles, and abnormal driving behaviours. The authorities' rapid response during emergencies is greatly enhanced by strategically positioned surveillance cameras at roads, junctions, and urban hubs. These cameras can prevent or at least mitigate the effects of accidents. In addition to ensuring everyone's safety, traffic surveillance is crucial for reducing gridlock. Authorities may improve traffic flow by collecting and analyzing data, which allows them to fine-tune traffic signal timings, deploy alternate routes, and make educated judgements [2], [3]. Predictive analytics, made possible using state-of-the-art technologies like machine learning and artificial intelligence, improve these systems. This helps decision-makers with urban planning and traffic control. Traffic surveillance systems prove indispensable in adverse weather conditions, such as rain, snow, or fog. Equipped with features like self-cleaning lenses, thermal imaging, and radar technology, they overcome visibility challenges and assist authorities in making informed decisions to ensure road safety. Overall, traffic surveillance is a dynamic and essential component of contemporary cities, fostering safer roads, efficient traffic management, and the sustainable development of urban landscapes [4].

Object detection in traffic surveillance, especially in varied weather conditions, presents a critical challenge that demands innovative solutions. Adverse weather, such as rain, snow, fog, or varying lighting conditions, can significantly impact the accuracy of object detection systems [5]. Traditional surveillance methods may struggle to distinguish and track objects effectively under these challenging circumstances. Innovation merges state-of-the-art computer vision methods with machine learning algorithms and deep learning models to overcome these obstacles. Faster R-CNN (Region-based Convolutional Neural Network) and YOLO

(You Only Look Once) are two object identification frameworks that these systems use to improve accuracy and real-time processing [6]. In varied weather conditions, deploying specialized sensors, such as thermal cameras and radar systems, becomes crucial. Thermal imaging helps overcome visibility issues caused by fog or low-light situations, while radar technology provides additional data for accurate object detection [7].

Machine learning models trained on diverse datasets that include different weather scenarios contribute to the adaptability and robustness of these systems. Bio-inspired optimization techniques, mimicking nature's evolutionary processes, have shown promise in refining object detection precision [8]. These methods adapt in real time to environmental challenges, making them well-suited for the unpredictability of weather conditions. The continuous evolution of object detection technologies in traffic surveillance reflects a commitment to creating safer roads by addressing the complexities introduced by varied weather conditions. As these innovations mature, they hold the potential to improve the reliability and effectiveness of traffic surveillance systems significantly, ensuring road safety across diverse climates [9].

1.1. Problem Statement

The efficacy of object classification in traffic surveillance is considerably hindered by the dual challenges posed by complex backgrounds and occlusions. Complex backgrounds encompass a broad spectrum of scenarios, from bustling urban environments with intricate architectural details to densely vegetated regions with foliage and clutter, where the objects of interest, such as vehicles and pedestrians, often become visually entwined with their surroundings. This results in a high degree of variability in the appearance of these objects, making it challenging for traditional object classification systems to accurately discriminate between the pertinent objects and the extraneous elements in the scene. The consequence can be the generation of false alarms (false positives) or the failure to detect critical objects (false negatives), potentially leading to erroneous decisions in traffic surveillance applications.

On the other hand, occlusions frequently manifest in congested traffic conditions, intersections, or during overtaking manoeuvres, obscuring parts or entire objects and thereby creating visual ambiguities that confound the object recognition process. These occlusions can result in tracking errors, missed object classifications, and unreliable traffic data, which are detrimental to functions like traffic flow analysis and collision avoidance systems. Effectively addressing the intertwined challenges of complex backgrounds and occlusions necessitates the development of innovative solutions that enhance the adaptability and robustness of object classification algorithms. These solutions should enable the algorithms to discern objects within intricate and cluttered backdrops and accurately classify objects when they are partially or entirely occluded. By conquering these hurdles, traffic surveillance systems can offer heightened reliability and safety, contributing significantly to efficient traffic management, accident prevention, and overall road safety.

1.2. Research Gap

Despite significant advancements in object classification for traffic surveillance, existing systems continue to struggle with complex backgrounds and occlusions. These challenges lead to frequent inaccuracies, such as false positives and missed detections, particularly in urban and densely vegetated environments or during traffic congestion. Current approaches have not fully addressed the need for algorithms that can robustly and adaptively distinguish objects in cluttered scenes and maintain accuracy when objects are partially or fully obscured. This gap underscores the necessity for innovative solutions to enhance the reliability and effectiveness of traffic surveillance systems across diverse conditions.

1.3. Motivation

The motivation to tackle the challenges of complex backgrounds and occlusions in object classification for traffic surveillance is rooted in the profound impact it can have on the safety and efficiency of our roadways. With the ever-increasing volume of vehicles and pedestrians navigating our streets, highways, and urban areas, accurate and reliable traffic surveillance is more critical than ever. Complex backgrounds, ranging from bustling urban landscapes to lush natural environments, demand advanced object classification techniques to differentiate between relevant objects and distracting elements, ensuring precise traffic data analysis and real-time decision-making. Likewise, occlusions are prevalent in our dynamic traffic scenarios, presenting significant risks when not addressed effectively. By developing robust solutions that empower object classification algorithms to thrive in the face of these challenges, we can enhance the accuracy of traffic surveillance systems, contributing to smoother traffic flow, proactive accident prevention, and, ultimately, saving lives. Such advancements' societal and economic benefits are immense, making this endeavour a compelling and crucial pursuit in computer vision and transportation safety.

1.4. Objectives

The primary objective in addressing the challenges associated with complex backgrounds and occlusions within the domain of object classification for traffic surveillance is to significantly enhance the reliability, accuracy, and effectiveness of traffic monitoring and management systems. The aim is to develop advanced algorithms and methodologies capable of robustly identifying objects amidst intricate and cluttered backgrounds. This ensures that traffic data analysis is precise and facilitates well-informed decision-making for optimizing traffic flow and accident prevention.

Simultaneously, the goal is to empower object classification systems to effectively handle occlusions, reducing tracking errors and ensuring consistent object recognition even when objects are partially or entirely obscured. This will result in more dependable traffic surveillance systems contributing to safer roadways, reduced traffic congestion, and more efficient transportation networks. Additionally, the objective is to bolster the adaptability and versatility of these algorithms to perform reliably across diverse environmental conditions, from urban to rural settings, further increasing their utility and impact. Ultimately, the central mission is to harness cutting-edge computer vision techniques to revolutionize traffic surveillance, making roadways safer, smarter, and more efficient for the benefit of society.

2. Literature Review

"AIS Traffic Anomaly Review"[10] offers a comprehensive review of AIS-based methods for detecting anomalies in maritime traffic, contributing to the enhancement of maritime traffic monitoring and safety. "Deep Learning in Agricultural Surveillance"[11] utilizes deep learning techniques for agricultural monitoring within video surveillance, improving crop management and yield predictions, which can benefit the agricultural industry. Simulating the spatiotemporal distribution of traffic loads on highway bridges using a combination of camera footage data. "Traffic Load Fusion for Bridge Safety" [12] aids in improved traffic management and bridge safety. "Small-Object Detection in Autonomous Driving" [13]uses YOLOv5 to improve road safety and object identification accuracy for autonomous driving systems.

To improve traffic management and safety in real-time, "Lightweight Backbone" [14] accomplishes dense traffic detection in real-time using a lightweight backbone and an upgraded path aggregation feature pyramid network. "Abnormal Event Detection"[15] introduces an enhanced two-stream fusion method for detecting abnormal events in video surveillance, improving security and incident response in surveillance applications. "Surveillance System Placement with 3D Scanning"[16] optimizes the placement of video surveillance systems using 3D scanning technology, enhancing traffic safety through strategically positioned surveillance cameras. "Lightweight Small Object Detection"[17] presents an improved lightweight framework for small object detection in real-time autonomous driving, enhancing safety and efficiency on the road.

"TransCNN for Anomaly Detection"[18] combines CNN and transformer mechanisms in TransCNN for surveillance anomaly detection, improving the identification of unusual activities and bolstering security in monitored areas. "Vehicle Detection for Traffic Scheduling"[5] focuses on vehicle detection from road image sequences to support intelligent traffic scheduling, enhancing traffic flow and management. "Traffic Perception from Aerial Images"[19] utilizes butterfly fields to analyze aerial images for traffic perception, advancing remote traffic monitoring capabilities. Contributing to public safety and security, "Remote Monitoring for Anti-Social Activity" [20]presents a model for remote monitoring using slow-fast deep convolution neural networks. This model enables the identification of anti-social actions in surveillance applications. Bio-inspired optimization plays a crucial role in research to attain better results [21]-[42]. "Dolphin Swarm Object Detection (DSOD)"[43] attempts to enhance automatic object detection and classification in surveillance videos. It optimizes the object recognition process without bias towards specific outcomes, improving accuracy and efficiency.

This approach promises to advance the field of surveillance technology by offering more reliable and precise object recognition, which has implications for enhancing security and surveillance applications. "DenseYOLO"[44] improves the YOLOv2 model for vehicle detection in surveillance videos. It leverages DenseNet-201 to streamline feature extraction and reduce model complexity. The dense architecture of DenseNet-201 enhances the extraction of image information. It provides an improved equilibrium between accuracy and model size, which might lead to a leap forward in vehicle recognition in surveillance applications compared to current approaches.

3. Persistent Fish School-Inspired Deep Belief Network

The Persistent Fish School-Inspired Deep Belief Network (PFSS-DBN) is a novel computational approach that draws inspiration from the coordinated and adaptive behaviour of persistent fish schools in nature. This algorithm harnesses the collective intelligence and resilience observed in fish schools to enhance the performance of Deep Belief Networks (DBNs). PFSS-DBN adapts to dynamic environmental conditions, particularly excelling in varied weather climates encountered in traffic surveillance scenarios. Its innovative integration of nature-inspired computing principles contributes to improved object detection accuracy, making it a promising solution for challenging surveillance applications where traditional methods face difficulties.

3.1. Enhanced Deep Belief Network

DBNs are a type of neural network with multiple layers of hidden units capable of learning intricate hierarchical representations of data. The "enhanced" version suggests modifications, advancements, or additional features incorporated into the DBN architecture to overcome specific challenges or improve performance. Enhancements can take various forms, including introducing novel activation functions, optimization algorithms, regularization techniques, or architectural adjustments. The goal is typically to boost the network's learning capabilities, generalization, and efficiency in handling complex patterns within datasets. The enhancements to a DBN depend on the targeted application or the challenges prevalent in a particular domain.

3.1.1. Initialization

The initialization of a Restricted Boltzmann Machine (RBM) marks the inaugural step in constructing an Enhanced Deep Belief Network (EDBN). The RBM is the foundational building block, capturing intricate relationships within the input data. At this stage, the RBM is designed as a bipartite graphical model consisting of visible units v and hidden units h. An energy function, written as $E(v,h)$, shows how the visual and secret units are spread out together:

$$E(v,h) = -\sum_i \sum_j w_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \quad (1)$$

In this context, w_{ij} stands for the weights that link visible unit i with hidden unit j,a_i for the bias linked to visible unit i, and b_j for the bias linked to hidden unit j. The energy function characterizes the compatibility between the visible and hidden units. The Combined Allocation and the Boltzmann distribution, which is based on the energy function, are used to define $P(v,h)$.

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \quad (2)$$

To guarantee that the distribution always adds up to 1 for all conceivable arrangements of visible and hidden units, we use a normalization constant denoted by Z. To facilitate efficient learning, the RBM employs a stochastic binary activation function. The probability of a hidden unit h_j being activated given visible units v is given by the logistic sigmoid function expressed as Equation (3).

$$P(h_j = 1|v) = \frac{1}{1 + e^{-(\sum_i w_{ij} v_i + b_j)}} \quad (3)$$

The probability of a visible unit v_i being activated given hidden units h is expressed as Equation (4).

$$P(v_i = 1|h) = \frac{1}{1 + e^{-(\sum_j w_{ij} h_j + a_i)}} \quad (4)$$

This stochastic activation enables the RBM to model complex data distributions efficiently and extract relevant features during the pre-training phase of the EDBN.

3.1.2. Stack RBMs to Form Deep Architecture

This process, commonly known as pre-training, facilitates the hierarchical learning of features from raw input data. The RBMs are arranged in layers, with the hidden units of the preceding RBM serving as the visible units for the subsequent one. The EDBN can learn complex hierarchical representations of the incoming data through this stacking process, which produces a progressive abstraction hierarchy. The units that can be seen and those that cannot are represented by $v^{(k)}$and $h^{(k)}$ accordingly in the k-th RBM. The weights used to connect the visible and hidden layers are $W^{(k)}$, while the biases for the visible and hidden layers are $a^{(k)}$and $b^{(k)}$ correspondingly. The following energy function, $E^{(k)}(v^{(k)}, h^{(k)})$: indicates the joint distribution for the k-th RBM:

$$E^{(k)}(v^{(k)}, h^{(k)}) = \sum_i \sum_j w_{ij}^{(k)} v_i^{(k)} h_j^{(k)} - \sum_i a_i^{(k)} v_i^{(k)} - \sum_j b_j^{(k)} h_j^{(k)} \quad (5)$$

The k-th RBM's hidden and visible layers interact with each other through this energy function. Equation (6) determines the probability distribution for exposed and concealed units.

$$P(v^{(k)}, h^{(k)}) = \frac{e^{-E(k)}(v^{(k)}, h^{(k)})}{Z^{(k)}} \quad (6)$$

Where $Z^{(k)}$is the partition function ensuring the normalization of the distribution. The activation probability for hidden units given visible units is determined by Equation (7), which is the logistic sigmoid function.

$$P(h_j^{(k)} = 1|v^{(k)}) = \frac{1}{1 + e^{-(\sum_i w_{ij}^{(k)} v_i^{(k)} + b_j^{(k)})}} \quad (7)$$

The activation probability for visible units given hidden units is expressed as Equation (8).

$$P(v_i^{(k)} = 1|h^{(k)}) = \frac{1}{1 + e^{-(\sum_j w_{ij}^{(k)} h_j^{(k)} + a_i^{(k)})}} \quad (8)$$

This procedure is repeated for every RBM that follows it, with the visible layer of the $(k+1)$th RBM being the hidden layer of the k-th RBM. This iterative stacking establishes a deep architecture, allowing the EDBN to progressively learn hierarchical representations of the input data, thereby capturing complex patterns and features.

3.1.3. EDBN Initialization

This step involves initializing the EDBN using the weights and biases obtained from the pre-trained RBMs. Denoting the weights, biases, visible, and hidden units of the EDBN as $W, a, v,$ and $h$respectively, the initialization builds upon the structure established in the previous steps. For the k-th RBM, the weights and biases are denoted as $w^{(k)}, a^{(k)}$ and $b^{(k)}$.The transition from the k-th RBM to the $(k+1)$th RBM involves setting the visible units $v^{(k+1)}$equal to the hidden units $h^{(k)}$of the k-th RBM:

$$v^{(k+1)} = h^{(k)} \quad (9)$$

This relationship ensures the continuity of information flows from one layer to the next. The energy function for the EDBN, considering all the RBM layers, is given by Equation (10).

$$E(\mathrm{v}, \mathrm{h}) = \sum_k \sum_i \sum_j w_{ij}^{(k)} v_i^{(k)} h_j^{(k)} - \sum_k \sum_i a_i^{(k)} v_i^{(k)} - \sum_k \sum_j b_j^{(k)} h_j^{(k)} \quad (10)$$

The joint distribution $P(v,h)$ is then defined using the Boltzmann distribution, ensuring the normalization of the distribution, and it is expressed as Equation (11).

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \quad (11)$$

Where Z is the partition function.

The activation probabilities for the hidden units given the visible units, and vice versa, are determined using Equations (12) and (13), which is the logistic sigmoid function, maintaining the stochastic nature of the network.

$$P\left(h_j^{(k)} = 1 | v^{(k)}\right) = \frac{1}{1 + e^{-\left(\sum_i w_{ij}^{(k)} v_i^{(k)} + b_j^{(k)}\right)}} \quad (12)$$

$$P\left(v_i^{(k)} = 1 | h^{(k)}\right) = \frac{1}{1 + e^{-\left(\sum_j w_{ij}^{(k)} h_j^{(k)} + a_i^{(k)}\right)}} \quad (13)$$

Algorithm 1: EDBN Initialization

Input:

- Set of pre-trained RBMs with weights $W^{(k)}$, biases, and $a^{(k)}$ and $b^{(k)}$for each layer k.

Output:

- Initialized EDBN with weights W, biases a, visible units v, and hidden units h.

Procedure:

Step 1: Initialize empty sets $W, a, v,$ and h for the EDBN.

Step 2: For each RBM layer k in the pre-trained RBMs:

- Add the weights $W^{(k)}$, biases $a^{(k)}$ and $b^{(k)}$to the corresponding sets $W, a,$ and h for the EDBN.

Step 3: Set the visible units v of the EDBN equal to the visible units $v^{(1)}$ of the first RBM.

Step 4: Set the hidden units h h of the EDBN equal to the hidden units $h^{(K)}$ of the last RBM.

3.1.4. Network Fine Tuning

Fine-tuning leverages supervised learning to adjust the weights and biases of the EDBN, refining its ability to map input features to the correct output labels. This process builds upon the structure established in the preceding steps, particularly the pre-training of individual RBMs and the initialization of the EDBN. Denote the weights, biases, visible units, and hidden units of the EDBN as $W, a, v,$ and $h,$ respectively. The energy function $E(v,h)$ for the entire EDBN remains consistent with the definition in the pre-training steps. The joint distribution $P(v,h)$ is defined using Equation (14).

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \quad (14)$$

The distribution is guaranteed to be normalized by Z, the partition function. In this step, this research introduces the concept of labelled data. Let y represent the output labels corresponding to the input data v. The objective is to maximize the conditional probability of the labels given the input data, expressed as Equation (15).

$$P(y|v) = \frac{P(v,y)}{P(v)} \quad (15)$$

Applying Bayes' rule, this conditional probability is further expanded as Equation (16).

$$P(y|v) = \frac{P(v,y)P(y)}{P(v)} \quad (16)$$

Algorithm 2: Network Fine Tuning

Input:

- Initialized Deep Belief Network (EDBN) with weights W, biases a, visible units v, and hidden units h.
- Labelled training dataset: Input data v and corresponding output labels y.

Output:

- Fine-tuned EDBN with adjusted weights W and biases a to improve prediction accuracy.

Procedure:

Step 1: Forward Pass

- Propagate the input data v through the EDBN to compute the predicted output probabilities.
- Calculate the log-likelihood of the labelled data using the predicted probabilities and the actual output labels y.

Step 2: Backward Pass

- Determine the log-likelihood gradients for the EDBN's weights and biases.
- Through the process of optimizing weights and biases, the negative log-likelihood may be diminished. Through the process of optimizing weights and biases, the negative log-likelihood may be reduced.

Step 3: Repeat

- Iterate steps 1 and 2 for a predefined number of epochs or until convergence.
- Monitor the performance on a validation set to avoid overfitting.

The term $P(y)$ represents the prior probability of the output labels, $P(v \mid y)$ is the likelihood of the input data given the labels, and $P(v)$ is the marginal probability of the input data. The fine-tuning process aims to maximize the log-likelihood of the labelled data, which can be expressed as Equation (17), and it is the log probability of the joint distribution.

$$log\ P(v,y) = log\ \frac{e^{-E(v,h)}}{Z} \quad (17)$$

Applying properties of logarithms and rearranging terms, Equation (17) can be simplified as Eqution (18).

$$log\ P(v,y) = E(v,h) - \log Z \qquad (18)$$

The negative log-likelihood, which serves as the objective function for fine-tuning, is given by Equation (19).

$$-log\ P(v,y) = E(v,h) + \log Z \qquad (19)$$

3.1.5. Backpropagation

The backpropagation optimizes the EDBN's parameters for specific tasks, and it refines the network further by adjusting the weights and biases based on the calculated gradients of the loss function concerning these parameters. The loss function, denoted as $J(W,a)$, measures how far the actual labels deviate from the anticipated ones. The goal is to minimize the negative log-likelihood or an analogous loss function, which continues the fine-tuning stage. The loss function is expressed as Equation (20).

$$K(W,a) = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} y_i^{(n)} log\left(\hat{y}_i^{(n)}\right) \qquad (20)$$

Where N is the total number of samples in the dataset, C is the total number of classes, $y_i^{(n)}$ is the actual label, and $\hat{y}_i^{(n)}$ is the probability that sample n will belong to class i. With the help of the weights W and the biases a, the backpropagation algorithm determines the gradient of the loss function. Utilizing the chain rule, the gradients are calculated using Equation (21).

$$\frac{\partial J}{\partial W_{:}^{(k)}} = -\frac{1}{N}\sum_{n=1}^{N}\left(\frac{y_i^{(n)}}{\hat{y}_{:}^{(n)}} - \frac{\left(1-y_i^{(n)}\right)}{1-\hat{y}_{:}^{(n)}}\right)\frac{\partial y_i^{(n)}}{\partial W_{:}^{(k)}} \qquad (21)$$

$$\frac{\partial J}{\partial a_i^{(k)}} = -\frac{1}{N}\sum_{n=1}^{N}\left(\frac{y_i^{(n)}}{\hat{y}_i^{(n)}} - \frac{\left(1-y_i^{(n)}\right)}{1-\hat{y}_i^{(n)}}\right)\frac{\partial \hat{y}_i^{(n)}}{\partial a_i^{(k)}} \qquad (22)$$

Gradient descent changes the weights and biases based on these slopes. The continuity from the fine-tuning step to backpropagation is evident in the shared objective of minimizing the loss function.

3.1.6. Updating Weights

This step iteratively adjusts the weights to minimize the loss function and enhance the network's performance in making accurate predictions. Let $W_{ij}^{(k)}$ be the weight that links neuron i in layer k to neuron j in layer $k+1$, and let α be the learning rate. The weight update is performed through an optimization algorithm, gradient descent. The updated weight $\Delta W_{ij}^{(k)}$ for each connection is expressed in Equation (23)

$$\Delta W_{ij}^{(k)} = -\alpha\frac{\partial J}{\partial W_{ij}^{(k)}} \qquad (23)$$

Where, $\frac{\partial J}{\partial W_{ij}^{(k)}}$ represents the gradient of the loss function concerning the weight $\partial W_{ij}^{(k)}$, as computed during backpropagation. Applying Equation (23) to each weight in the network ensures that the weights are adjusted in the direction that minimizes the loss function. This process is iterated for multiple epochs to progressively refine the network's ability to predict output labels accurately. The weight update equation can be expressed more broadly as Equation (24).

$$W_{ij}^{(k)} \leftarrow W_{ij}^{(k)} + \Delta W_{ij}^{(k)} \qquad (24)$$

Equation (24) symbolizes the iterative nature of weight updates, where the weights are continuously refined in the direction that reduces the overall loss. One of the most essential hyperparameters to tune while updating the weights is the learning rate (α). The weight update process intends to minimize the loss function (J), ensuring that the network's predictions align closely with the true labels in the training dataset.

3.1.7. Fine-Tuning

Fine-tuning is an iterative process, and determining whether additional iterations are necessary depends on the convergence and accuracy achieved during the training process. The loss function, denoted by $J(W,a)$, is the difference between the expected and actual labels. The continuity from the previous steps is maintained, as the objective remains to minimize this loss function. The loss function is expressed as Equation (25).

Algorithm 3: Updating Weights

Input:

- Gradients of the loss function concerning the weights: $\frac{\partial J}{\partial W_{ij}^{(k)}}$ for all weights $W_{ij}^{(k)}$ in the network.
- Learning rate (α).

Output:

- Updated weights $W_{ij}^{(k)}$ for all connections in the network.

Procedure:

Step 1: Set up the weights $W_{ij}^{(k)}$ for each network link.

Step 2: For each weight $W_{ij}^{(k)}$, compute the weight update $\Delta W_{ij}^{(k)}$

Step 3: Update each weight $W_{ij}^{(k)}$

Step 4: Repeat steps 2-3 for all weights in the network.

Step 5: Repeat the entire process for a predefined number of epochs or until convergence.

$$J(W,a) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} y_i^{(n)} log\left(\hat{y}_i^{(n)}\right) \qquad (25)$$

The sentence may be paraphrased as follows: The number of samples in the dataset is N, the number of classes is C, the

true label is $y_i^{(n)}$ and the projected probability for class i in sample n is $\hat{y}_i^{(n)}$. The decision to repeat the fine-tuning process is contingent upon assessing the convergence of the training process and the accuracy of validation data achieved. The decision can be formulated using a condition based on predefined criteria, such as a target accuracy or a convergence threshold. This condition reflects a logical decision-making process. The fine-tuning process is repeated if the network has not converged or the validation accuracy falls below the target accuracy. Otherwise, the training is stopped, indicating that the network has reached a satisfactory state. Algorithm 4 provides a logical decision-making process to determine whether to repeat the fine-tuning process based on the convergence state and validation accuracy. It ensures that the training continues until the network achieves satisfactory convergence and accuracy on the validation dataset.

Algorithm 4: Fine-Tuning

Input:
- The current state of the EDBN
- Validation dataset
- Convergence criteria.

Output:
- Decision on whether to repeat the fine-tuning process.

Procedure:

Step 1: Train the EDBN using the fine-tuning process.

Step 2: Assess the convergence state by evaluating predefined criteria, including but not limited to the loss function or training epoch increments.

Step 3: Evaluate the accuracy of the EDBN on the validation dataset.

Step 4: Check whether the convergence criteria are unmet or the validation accuracy is below the target.
- If true, repeat the fine-tuning process and return to step 1.
- If false, stop the training process.

This comprehensive algorithm outlines the step-by-step process of training a Deep Belief Network, starting from pre-training with unlabelled data, then fine-tuning with labelled data, updating weights, and repeating fine-tuning if necessary. The algorithm ensures the network's iterative refinement until convergence and satisfactory accuracy are achieved.

3.2. Persistent Fish School Search

An optimization method that takes cues from how fish schools act collectively in nature is called Persistent Fish School Search (PFSS). Like other nature-inspired algorithms, PFSS leverages the principles of swarm intelligence to solve optimization problems. In the case of PFSS, the algorithm mimics fish schools' persistent and adaptive nature in their search for resources. The term "persistent" in PFSS highlights the algorithm's emphasis on sustained exploration and exploitation of the search space. This quality is precious in optimization tasks where a balance between exploring new solutions and exploiting promising ones is crucial for finding optimal or near-optimal solutions.

3.2.1. Initialization

In the initialization step of PFSS, the algorithm sets the groundwork for the optimization process by defining crucial parameters and initializing the state of the fish population and persistent memory structures.

Algorithm 5: EDBN

Input:
- The unlabelled training dataset for pre-training.
- Labelled training dataset for fine-tuning.
- Validation dataset for assessing convergence and accuracy.
- Hyperparameters: learning rate, number of hidden layers, units in each layer, convergence criteria, and target accuracy.

Output:
- Trained EDBN with optimized weights and biases.

Procedure:

Step 1: Pre-training
- Initialize RBMs with visible and hidden units.
- Train RBMs layer by layer using the unlabelled training data.
- Stack RBMs to form the initial structure of the EDBN.

Step 2. Fine-tuning
- Initialize the EDBN using the weights and biases from pre-trained RBMs.
- Propagate labelled training data through the network to compute predictions.
- Determine the loss function by comparing the actual and expected labels.
- Backpropagate the error to calculate gradients of the loss concerning weights and biases.
- Weights and biases can be updated using an optimization process (like gradient descent).
- Repeat the process for a predefined number of epochs or until convergence.

Step 3. Update Weights
- Iterate through the network's weights and update them based on the calculated gradients.

Step 4: Repeat Fine-Tuning if Necessary
- Assess the convergence state based on predefined criteria (e.g., change in loss, number of epochs).
- Evaluate the accuracy of the validation dataset.
- Repeat the fine-tuning process if convergence criteria are not met or validation accuracy is below the target.
- Otherwise, stop the training.

The objective is to create a starting point for the algorithm's iterative search space exploration. Let N represent the number of fishes in the school, D denote the dimensionality of the search space, and P indicate the persistence-related parameters. The user-defined parameters include the maximum number of iterations (It_{max}), step size for individual movement $(step_{ind})$, step size for volitive movement $(step_{vol})$, and the persistence strength (α). The initialization of fish positions (X) is carried out randomly using Equation (26) within the search space, where $x_{i,d}$ represents the position of fish i in dimension d:

$$x_{i,d} \sim U(Search\_Space\_Min_d, Search\_Space\_Max_d) \quad (26)$$

Where $U(a,b)$ represents a uniform distribution between a and b. Additionally, weights (W) are assigned to each fish, initialized uniformly within a defined range expressed in Equation (27).

$$W_i \sim U(W_{scale}/2, W_{scale}) \quad (27)$$

Where W_{scale} is a user-defined parameter setting the maximum weight. The persistent memory structures, denoted as M, are initialized as empty arrays mathematically expressed as Equations (28) and (29).

$$M_{positions} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{n \times d} \quad (28)$$

$$M_{fitness} = \begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix}_{n} \quad (29)$$

These structures will store successful fish positions and associated fitness values across iterations. The initialization process ensures that the fish school and memory structures are prepared for the subsequent iterative optimization process, forming the foundation for PFSS to explore and exploit the search space adaptively based on historical information.

3.2.2. Loop

The main objective of this step is to explore the search space adaptively while leveraging persistent memory to guide the fish school toward promising regions. Let t represent the current iteration, $X_{i,t}$ denote the position of fish i at iteration t, and $F_{i,t}$ be the fitness of fish i at iteration t. Additionally, B_t represents the barycenter of the fish school at iteration t, I_t denotes the collective-instinctive movement vector and $\Delta f_{i,t}$ represents the fitness variation of fish i from the last to the current iteration. The main loop begins with calculating the fitness for each fish at the present iteration using Equation (30).

$$F_{i,t} = Fitness(X_{i,t}) \quad (30)$$

Equation (30) is followed by the application of the individual movement operator to every fish, with the parameters $step_{ind}$ determining the maximum displacement for this movement and $rand(-1,1)$ representing a uniformly distributed random value between -1 and 1, and it is expressed as Equation (31).

$$X_{i,t+1} = X_{i,t} + rand(-1,1) \times step_{ind} \quad (31)$$

Only when the fitness increases then the new position $X_{i,t+1}$ accepted, which is expressed as Equations (32) and (33).

$$F_{i,t+1} = Fitness\left(X_{i,t+1}\right) \quad (32)$$

$$\text{If } F_{i,t+1} > F_{i,t} \text{ then } X_{i,t+1} = X_{i,t+1} \text{ else } X_{i,t+1} = X_{i,t} \quad (33)$$

The feeding operator adjusts the fish's weights in response to changes in fitness after each swim. In this iteration t, let's say that fish i weighed $W_{i,t}$:

$$W_{i,t+1} = W_{i,t} + \frac{\Delta f_{i,t}}{max(|\Delta f_{i,t}|)} \quad (34)$$

The weighted average of the fish's displacements is computed via the collective-instinctive movement operator and is expressed as Equation (35).

$$I_t = \frac{\sum_{i=1}^{N} \Delta X_{i,t} \Delta F_{i,t}}{\sum_{i=1}^{N} \Delta F_{i,t}} \quad (35)$$

Algorithm 6: Loop

Input:

- N : Number of fish in the school
- D : Dimensionality of the search space
- It_{max}: Maximum number of iterations
- $step_{ind}$: Step size for individual movement
- $step_{vol}$: Step size for volitive movement
- α: : Persistence strength
- Other user-defined parameters

Output:

- Optimized fish positions X representing solutions in the search space.

Procedure:

Step 1: Initialization

- Initialize fish positions randomly in the search space.
- Initialize weights for each fish.
- Initialize persistent memory structures.

Step 2: Main Loop

- For each iteration t until It_{max} is reached:
- Calculate the fitness of each fish based on their positions.
- Apply the individual movement operator to update fish positions.
- Calculate fitness again for each fish.
- Update weights using the feeding operator based on fitness variations.
- Determine the collective-instinctive locus of motion.
- Revise the locations of the fish by using the collective-volitive movement controller.
- Update persistent memory with successful fish positions and associated fitness values.

When $\Delta X_{i,t} = X_{i,t+1} - X_{i,t}$ and $\Delta F_{i,t} = F_{i,t+1} - F_{i,t}$. The locations of the fish are adjusted using the collective-volitive movement operator using the barycenter B_t. The parameter that specifies the maximum displacement for this movement is $step_{vol}$, and the distance between fish i and the barycenter is $distance(X_{i,t}, B_t)$. Here, the value between 0 and 1 is equally distributed as rand(0,1). This algorithm outlines the main loop of Persistent Fish School Search (PFSS), where the fish school iteratively refines their positions, adjusts weights based on fitness variations, and utilizes collective movement operators. The persistent memory structures store information about successful solutions, guiding the search across iterations. The procedure continues until an infinite number of iterations have been exhausted. The final fish positions serve as the output, representing the optimized solutions obtained by PFSS.

3.2.3. Movement Operator

The movement operator focuses on updating the positions of each fish within the school. The primary objective is to explore the search space individually while considering the historical success of each fish. Let t denote the current iteration, $X_{i,t}$ represent the position of fish i at iteration t, and $F_{i,t}$ denote the fitness of fish i at iteration t.

Algorithm 7: Movement Operator

Input:

- N : Number of fish in the school
- D : Dimensionality of the search space
- X : Current positions of fish
- F : Current fitness values of fish
- $step_{ind}$: Step size for individual movement

Output:

- Updated fish positions X based on individual movements
- Updated fitness values F reflecting the success of the movements

Procedure:

Step 1: **For each fish i in the school:**

- Calculate the fitness of fish i at the current position.
- Introduce a random perturbation to the current position using rand(−1,1) scaled by $step_{ind}$.
- Update the position of fish i based on the perturbation.
- Recalculate the fitness of fish i at the new position.
- If the new fitness is greater than the current fitness, accept the new position; otherwise, retain the current position.

Step 2: **Output the updated fish positions X and fitness values F.**

The process begins with the calculation of fitness using Equation (36) for each fish in the current iteration.

$$F_{i,t} = Fitness(X_{i,t}) \quad (36)$$

Each fish's position is updated using the individual movement operator. The new position $X_{i,t+1}$ is determined by introducing a random perturbation, controlled by $rand(-1,1)$, and scaled by $step_{ind}$ representing the maximum displacement for this movement.

$$X_{i,t+1} = X_{i,t} + rand(-1,1) \times step_{ind} \quad (37)$$

The algorithm ensures that the new position uses Equations (38) and (39), and it is accepted only if it improves fitness. Equations (38) and (39) ensure that a fish only moves to a new position if that position yields better fitness.

$$F_{i,t+1} = Fitness(X_{i,t+1}) \quad (38)$$

$$\text{If } F_{i,t+1} > F_{i,t} \text{ then } X_{i,t+1} = X_{i,t+1} \text{ else } F_{i,t+1} = F_{i,t} \quad (39)$$

3.2.4. Collective-Instinctive Movement

The collective-instinctive movement behaviour of a school of fish determines how each fish moves in response to the group's overall conduct. The primary goal is to direct the fish to potentially fruitful areas of the search space by considering their average motions. Let's denote t as the current iteration, $X_{i,t}$as the position of fish i at iteration t, $F_{i,t}$ as the fitness of fish i at iteration t,I_t as the collective-instinctive movement vector, $\Delta X_{i,t} = X_{i,t+1} - X_{i,t}$ as the displacement vector of fish i between iterations t and $t+1$, and $\Delta F_{i,t} = F_{i,t+1} - F_{i,t}$ as the fitness variation of fish i from iteration t to $t+1$. The collective-instinctive movement operator involves three main steps:

The fitness improvements define the weights in Equation (40)$(\Delta F_{i,t})$ linked to the fishes' motions, and the equation calculates the weighted average of their displacements. Fish with more significant fitness improvements influence the collective movement more.

$$I_t = \frac{\sum_{i=1}^{N} \Delta X_{i,t} \Delta F_{i,t}}{\sum_{i=1}^{N} \Delta F_{i,t}} \quad (40)$$

The new position of each fish is updated based on the calculated collective-instinctive movement vector. This step ensures that each fish is encouraged to move towards regions in the search space where their fellow fishes have experienced fitness improvements.

$$X_{i,t+1} = X_{i,t} + I_t \quad (41)$$

After updating the positions, the fitness values of the fish are recalculated based on their new positions in the search space. This step is crucial as it evaluates the performance of the fish in their updated positions.

$$F_{i,t+1} = Fitness(X_{i,t+1}) \quad (42)$$

The individual movement operator is directly related to the collective-instinctive movement operator through the displacement vectors $\Delta X_{i,t}$ Each fish wanders about the search space in this operator according to a random perturbation.

$$\Delta X_{i,t} = \Delta X_{i,t+1} - X_{i,t} \quad (43)$$

Algorithm 8: Collective-Instinctive Movement Operator

Input:

- N : Number of fish in the school
- X: Current positions of fish
- ΔX : Displacement vectors of fish
- ΔF : Fitness variations associated with fish movements

Output:

- Updated fish positions X based on collective-instinctive movements

Procedure:

Step 1: Calculate the weighted average displacement vector I based on fitness improvements:

- For each fish i in the school:
- Calculate ΔX_i as the displacement vector between the current and next positions.
- Calculate ΔF_i as the fitness variation associated with the movement.
- Compute I as the weighted average of ΔX_i using fitness variations ΔF_i.

Step 2: Update the positions of each fish based on the collective-instinctive movement vector I:

- For each fish i in the school:
- Update the position X_i using the collective-instinctive movement vector I.

The fitness variations $(\Delta F_{i,t})$ associated with the movements in the individual movement operator are reused in the collective-instinctive movement operator. These fitness variations determine the weights for the collective movement, reinforcing the significance of successful individual movements in guiding collective behaviour.

$$\Delta F_{i,t} = F_{i,t+1} - F_{i,t} \quad (44)$$

3.2.5. Collective-Volitive Movement Operator

This operator regulates the exploration and exploitation balance within the fish school by adjusting their positions based on the school's barycenter. The purpose is to enhance the adaptive search capability of the algorithm, considering both the barycenter and the weight of each fish. Let's denote t as the current iteration, $X_{i,t}$ as the position of fish i at iteration t, $W_{i,t}$ as the weight of fish i at iteration t, B_t as the barycenter of the fish school at iteration t, $step_{vol}$ as the parameter defining the maximum displacement for the volitive movement and distance $(X_{i,t}, B_t)$ as the Euclidean distance between the position of fish i and the school barycenter. The Collective-Volitive Movement Operator consists of two components based on whether the total school weight has increased or not:

Attraction to Barycenter

$$X_{i,t+1} = X_{i,t} - step_{vol} \times rand(0,1) \times \frac{X_{i,t} - B_t}{\text{distance}\,(X_{i,t}, B_t)} \quad (45)$$

where $X_{i,t+1}$is the updated position of fish i after volitive movement, $rand(0,1)$ is a uniformly distributed random number between 0 and 1, $step_{vol}$ is the parameter defining the maximum displacement for this movement. In this context, distance $(X_{i,t}, B_t)$ refers to the geometric measure between the fish i current location and the school barycenter B_t.

Dispersion from Barycenter

$$X_{i,t+1} = X_{i,t} - step_{vol} \times rand(0,1) \times \frac{X_{i,t} - B_t}{\text{distance}\,(X_{i,t}, B_t)} \quad (46)$$

Algorithm 9: Collective-Instinctive Movement Operator

Input:

- N : Number of fish in the school
- X : Current positions of fish
- ΔX : Displacement vectors of fish
- ΔF : Fitness variations associated with fish movements

Output:

- Updated fish positions X based on collective-instinctive movements

Procedure:

Step 1: Calculate Weighted Displacements:

- For each fish i in the school:
- Calculate ΔX_i as the displacement vector between the current and next positions.
- Calculate ΔF_i as the fitness variation associated with the movement.

Step 2: Compute Weighted Average Displacement Vector I:

- Calculate I as the weighted average of ΔX_iusing fitness variations ΔF_i

Step 3: Update Fish Positions:

- For each fish i in the school:
- Update the position X_i using the collective-instinctive movement vector I.

In this context, $X_{i,t+1}$ is the current position of fish i after it has moved its body, $rand(0,1)$ is a randomly distributed integer between 0 and 1, $step_{vol}$ is the maximum

displacement that this movement can achieve, and the distance between the fish i position and the school barycenter B_t is given by the equation distance $(X_{i,t}, B_t)$. These equations capture the volitive movement of fish in response to the school barycenter.

Component 1 is used to entice fish towards the barycenter if the overall school weight $\sum_{i=1}^{N} W_{i,t}$ has grown from the previous to the current iteration. Otherwise, Component 2 is applied, dispersing fishes away from the barycenter.

3.2.6. Feeding Operator

This operator is responsible for updating the weights of each fish in the school based on their fitness variations, reflecting the success of their movements. The feeding operator plays a crucial role in determining the influence of individual fish on the collective behaviour of the school.

Let's denote t as the current iteration, $W_{i,t}$ as the weight of fish i at iteration t,$\Delta F_{i,t}$ as the fitness variation associated with the movement of fish i from iteration t to $t+1$, and W_{scale} as the user-defined attribute restricting the variation range of weights.

Based on the change in fitness $\Delta F_{i,t}$, Equation (47) modifies the weight of every fish. The term $max(|\Delta F_{i,t}|)$. This normalization ensures that weights are adjusted proportionally, considering the magnitude of fitness improvements.

$$W_{i,t+1} = W_{i,t} + \frac{\Delta F_{i,t}}{max(|\Delta F_{i,t}|)} \quad (47)$$

The fitness variations $\Delta F_{i,t}$ used in the feeding operator are directly linked to the success of individual movements in the earlier steps. These variations represent how well each fish has performed regarding fitness improvement.

$$\Delta F_{i,t} = F_{i,t+1} - F_{i,t} \quad (48)$$

Equation (49) ensures that the weights $W_{i,t+1}$are bounded within a specified range. The clip function restricts the values to stay within the range $\left[1, \frac{W_{scale}}{2}\right]$, preventing weights from becoming too small or too large.

$$W_{i,t+1} = clip\left(W_{i,t+1}, 1, \frac{W_{scale}}{2}\right) \quad (49)$$

The weights are initialized using Equation (50) at the beginning of the optimization process. The initial value is set to half of the user-defined attribute W_{scale}.

$$W_{i,0} = \frac{W_{scale}}{2} \quad (50)$$

Algorithm 10: Feeding Operator

Input:
- N : Number of fish in the school
- W_{scale}: User-defined attribute restricting the variation range of weights
- $F_{i,t}$: Fitness value of fish i at iteration t

Output:
- Updated weights $W_{i,t+1}$for each fish

Procedure:

Step 1: Calculate Fitness Variations
- For each fish i in the school:
- Calculate the fitness variation $\Delta F_{i,t} = F_{i,t+1} - F_{i,t}$ based on the fitness values.

Step 2: Update Weights
- For each fish i in the school:
- Update the weight $W_{i,t+1} = W_{i,t} + \frac{\Delta F_{i,t}}{max(|\Delta F_{i,t}|)}$.
- Normalize the weight to stay within the range $\left[1, \frac{W_{scale}}{2}\right]$ using the clip function.

Step 3: Initialization
- Initialize weights $W_{i,0} = \frac{W_{scale}}{2}$ at the beginning of the optimization process.

3.2.7. Persistent Memory Mechanisms

The Persistent Memory Mechanisms involve memory and partition operators, providing a form of persistence to guide the fish school towards promising regions in the search space. Let's denote N as the number of fish in the school, $X_{i,t}$ as the position of fish i at iteration t,M as the number of memory positions stored, and P as the number of partitions created within the fish school.

The Memory Operator involves updating and maintaining a memory of promising positions visited by the fish school. At each iteration, the memory is updated based on the fitness values of the current positions. The Memory Update function specified in Equation (51) considers the current positions $X_{i,t}$and their corresponding fitness values $F_{i,t}$.It updates the memory $M_{t+1}$to store the most promising positions visited by the fish school.

$$M_{t+1} = UpdateMemory(M_t, X_{1,t}, X_{2,t}, \ldots., X_{N,t}, F_{1,t}, F_{2,t}, \ldots. \quad (51)$$

The Partition function specified in Equation (52) considers the current positions $X_{i,t}$ and their corresponding fitness values $F_{i,t}$.It partitions the fish school into P_{t+1}subgroups based on fitness values, promoting diversity in the exploration strategy.

$$P_{t+1} = PartitionFishSchool(X_{1,t}, X_{2,t}, \ldots., X_{N,t}, F_{1,t}, F_{2,t}, \ldots., F_{N,t}) \quad (52)$$

Algorithm 11: Persistent Memory Mechanisms

Input:

- N : Number of fish in the school
- M_t: Memory positions at iteration$t$
- $X_{i,t}$: Positions of fish i at iteration t
- $F_{i,t}$: Fitness values of fish i at iteration t

Output:

- Updated memory positions M_{t+1}

Procedure:

Step 1: Memory Update

- For each fish i in the school:
- Check if the fitness value $F_{i,t}$ is better than the fitness value associated with the corresponding position in the memory.
- If better, update the memory position with the current position
- If not better, retain the existing memory position: $M_{i,t+1} = X_{1,t}$.

Step 2: Partitioning:

- Determine the number of partitions P_{t+1}based on the fitness values of the fish in the current iteration.
- Group fish into P_{t+1}partitions, considering their fitness values and positions.
- Each partition represents a subgroup of the fish school with similar fitness characteristics.

3.2.8. Stopping Condition

The Stopping Condition establish a well-defined criterion to determine whether the algorithm has achieved a satisfactory solution or if further iterations are necessary. One common approach is to monitor the number of iterations and compare it with a predefined maximum iteration count.

Iteration Count as a Stopping Criterion

The iteration count is a straightforward and widely used metric for determining when to stop the optimization process. The goal is to give the algorithm time to run through its iterations to thoroughly explore the search space and find the best possible answers. Let It_{max} represent the maximum allowed number of iterations. The stopping condition can be formulated as Equation (53)

$$Stopping\ Condition: t \geq It_{max} \tag{53}$$

Where t denotes the current iteration, the optimization continues until the current iteration surpasses or equals the predefined maximum iteration count.

Iteration Count Decay

PFSS often incorporates a linear decay mechanism for specific parameters to enhance adaptability. The iteration count can be utilized in the decay process to gradually reduce the impact of specific operators or adjust exploration-exploitation trade-offs as the optimization progresses.

$$Linear\ Decay: \alpha(t+1) = \alpha(t) - \frac{\alpha_{initial}}{It_{max}} \tag{54}$$

Where α represents the parameter being decayed, $\alpha_{initial}$ initialize the initial value of the parameter, and It_{max} is the maximum iteration count. This linear decay ensures a smooth transition, allowing the algorithm to adapt its behaviour throughout iterations dynamically.

This concise PFSS algorithm outlines the key steps and operators involved in the optimization process. It provides a framework for solving optimization problems by simulating the collective behaviour of a fish school while incorporating persistent memory mechanisms and adaptive parameter adjustments.

Algorithm 12: Persistent Fish School Search (PFSS) Algorithm

Input:

- N: Number of fish in the school
- It_{max}: Maximum number of iterations
- W_{scale}: User-defined weight scale
- $step_{ind}(initial)$: Initial step size for individual movement
- $step_{vol}(initial)$: Initial step size for collective-volitive movement
- $\alpha_{initial}$: Initial value for linear decay parameter
- Other problem-specific parameters

Output:

- Optimal solution or solutions

Procedure:

Step 1. Initialization

- Initialize fish positions randomly.
- Initialize weights for each fish within the range $\left[1, \frac{W_{scale}}{2}\right]$.
- Set iteration count $t = 0$.

Step 2. Main Loop

- While $t < It_{max}$, do:
- Determine each fish's fitness level.
- Perform Individual Movement Operator.
- Calculate fitness again.
- Feeding Program Manager.
- Launch the Operator for Collective-Instinctive Movement.
- Execute the Operator for Collective-Volitive Movement
- Run Feeding Operator.
- Run Persistent Memory Mechanisms (Memory Operator and Partition Operator).
- Check Stopping Condition.

Step 3: Stopping condition

- If $\geq It_{max}$, exit the loop.

4. Dataset

The AAU RainSnow Traffic Surveillance Dataset was created to overcome the limitations of traffic surveillance systems in inclement weather. A total of twenty-two five-minute movies shot at seven separate Aalborg and Viborg, Denmark, junctions make up the collection. These films showcase a range of lighting and environmental circumstances, from daylight to twilight and evening.

There are obstacles in the scenes, like car headlight glare, puddle reflections, and rain obscuring the images. A thermal infrared camera and a standard RGB colour camera record these sceneries. The extensive annotations are the distinguishing feature of this dataset. From each series, one hundred frames are chosen at random. Each frame is annotated at the instance level, per pixel, and includes road user category labels. A dataset with 2,200 annotated frames and 13,297 items is the outcome of this process. Thanks to these annotations, the dataset is now interoperable with tools and frameworks such as the COCO API, which adhere to MSCOCO category names. Table 1 presents a meticulous compilation of essential parameters and their descriptions, shedding light on a remarkable dataset.

Table 1. Essential parameters

Parameter	Description
Number of Videos	22
Video Duration	5 minutes each
Number of Intersections	7
Resolution (RGB Camera)	640x480 pixels
Resolution (Thermal Camera)	640x480 pixels
Frame Rate	20 frames per second
Annotated Frames	2,200 frames
Annotated Objects	13,297 objects
Weather Conditions	Rainfall, Snowfall, Adverse weather scenarios
Lighting Conditions	Daylight, Twilight, Night
Challenges	Headlight Glare, Reflections, Raindrop Blur
Annotation Format	JSON (Compatible with COCO API)

5. Performance Metrics

- Precision (PRCS): By comparing the proportion of properly predicted positive examples to the total number of anticipated positives, precision measures the accuracy of a classification model.
- Recall (RCLL): The sensitivity or recall of a model is the proportion of real positives to the total number of positive cases; it shows how well the model recognizes positive occurrences.
- Classification Accuracy (CL-ACC): To get a feel for the model's general accuracy, this key statistic determines the proportion of correct predictions relative to all forecasts.
- F-Measure (FMS): The F-MS, or F1 score, harmonizes precision and recall into a single value, offering a balanced evaluation of the model's effectiveness.
- Matthew Correlation Coefficient (MCC): MCC comprehensively evaluates binary and multiclass classification models by considering true and false positives and negatives.
- Fowlkes-Mallows Index (FMI): FMI assesses data similarity in clustering scenarios, calculating the geometric mean of precision and recall for a comprehensive evaluation.

6. Results and Discussion

6.1. PRCS and RCLL Analysis

Figure 1 presents a thorough analysis of Precision (PRCS) and Recall (RCLL) metrics, focusing on the performance of three distinct classification algorithms: DSOD, DenseYOLO, and PFSS-DBN. Table 2 provides a detailed view of Figure 1. DSOD achieves a PRCS of 53.924% and an RCLL of 59.428%. These metrics indicate that DSOD's approach emphasizes moderate precision in generating accurate positive predictions. It also demonstrates a commendable ability to identify relevant instances within the dataset.

DSOD's mechanism strives to balance minimizing false positive predictions and capturing many true positive instances.DenseYOLO, on the other hand, exhibits improved performance, with PRCS and RCLL values of 64.274% and 65.290%, respectively. DenseYOLO's working mechanism prioritizes higher precision in producing positive predictions and captures many relevant instances within the dataset. This reflects a more refined approach that prioritizes accuracy and identifying true positive instances.PFSS-DBN stands out as an outperforming algorithm, boasting remarkable PRCS and RCLL values of 87.401% and 88.464%, respectively.

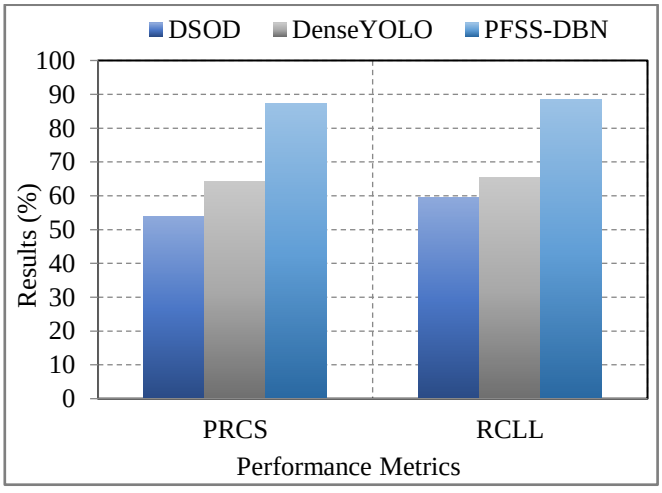

Fig. 1 PRCS and RCLL analysis

Table 2. PRCS and RCLL analysis results values

Classification Algorithms	PRCS(%)	RCLL(%)
DSOD	53.924	59.428
DenseYOLO	64.274	65.290
PFSS - DBN	87.401	88.464

The working mechanism of PFSS-DBN strongly emphasizes achieving superior precision in generating accurate positive predictions. It also demonstrates an exceptional ability to capture the most relevant instances within the dataset. This exceptional performance underscores a highly refined working mechanism that excels in accuracy and the identification of true positive instances.

When compared with Table 2, Figure 1 shows how each of the three classification algorithms performs differently when it comes to PRCS and RCLL. DSOD attains a modest level of accuracy and recall, DenseYOLO shows heightened memory and precision, and PFSS-DBN is the best with excellent recall and precision. Because algorithms' underlying processes strike a compromise between reducing the number of false positive predictions and increasing the number of real positive cases, these findings are critical for determining which algorithms are most suited to certain jobs.

6.2. CL-ACC and FMS Analysis

Two measures, the Fowlkes-Mallows Index (FMS) and Classification Accuracy (CL-ACC), are key to Figure 2, which depicts three different classification algorithms: DSOD, DenseYOLO, and PFSS-DBN. Figure 2 is significantly shown in Table 3. DSOD achieves a CL-ACC of 55.167% and an FMS of 56.542%. These metrics reflect DSOD's approach, which prioritizes moderate CL-ACC while maintaining a reasonable balance between precision and recall, as indicated by the FMS score.

DSOD's method aims to provide accuracy while ensuring a satisfactory trade-off between precision and recall.DenseYOLO surpasses DSOD with CL-ACC and FMS values of 64.997% and 64.778%, respectively. DenseYOLO's working mechanism emphasizes higher classification accuracy (CL-ACC) and a more refined balance between precision and recall, as depicted in the FMS score. This implies that DenseYOLO values accuracy while ensuring a commendable equilibrium between precision and recall.PFSS-DBN excels with exceptional CL-ACC and FMS values of 87.675% and 87.930%, respectively. The high FMS score demonstrates that PFSS-DBN's operational mechanism optimizes recall and accuracy, focusing on outstanding CL-ACC.

Table 3. CL-ACC and FMS analysis result values

Classification Alogirithms	CL-ACC(%)	FMS(%)
DSOD	55.167	56.542
DenseYOLO	64.997	64.778
PFSS - DBN	87.675	87.930

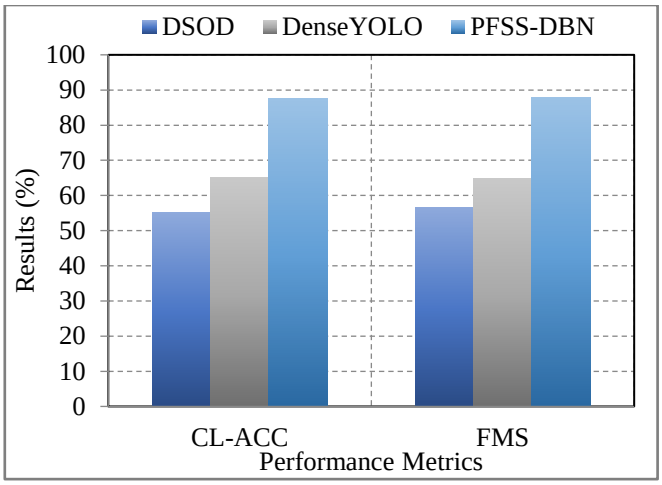

Fig. 2 CL-ACC and FMS analysis

The method used by PFSS-DBN guarantees a substantial equilibrium between recall and precision and good accuracy. The unique performance traits of these three classification algorithms concerning CL-ACC and FMS are illustrated in Figure 2 and Table 3. DSOD provides a balanced approach with modest precision, whereas DenseYOLO greatly enhances both. Regarding categorization accuracy and striking a balance between recall and precision, PFSS-DBN is unrivalled. These findings are critical for determining which algorithms are most suited to specific jobs that need high accuracy, recall, and precision.

6.3. FMI and MCC Analysis

Figure 3 displays the results of this study's meticulous examination of the FMI and the MCC for three separate categorization methods: DSOD, DenseYOLO, and PFSS-DBN. These metrics provide valuable insights into the working mechanisms of these algorithms and their unique approaches to capturing similarities and correlations within their classifications. With DSOD, we can reach 56.609% FMI and 10.522% MCC. These numbers show how DSOD works; it keeps the true and projected categories fairly close while establishing less connection between them. By striking a compromise between the two, DSOD can capture numerous real-life situations while reducing the number of false positive predictions. With an FMI of 64.780% and an MCC of 29.999%, DenseYOLO outperforms DSOD. This proves that DenseYOLO's algorithm gets a better correlation between anticipated and actual values and captures commonalities between actual and expected categories. The method used by DenseYOLOprioritises precision, emphasizing finding genuine positive cases. PFSS-DBN emerges as a high-performer with exceptional FMI and MCC values of 87.931% and 75.346%, respectively. This outstanding result is proof of how PFSS-DBN works. Its main goal is to achieve comparable real and projected classifications and build a strong link between the two. PFSS-DBN's approach ensures high accuracy and a robust correlation between precision and recall.

Table 4. FMI and MCC analysis result values

Classification Algorithms	FMI (%)	MCC(%)
DSOD	56.609	10.522
DenseYOLO	64.780	29.999
PFSS - DBN	87.931	75.346

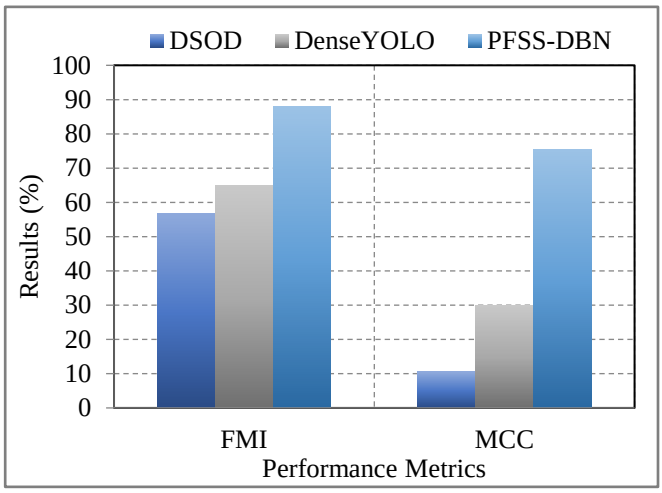

Fig. 3 FMI and MCC analysis

The three classification algorithms' unique performance characteristics concerning FMI and MCC are illustrated in Figure 3 and Table 4, which offer valuable insights. PFSS-DBN is excellent at collecting correlations and similarities, DenseYOLO is accurate, and DSOD is good at balancing recall and precision. These insights are priceless when evaluating algorithms for jobs that require a delicate balancing act between classification accuracy and the similarity of projected classifications to actual values.

7. Conclusion

This research marks a significant advancement in traffic surveillance, emphasizing the critical importance of robust object detection in the face of diverse weather conditions. Addressing the challenges inherent in such scenarios, the proposed PFSS-DBN algorithm, drawing inspiration from fish schools' persistent and adaptive nature, showcases its effectiveness in elevating detection accuracy. The adaptability of PFSS-DBN is a standout feature, particularly evident in its dynamic parameter optimization, ensuring reliable performance even in adverse weather, including rain and snow. Through extensive experimentation on the AAU RainSnow Traffic Surveillance Dataset, PFSS-DBN consistently outperforms conventional methods, affirming its potential as a resilient solution for traffic surveillance in regions experiencing varied weather climates. This study advances traffic surveillance methodologies and adds to the broader discourse on employing nature-inspired algorithms to address complex computer vision challenges. PFSS-DBN emerges as a promising tool for enhancing the efficacy of surveillance systems, offering dependable object detection crucial for public safety and efficient traffic management in urban environments facing unpredictable weather conditions. The research findings highlight the practical applicability of PFSS-DBN, providing a valuable contribution to the ongoing efforts to ensure the reliability of surveillance technologies in dynamic and challenging real-world scenarios.

References

[1] Enrico_Lagona et al., "Autonomous Trajectory Optimisation for Intelligent Satellite Systems and Space Traffic Management," *Acta Astronaut.*, vol. 194, pp. 185–201, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Yi-Chieh Sun, and Inseok Hwang, "Gaussian Mixture Probability Hypothesis Density Filter with Dynamic Probabilities: Application to Road Traffic Surveillance," *European Journal of Control*, vol. 69, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Xueqian Xu et al., "Exploiting High-Fidelity Kinematic Information from Port Surveillance Videos via A YOLO-Based Framework," *Ocean & Coastal Management*, vol. 222, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Shenghua_Zhou et al., "Integrating Computer Vision and Traffic Modeling for Near-Real-Time Signal Timing Optimization of Multiple Intersections," *Sustainable Cities and Society*, vol. 68, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Yaochen Li et al., "Vehicle Detection from Road Image Sequences for Intelligent Traffic Scheduling," *Computers and Electrical Engineering*, vol. 95, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Syed Khandker et al., "Cybersecurity Attacks on Software Logic and Error Handling Within ADS-B Implementations: Systematic Testing of Resilience and Countermeasures," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 4, pp. 2702–2719, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Moein Shakeri, and Hong Zhang, "COROLA: A Sequential Solution to Moving Object Detection using Low-Rank Approximation," *Computer Vision and Image Understanding*, vol. 146, pp. 27–39, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[8] H.B. Resmi, V.A. Deepambika, and M. Abdul Rahman, "Symmetric Mask Wavelet Based Detection and Tracking of Moving Objects Using Variance Method," *Procedia Computer Science*, vol. 58, pp. 58–65, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[9] Sarmad Rafique et al., "Optimized Real-Time Parking Management Framework using Deep Learning," *Expert Systems with Applications*, vol. 220, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Claudio V. Ribeiro, Aline Paes, and Daniel de Oliveira, "AIS-Based Maritime Anomaly Traffic Detection: A Review," *Expert Systems with Applications*, vol. 231, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Shakir Khan, and Lulwah AlSuwaidan, "Agricultural Monitoring System in Video Surveillance Object Detection Using Feature Extraction And Classification By Deep Learning Techniques," *Computers and Electrical Engineering*, vol. 102, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Zhaofeng Xu, Bin Wei, and Jian Zhang, "Reproduction of Spatial–Temporal Distribution of Traffic Loads on Freeway Bridges via Fusion of Camera Video and ETC Data," *Structures*, vol. 53, pp. 1476–1488, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Bharat Mahaur, and K.K. Mishra, "Small-Object Detection Based on YOLOv5 in Autonomous Driving Systems," *Pattern Recognition Letters*, vol. 168, pp. 115–122, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Feng Guo, Yi Wang, and Yu Qian, "Real-Time Dense Traffic Detection using Lightweight Backbone and Improved Pathaggregation Feature Pyramid Network," *Journal of Industrial Information Integration*, vol. 31, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Yuxing Yang, Zeyu Fu, and Syed Mohsen Naqvi, "Abnormal Event Detection for Video Surveillance Using an Enhanced Two-Stream Fusion Method," *Neurocomputing*, vol. 553, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Veronika Adamová, and Martin Boroš, "Effective Placement of Video Surveillance System Using 3D Scanning Technology for Traffic Safety," *Transportation Research Procedia*, vol. 55, pp. 1665–1672, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Bharat Mahaur, K.K. Mishra, and Anoj Kumar, "An Improved Lightweight Small Object Detection Framework Applied to Real-Time Autonomous Driving," *Expert Systems with Applications*, vol. 234, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Waseem Ullah et al., "TransCNN: Hybrid CNN and Transformer Mechanism for Surveillance Anomaly Detection," *Engineering Applications of Artificial Intelligence*, vol. 123, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] George Adaimi, Sven Kreiss, and Alexandre Alahi, "Traffic Perception from Aerial Images using Butterfly Fields," *Transportation Research Part C: Emerging Technologies*, vol. 153, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Edeh Michael Onyema et al., "Remote Monitoring System Using Slow-Fast Deep Convolution Neural Network Model for Identifying Anti-Social Activities in Surveillance Applications," *Measurement: Sensors*, vol. 27, pp. 1-11, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] J. Ramkumar et al., "Optimal Approach For Minimizing Delays In Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 3, pp. 1099–1111, 2024. [Google Scholar] [Publisher Link]

[22] J. Ramkumar, and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wireless Personal Communications*, vol. 120, no. 2, pp. 887–909, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] S.P. Geetha et al., "Energy Efficient Routing in Quantum Flying Ad Hoc Network (Q-FANET) Using Mamdani Fuzzy Inference Enhanced Dijkstra'S Algorithm (MFI-EDA)," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 9, pp. 3708–3724, 2024. [Google Scholar] [Publisher Link]

[24] M.P. Swapna, and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," *Second International Conference on Advancements in Smart Computing and Information Security*, Rajkot, India, pp. 131–140, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Nitish Kumar Ojha, Archana Pandita, and J. Ramkumar, "Cyber Security Challenges and Dark Side of AI: Review and Current Status," *Demystifying the Dark Side of AI in Business*, pp. 117–137, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ramkumar Jaganathan, and Vadivel Ramasamy, "Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 221–231, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[27] J. Ramkumar et al., "Gallant Ant Colony Optimized Machine Learning Framework (GACO-MLF) for Quality of Service Enhancement in Internet of Things-Based Public Cloud Networking," *Data Science and Communication*, pp. 425–438, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] 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," *International Journal of Computer Networks and Applications*, vol. 10, no. 4, pp. 668–687, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] J. Ramkumar, and R. Vadivel, "CSIP—Cuckoo Search Inspired Protocol for Routing in Cognitive Radio Ad Hoc Networks," *Proceedings of the International Conference on Computational Intelligence in Data Mining*, pp. 145–153, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[30] J. Ramkumar, and R. Vadivel, "Improved Frog Leap Inspired Protocol (IFLIP) – for Routing in Cognitive Radio Ad Hoc Networks (CRAHN)," *World Journal of Engineering*, vol. 15, no. 2, pp. 306–311, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[31] D. Jayaraj et al., "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *International Journal of Computer Networks and Applications*, vol. 10, no. 1, pp. 119–129, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[32] M. Lingaraj et al., "Query Aware Routing Protocol for Mobility Enabled Wireless Sensor Network," *International Journal of Computer Networks and Applications*, vol. 8, no. 3, pp. 258–267, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[33] R. Vadivel, and Ramkumar Jaganathan, "QoS-Enabled Improved Cuckoo Search-Inspired Protocol (ICSIP) for Iot-Based Healthcare Applications," *Incorporating the Internet of Things in Healthcare Applications and Wearable Devices*, pp. 109–121, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] J. Ramkumar, and R. Vadivel, "Improved Wolf Prey Inspired Protocol for Routing in Cognitive Radio Ad Hoc networks," *International Journal of Computer Networks and Applications*, vol. 7, no. 5, pp. 126–136, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] A. Senthilkumar et al., "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *International Journal of Computer Networks and Applications*, vol. 10, no. 2, pp. 217–230, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[36] Ramkumar Jaganathan, and Ramasamy Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *International Journal of Computing and Digital Systems*, vol. 10, no. 1, pp. 1063–1074, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[37] P. Menakadevi, and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," *International Conference on Advanced Computing Technologies and Applications*, Coimbatore, India, pp. 1–5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[38] J. Ramkumar et al., "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," *International Conference on Advanced Computing Technologies and Applications*, Coimbatore, India, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[39] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *International Journal of Computer Networks and Applications*, vol. 8, no. 6, pp. 795–803, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[40] Lingaraj Mani, Senthilkumar Arumugam, and Ramkumar Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*, Jaipur India, pp. 1–5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[41] J. Ramkumar et al., "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," *Soft Computing Applications in Modern Power and Energy Systems*, pp. 17–27, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[42] J. Ramkumar, and R. Vadivel, "Whale Optimization Routing Protocol for Minimizing Energy Consumption in Cognitive Radio Wireless Sensor Network," *International Journal of Computer Networks and Applications*, vol. 8, no. 4, pp. 455–464, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[43] Shaharyar Alam Ansari, and Aasim Zafar, "A Fusion of Dolphin Swarm Optimization and Improved Sine Cosine Algorithm for Automatic Detection and Classification of Objects from Surveillance Videos," *Measurement*, vol. 192, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[44] Malik Javed Akhtar et al., "A Robust Framework for Object Detection in a Traffic Surveillance System," *Electronics*, vol. 11, no. 21, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]