Research Article

Rabie Rehan*, Shahnorbanun Sahran, Zaid Abdi Alkareem Alyasseri, Nor Samsiah Sani, and Mohammed Azmi Al-Betar

Hyperparameters optimization of evolving spiking neural network using artificial bee colony for unsupervised anomaly detection

https://doi.org/10.1515/jisys-2024-0235
received April 25, 2024; accepted October 09, 2024

Abstract: Nowadays, anomaly detection in streaming data has gained considerable attention due to the exponential growth in the data gathered by Internet of Things applications. Analyzing and processing vast data volumes requires a system capable of working in real-time. Moreover, obtaining labeled data for supervised learning is challenging, as it requires human involvement, is time-consuming, and costly. A promising direction is implementing evolving spiking neural networks (eSNN), which can be updated whenever new data becomes available without re-training previous samples. However, eSNN encounters significant challenges when it comes to manually tuning its hyperparameter values. As such, this work covers the current research gap by suggesting a novel method to optimize the hyperparameters of eSNN called online evolving spiking neural networks-artificial bee colony (OeSNN-ABC). Multiple scenarios have been utilized to evaluate the proposed method using two benchmark datasets: the Numenta anomaly benchmark (NAB) and the Yahoo Webscope using different criteria. Further validation was provided by comparing the proposed OeSNN-ABC against five well-known optimization algorithms: particle swarm optimization, grey wolf optimization, flower pollination algorithm, whale optimization algorithm, and grid search, alongside other classifiers such as random forest, support vector machine, and k-nearest neighbor. The findings revealed that OeSNN-ABC had the best performance among all compared optimization algorithms and classifiers, outperformed prior anomaly detection techniques for the NAB dataset, and achieved competitive results for the Yahoo Webscope dataset.

Keywords: anomaly detection, evolving spiking neural networks, hyperparameters optimization, artificial bee colony, deep learning, optimization

*** Corresponding author: Rabie Rehan**, Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, UKM, Bangi, 43600, Selangor, Malaysia; Department of Biomedical Engineering, College of Engineering, University of Thi-Qar, Nasiriyah, 64001, Iraq, e-mail: 103739@siswa.ukm.edu.my, rabie.rehan@utq.edu.iq
Shahnorbanun Sahran: Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, UKM, Bangi, 43600, Selangor, Malaysia, e-mail: shahnorbanun@ukm.edu.my
Zaid Abdi Alkareem Alyasseri: Information Technology Research and Development Center (ITRDC), University of Kufa, 54001, Najaf, Iraq; College of Engineering, University of Warith Al-Anbiyaa, 56001, Karbala, Iraq, e-mail: zaid.alyasseri@uokufa.edu.iq
Nor Samsiah Sani: Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, UKM, Bangi, 43600, Selangor, Malaysia, e-mail: norsamsiahsani@ukm.edu.my
Mohammed Azmi Al-Betar: Artificial Intelligence Research Center (AIRC), Ajman University, 0001 Ajman, United Arab Emirates; Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, 19328, Amman, Jordan, e-mail: m.albetar@ajman.ac.ae

1 Introduction

The importance of detecting anomalies in streaming data has risen significantly due to the rapid growth of data gathered by Internet of Things applications [1–5]. Because the streaming data may change, evolve, and arrive in real time [6], the anomaly detectors should adjust their parameters in real-time based on the current input data to better detect anomalies in new input data. Hence, traditional static models may not be accurate when detecting anomalies in streaming data [7]. Moreover, the task of gathering labeled data exhibits significant challenges, often impractical due to its requiring human involvement, time consumption, and high cost. Therefore, real-time algorithms are essential for processing large amounts of streaming data in an unsupervised way [8]. One of the potential directions is toward more biologically plausible neuron models, especially spiking neural networks (SNNs) [9].

The SNNs are a highly promising avenue that emphasizes the development of neuron models with greater biological realism. Neuron models communicate by transmitting and receiving action potentials ("spike trains") [10]. The SNN's neuron generates spikes only when the membrane potential reaches the desired threshold, which is a highly energy-efficient process, making SNN suitable to use in real-time systems specifically for online anomaly detection [11]. Although there are multiple enhanced versions of the SNN, an evolving spiking neural network (eSNN) is one of the well-known SNN types [12]. The eSNN offers a range of advantages, including being a fast one-pass learning method and an essential and effective neural model for online learning as it is evolving and modified whenever new data is available without re-training previous samples [9]; as such, it is appropriate for online anomaly detection in an unsupervised mode [2].

Nevertheless, the eSNN still suffers from manually setting the values of the hyperparameters before running through a trial-and-error approach [13]. Furthermore, due to dependencies in time series, two critical hyperparameters, window size (*Wsize*) and anomaly classification factor (ε), have a considerable influence on the effectiveness of anomaly detection on streaming data [2,8,14]. Therefore, choosing inappropriate values for these hyperparameters can compromise the efficiency of the anomaly detection system. Regarding *Wsize*, the selection of the proper value presents a particular challenge, as there is no general fixed (*Wsize*) that can be applied across diverse time series [14]. For example, a large (*Wsize*) value would likely lose some contextual drift, potentially increasing false positives by misinterpreting normal values as anomalies. Conversely, a small (*Wsize*(value might lead to an increased number of false positives, in which normal data are incorrectly identified as anomalies, due to the heightened sensitivity to short-term changes [15]. Also, finding the best value of (ε) is crucial, as it determines whether the input values are normal or anomalous. For instance, if the (ε) value is set too high, it can cause more false negatives, resulting in low recall. Whereas, setting it too low may increase the false positives, thereby reducing precision and detracting overall accuracy [2]. Hence, the values of (*Wsize*) and (ε) are recommended to be optimized to automate the search for optimal hyperparameter values, leading to enhanced anomaly detection accuracy and improved effectiveness and performance compared to manual tuning.

Various swarm nature-based optimization algorithms have been integrated with eSNN models in earlier works to overcome these challenges and fill the significant gap in finding the values of the optimal hyperparameters, which have shown encouraging results [16–18]. In this work, we utilized the artificial bee colony (ABC) algorithm since it is a well-established algorithm and has the following advantages over state-of-the-art metaheuristic algorithms: It has fewer control parameters and achieves faster convergence toward an optimal target point, resulting in lower computing complexity [19,20]. Furthermore, the ABC algorithm has superior exploitation ability in specific situations compared to other algorithms. It is also derivative-free, which does not require derived information in the initial search. In addition, it is parameter-less, easy-to-use, simple-in-concepts, scalable, adaptable, and sound-and-complete. Despite the integration of ABC with other algorithms has been investigated [21–23], to the best of the authors' knowledge, this is the first attempt to integrate the ABC algorithm with the eSNN classifier to produce an efficient anomaly detection system.

The overall objective of this research is to introduce a novel method for unsupervised anomaly detection in streaming data, named online evolving spiking neural networks-artificial bee colony (OeSNN-ABC). This method cooperatively combines the strength of the online evolving spiking neural network (OeSNN) and the ABC to find the optimal values of hyperparameters: window size (*Wsize*) and anomaly classification factor (ε).

Although other solutions provided in the literature have examined the efficiency of eSNN in supervised or semi-supervised anomaly detection models, this work favors OeSNN mainly for unsupervised anomaly detection, which is particularly valuable for real-time streaming data where labeled data is scarce. That's why we believe this combined approach offers increased speedup and computational efficiency compared to other previous approaches. In addition, OeSNN-ABC is well-suited for deployment in environments with limited memory resources.

To achieve this objective, several contributions are established, which can be summarized as follows:

(1) Offering first-time integration between ABC and OeSNN to optimize two crucial OeSNN hyperparameters: window size (*Wsize*) and anomaly classification factor (ε) to adapt in unsupervised online anomaly detection.
(2) Test the proposed approach (OeSNN-ABC) based on two popular benchmark datasets [i.e., Numenta anomaly benchmark (NAB) and Yahoo Webscope] and conduct a comprehensive comparative analysis with the 17 state-of-the-art anomaly detection methods that have used the same datasets. The OeSNN-ABC gave a higher performance than the methods compared in most cases.
(3) Adopting suitable evaluation metrics such as Precision, Recall, *F*1-score, balanced accuracy (BA), and Matthews correlation coefficient to measure the efficiency of the proposed method.
(4) Compare the performance of the proposed method (OeSNN-ABC) against five well-known optimization algorithms: particle swarm optimization (PSO), grey wolf optimization (GWO), flower pollination algorithm (FPA), whale optimization algorithm (WOA), and grid search (GS).

The remainder parts of this article discuss the related works in Section 2. Section 3 presents a background of the concept of principles. Section 4 describes the proposed methodology, and Section 5 discusses the experiments provided and the results. Finally, Section 6 describes the conclusion and possible future research directions.

2 Related work

The great necessity for efficient and adaptable algorithms for anomaly detection has led to the study of various methodologies in the state-of-the-art literature. Different types of algorithms and techniques have been applied, depending on the application domain. Many anomaly detection methods exist for processing data in an offline manner, which means it's inappropriate for real-time streaming applications [8]. The Numenta and Numenta^{TM} presented by Ahmad et al. are two-bit different algorithms built for real-time applications. Both of them are composed of three main modules. The first module involves a hierarchical temporal memory (HTM) network, which predicts the current value of an input data stream. The second module is responsible for error calculation, while the third module determines input values as normal or anomalous based on this error likelihood [8]. Another benchmark study by Munir et al. [14] proposed a new deep learning approach called DeepAnT for semi-supervised anomaly detection in time series data. The DeepAnT model consists of two sub-models, an encoder and a decoder, where the input data were reconstructed by training the sub-models. The difference between the original and reconstructed data will identify anomalies during testing. The authors demonstrate the effectiveness of DeepAnT on several real-world datasets, achieving state-of-the-art performance compared to other unsupervised anomaly detection methods.

Although various anomaly detection methods have been proposed in the literature, the distinctive temporal and sequence characteristics of streaming data require specialized approaches for anomaly detection [24]. Thus, SNNs are considered to be more suitable for analyzing streaming data than traditional ANNs due to their ability to handle temporal patterns and increase their effectiveness in detecting anomalies [25]. In this context, Demertzis et al. introduced the anomaly system "Gryphon." Gryphon is constructed on a one-class evolving spiking neural network (eSNN-OCC). The proposed system uses a semi-supervised approach with a small set of labeled anomalies initializing the network, and the network evolves over time to learn the normal patterns of the data. All eSNN parameters are optimized using the versatile quantum-inspired evolutionary algorithm. The authors show that "Gryphon" outperforms several other anomaly detection approaches on real-world datasets [26]. The online evolving spiking neural network for unsupervised anomaly detection (OeSNN-UAD)

method was developed by Maciąg et al. for online anomaly detection. The OeSNN-UAD has improved the OeSNN classifier proposed by Lobo et al. [27]. The proposed method outperforms the competitive detectors for most of the time series in each category of the NAB dataset and, specifically, real data in the Yahoo Webscope dataset [2]. The OeSNN-UAD algorithm has emerged as an important benchmark algorithm for future research studies. A modified version of the OeSNN was used by Bäßler et al. to detect anomalies in multivariate time series. Using multi-dimensional Gaussian receptive fields, the authors presented an improved, practically achievable, and reliable learning method for multivariate time series. They also presented a replacement rank-order-based autoencoder that improved classification interpretability by adjusting network parameters depending on the exact timings of input spikes and a consistent anomaly score algorithm [28].

Even though eSNN exhibits promising results when applied to the anomaly detection domain, it still has the drawback of requiring manual tuning through trial and error [13]. Many attempts have been made by researchers to address these issues by optimizing the hyperparameters of eSNN through integration with metaheuristic algorithms. Recently metaheuristic optimization techniques are being used to develop hybrid machine learning solutions, enhancing algorithm performance and addressing diverse challenges in various applications [29,30]. The research study by Roslan et al. presents an integration between eSNN and Firefly (FA) optimization algorithm to search for the best values of the hyperparameters: mod, sim, and C factor (threshold) to achieve high performance [17]. Their findings show superior accuracy compared to using a standard eSNN and also reveal that there is no particular combination of parameter values. This indicates that the best accuracy results may be obtained when the optimal parameter has been found. The hybridization between eSNN and the differential evaluation (DE) algorithm is proposed by Saleh et al. [18]. The main reason behind this work is to overcome the issue related to eSNN hyperparameters tuning and get optimal values that improve classification accuracy by enhancing eSNN learning through DE algorithm. The outcomes demonstrate that the proposed method outperforms the standard eSNN in most cases when it comes to classification accuracy.

Nur et al. [31] claimed that combining dynamic population particle swarm optimization (DPPSO) and eSNN can improve the accuracy of time-series prediction and classification. The suggested approach is evaluated using several benchmark datasets and compared with other optimization algorithms. The results show that the DPPSO-ESNN approach outperforms other methods and achieves higher prediction accuracy. Another research conducted by Yusuf et al. [32] developed an integration between eSNN and harmony search algorithm (HSA), which influences the eSNN model's accuracy when the HSA optimizes the three crucial eSNN parameters: Mod, Sim, and C factors. It keeps looking for the best parameters by making improvements to the HSA competitors until the maximum number of iterations is reached. The outcome indicates improved classification accuracy and shows that the combination provided by eSNN-HSA was superior to that of standard eSNN.

Further, Ibad et al. [16] introduced eSNN-SSA, an optimization approach for the eSNN hyperparameters, and utilized it for time series classification. In the suggested method, the hyperparameters were optimized by integrating the salp swarm algorithm optimization algorithm with eSNN. The proposed approach is evaluated on several real-world datasets and shows significant improvements in classification accuracy compared to the benchmarking classifiers and the standard eSNN.

The aforementioned studies have focused on exploring the optimal eSNN frameworks for simple classification tasks. Also, most of the studies have concentrated on supervised or semi-supervised learning. Therefore, the main objective of this study is to optimize OeSNN hyperparameters by integrating with the ABC algorithm and applying it for unsupervised anomaly detection, aiming to achieve the highest possible accuracy.

3 Background

This section provides the reader with the background knowledge and basic concepts required to understand the proposed method. Section 3.1 describes the anomaly detection notion and its types. Section 3.2 explains the main concepts of the eSNN. Section 3.3 presents an overview of the essential steps of the ABC algorithm.

3.1 Anomaly detection

Anomaly or outlier detection identifies unexpected patterns, observations, or events in datasets that differ from the norm [33]. Anomaly detection stands out as a frequently encountered challenge within the realm of machine learning, exhibiting a diverse array of applications across various domains. Such applications encompass fraud detection, defect detection, and intrusion detection [34]. Anomalies can arise from unforeseen processes producing the data. For instance, in chemistry, an anomaly could be triggered by the improper execution of an experiment. Similarly, within medicine, a particular disease might give rise to unusual symptoms, constituting an anomaly. Moreover, in predictive maintenance, an anomaly can signal the early onset of system malfunction [35]. Popular techniques for anomaly detection include clustering-based, probabilistic-based, density-based, distribution-based, and distance-based approaches [36].

The anomalies generally fall into three categories significantly influencing the anomaly detection algorithm: point, contextual, and collective anomalies [1]. Point anomalies could be considered anomalous, among other instances in the dataset, such as a building temperature value having a typical shape and a noticeable rise/fall in the value, which may be considered an abnormal value. Contextual anomalies are observations considered normal but abnormal in other instances. For example, while heavy traffic on the highway is usual during working hours, it is improper traffic behavior after midnight. A collective anomaly is a set of related points that are normal individually but anomalous if taken together; an example of a collective anomaly is cardiac behavior, which cannot be detected by a single time interval observation but may be recognized by collective signals. Graphical representations of these anomaly categories are shown in Figure 1.

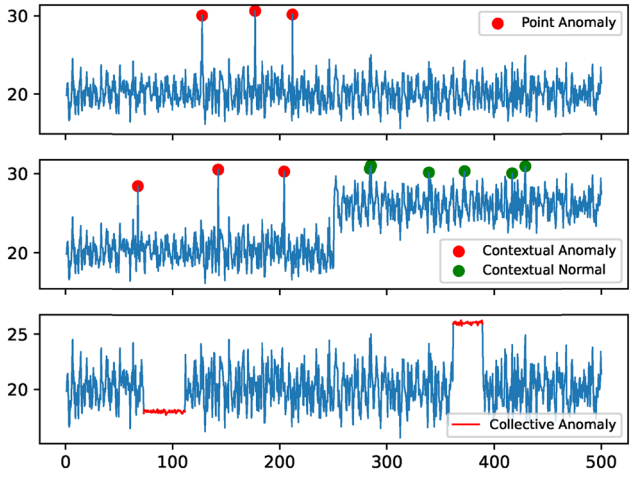


Figure 1: Anomaly categories [1].

3.2 eSNNs

eSNN is one of the most promising SNN architectures. It was first presented by Wysoski et al. [37] and used for visual pattern recognition. eSNN increases the adaptability of SNN by combining the produced neurons incrementally to identify the pattern in the given problem [38]. In this kind of network, the spiking neurons are generated (evolved) and incrementally combined to collect clusters and patterns from new data [12]. The eSNN iteratively created repositories for each class to make new data available without re-training the learned dataset [39]. The numerical data is converted into an encoding or spike trail for each input sample. All input samples are distributed among neurons using overlapping Gaussian receptive field neurons [40]. Each input feature is represented in a spatio-temporal spike pattern after encoding. Each GRF pre-synaptic neuron center μ_j and width are calculated [27]

$$\mu_j = I^n_{\min} + \frac{2j-3}{2}\left(\frac{I^n_{\max} - I^n_{\min}}{N-2}\right), \tag{1}$$

$$\sigma_j = \frac{1}{\beta}\left(\frac{I^n_{\max} - I^n_{\min}}{N-2}\right). \tag{2}$$

Here, $I^n_{\max}$ and $I^n_{\min}$ represent the interval values of the nth feature in a given window size, N is the number of receptive fields (GRF) for each feature, and parameter β (also known as overlap factor) controls the degree to which Gaussian random fields overlap. The output of neuron j is represented as

$$\text{out}_j = \exp\left(\frac{(x-\mu_j)^2}{2\sigma_j^2}\right), \tag{3}$$

where x is the input sample, and the time of firing for every pre-synaptic neuron j is defined as

$$T_j = T \cdot (1 - \text{out}_j), \tag{4}$$

where T is the synchronization time or spike interval.

To create initial neurons, the LIF model is used. The neuron will only fire whenever the post-synaptic potential (PSP) value exceeds its threshold value and fires at most once.

PSP of the ith neuron is identified as

$$\text{PSP}_i = \begin{cases} 0 & \text{if fired} \\ \sum_j w_{ji} \cdot \text{mod}^{\text{order}(j)} & \text{otherwise,} \end{cases} \tag{5}$$

where w_{ji} denotes the weight between pre-synaptic neuron j and output neuron i, order(j) represents the rank of the pre-synaptic neuron spike, and mod represents the modulation factor with value in [0, 1]. A new output neuron is created and linked to every pre-synaptic neuron in the previous layer, and the weights are determined using rank order for each sample that falls within the same class as follows:

$$w_{ji} = \text{mod}^{\text{order}(j)}. \tag{6}$$

The threshold γ_i of a newly generated output neuron is defined as

$$\gamma_i = \text{PSP}_{\max,i} \cdot C, \tag{7}$$

where C is a user-fixed value in [0, 1].

The weight vector for a recently produced output neuron is compared with existing output neurons in the repository. When the Euclidean distance between the weight vector of the already created output neuron and either of the prior output neurons is less than or equal to the SIM, then they are merged, and the following formulas are employed to calculate their thresholds and weight vectors:

$$w_{ji} = \frac{w_{\text{new}} + (w_{ji} \cdot M)}{M+1}, \tag{8}$$

$$\gamma_{ji} = \frac{\gamma_{\text{new}} + (\gamma_{ji} \cdot M)}{M+1}. \tag{9}$$

The variable M defines the count of earlier mergers involving similar neurons in the learning history of the eSNN. Once a merger occurs, the vector of weight for the newly formed output neuron is no longer considered, and the model is presented with the updated pattern. If no one of the existing neurons in the repository matches the newly generated output neuron (based on the SIM parameter), it is added to the repository. The entire concept of eSNN may be found in [41].

Recently, the OeSNN-UAD approach was introduced by Maciąg et al. [2] as a detector for streaming data, which adapts the OeSNN proposed by Lobo et al. [27]. In contrast to OeSNN, OeSNN-UAD did not split the output neurons into distinct classes; instead, it consisted of three additional modules: value correction, anomaly classification, and generation of output values of candidate output neurons, as shown in Figure 2.

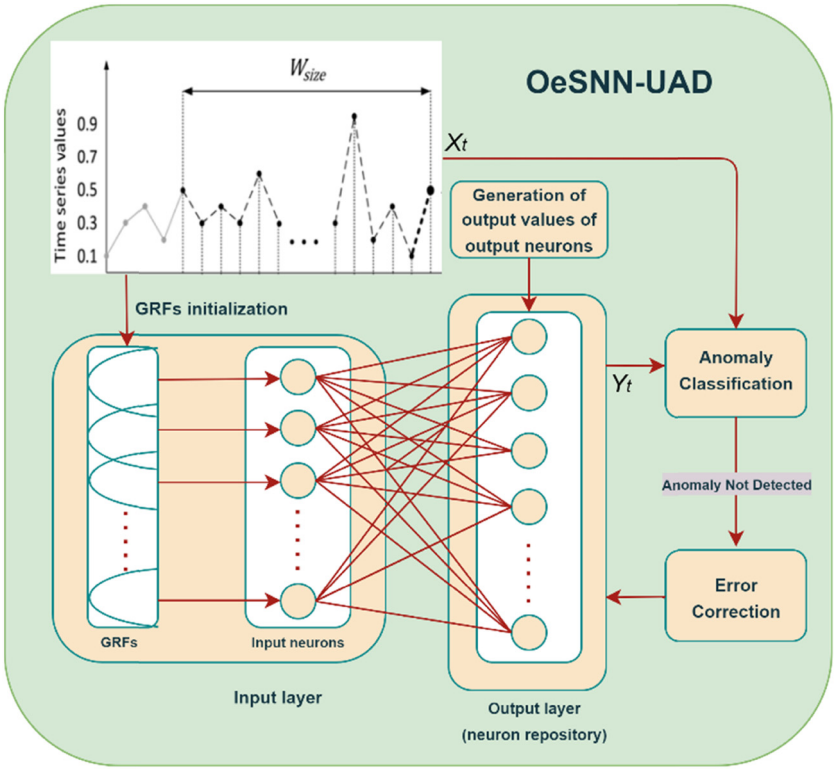


Figure 2: OeSNN-UAD architecture [2].

Under two separate conditions, the OeSNN-UAD classifies the input value as anomalous. First, if no output neuron in the repository is firing. Second, if the difference between the actual input value and its OeSNN-UAD prediction exceeds the average of prediction error plus a user-given multiplicity of the standard deviation of the most recent prediction errors, more details are in [2]. In this work, the OeSNN-UAD was employed as a baseline architecture.

3.3 Overview of ABC algorithm

Metaheuristic algorithms are stochastic optimization approaches that search for the most optimal solution to an optimization problem from among all potential solutions [42]. Some of these algorithms focus on the alteration and enhancement of a singular candidate solution, exemplified by simulated annealing, tabu search, and iterated local search. Other algorithms, such as PSO, CS, ACO, and FPA, are population-based, which means that several candidate solutions are maintained and improved during the search until an optimal solution is obtained when the solutions of the whole population are similar [43,44]. Most population-based metaheuristic algorithms were influenced by the collective intelligence of swarms of animals and insects, including fish, birds, bacteria, ants, and fireflies. Honeybees are one of the social creatures that exhibit interesting behaviors and traits. When they act as swarms, honeybees are fascinating creatures that display a variety of unexpectedly clever behaviors [45].

The ABC is one of the bee-based algorithms developed by Karaboga [46]. ABC involves three kinds of bees: employed, onlooker, and scout bees. The employed bees move randomly to explore positions of food source, and when they reach the food source, they generate a new solution and use a greedy selection technique to compare it with the previous one. Then, by waggle dancing, they interact with the onlooker bees about the path of food source, profitability, and nectar amount. The onlooker bees select the new food source based on a probability proportional to the value of the food source. When the food source does not improve after specified trials, a different food source chosen randomly will be abandoned and replaced. The employee bee assigned to that food position leaves it and goes to work as a scout bee [47]. The ABC procedure requires a cycle of four phases: initialization, employed, onlooker, and scout phases:

3.3.1 Initialization phase

The first phase of the ABC algorithm is initializing the population represented by P of SN solution (food source). Each solution x_m^n is a D-dimensional vector, where D is the number of parameters needed to optimize their values. Each food source is randomly generated relative to the upper and lower boundaries of the constraint variables as follows:

$$x_m^n = x_{\min}^n + \text{rand}(0,1)(x_{\max}^n - x_{\min}^n), \tag{10}$$

where $x_{\min}^n$ and $x_{\max}^n$ are the upper and lower bound of the variable for the search space, respectively.

3.3.2 Employed bee phase

Employed bees are dispatched out searching for food sources during this phase, updating their recollection of feasible food sources. To create a feasible candidate, the memory is updated using (11)

$$v_m^n = x_m^n + \varnothing_m^n (x_m^n - x_k^n), \tag{11}$$

where $n \in \{1, 2, \ldots, d\}$, $k \in \{1, 2, \ldots, \text{SN}\}$, where $k \neq n$ and v_m^n represents the new solution created by the employed bee and $\varnothing_m^n$ is a uniformly distributed random number between [−1, 1] for controlling the exploration of nearby food sources near x_m^n, and compare the positions of the two food sources. In case v_m^n fails to correspond to boundary constraints, then they are handled using (12)

$$v_m^n = \begin{cases} \max_m \text{ if } v_m^n > \max_m \\ \min_m \text{ if } v_m^n < \min_m. \end{cases} \tag{12}$$

3.3.3 Onlooker bee phase

A roulette wheel selection method is employed to ascertain which food source will be investigated by the onlooker bees. The onlooker bee selects the best location of the food source depending on the probability proportional to the worth of the food source by using (14). After evaluating the quality (fitness) of the food source positions in the memory by using (13), the onlooker bees apply (11) to update the feasible candidate

$$\text{fit}_m = \begin{cases} \dfrac{1}{(1 + F_m)} & \text{if } F_m \geq 0 \\ 1 + abs(F_m) & \text{if } F_m < 0, \end{cases} \tag{13}$$

$$p_m = \frac{\text{fit}_m}{\sum_{i=1}^{\text{SN}} \text{fit}_i}, \tag{14}$$

where F_m denotes the objective function, fit_m represents the fitness value of the solution m proportionate to the nectar amount of the food source, p_m the probability of the food source, and NS is the number of employed/onlooker bees.

3.3.4 Scout bees phase

The food source is disregarded if it cannot be enhanced through a predetermined limited number of attempts (*limit*). The employed bee corresponding to this food source becomes a scout bee to search randomly for a new food source using equation (10).

3.3.5 Stopping condition

The stopping criteria in ABC are determined by the number of iterations, called maximum cycle number (MCN), so, when the stopping condition is met, the iterations stop, and the results are reported; otherwise, back to the employed bee phase.

4 The proposed method

This section explains the proposed method of integrating the OeSNN with the ABC algorithm to discover suitable hyperparameter values of OeSNN and adopt it in anomaly detection on streaming data. The integration was performed by using the well-known wrapper approach. The wrapper approach combines the classifier and an optimization algorithm. The optimizer provides the information to the classifier, which uses this information provided by the optimizer solution candidates and operates as a quality measure for a given solution [48]. In this work, the classifier OeSNN is integrated with the ABC algorithm to optimize and find the best combination of the hyperparameters, named window size (*Wsize*) and anomaly classification factor (ε). These two parameters are present in every candidate solution and together they form the OeSNN structure. Algorithm 1, Figures 3 and 4 depict pseudocode, architecture, and the flowchart of the proposed method, respectively. For more details, the main procedure of OeSNN-ABC is discussed subsequently.

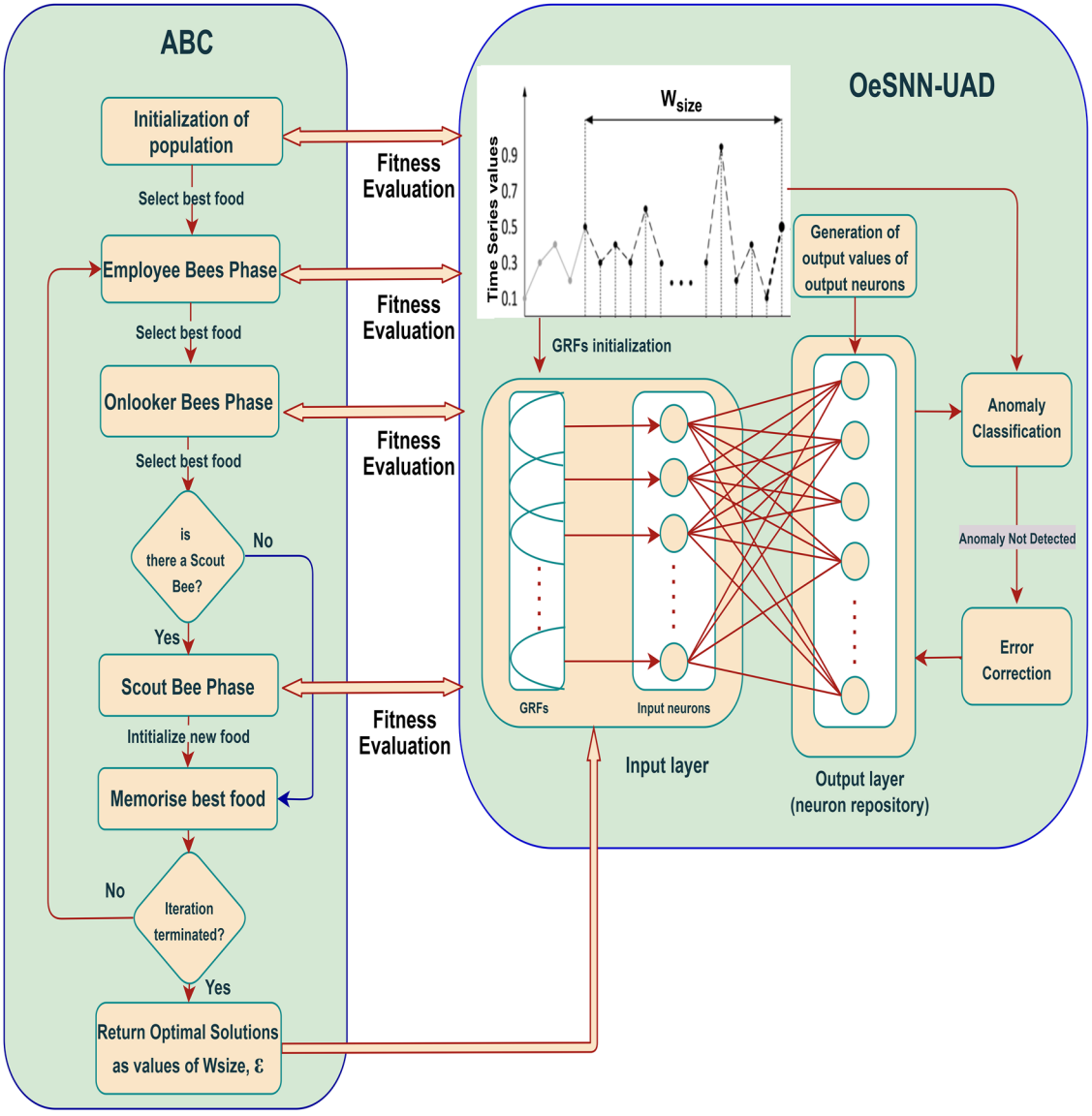


Figure 3: The proposed OeSNN-ABC architecture. Source: Created by the authors.

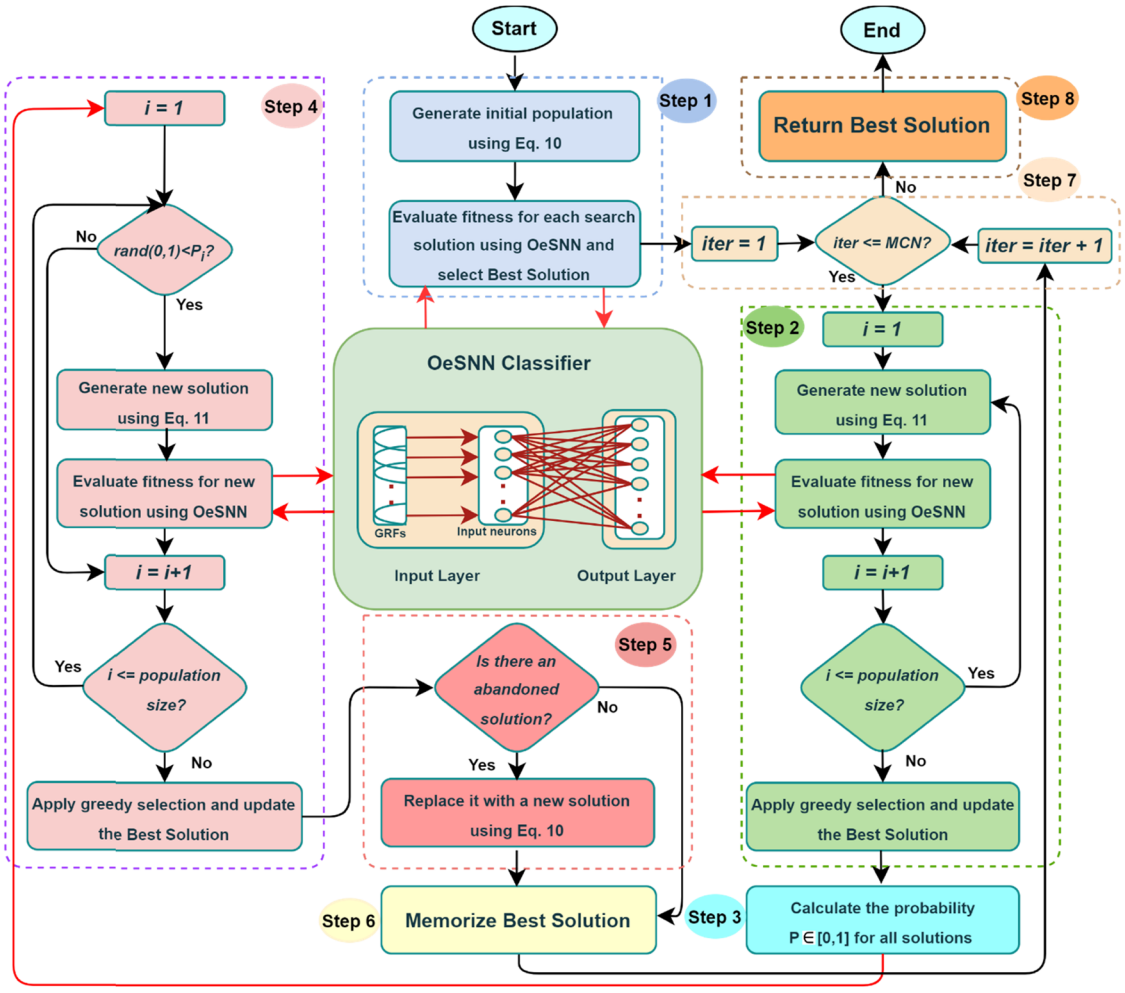


Figure 4: Flowchart of the Proposed OeSNN-ABC. Source: Created by the authors.

Generally, one can recap the proposed OeSNN-ABC working mechanism in eight steps as below:

Step 1: Initialization Phase:

(a) All the candidate solutions in the population are randomly initiated from the possible ranges of OeSNN hyperparameters.
(b) Initialize the control parameters of **ABC**.
(c) Calculate the objective function (i.e. ***Max(F1 Score)***) using **OeSNN** and evaluate the fitness value for all solutions to determine the ***best solution*** (g^*).

Step 2: Employee Bees Phase:

(a) Generate new solution.
(b) Calculate the objective function using **OeSNN** and evaluate the fitness value for the new solution.
(c) Evaluate the new solution; if an improvement is observed, update current with the new solution.
(d) Repeat steps (a–c) for all solutions.

Step 3: Calculate the **probability value** ***(P)*** for all food sources obtained from employed bees

Step 4: OnLooker Bees Phase:

(a) Select a food source depending on its ***probability (Pi)***.
(b) Generate new solution.
(c) Calculate the objective function using **OeSNN** and evaluate the fitness value for the new solution.
(d) Evaluate the new solution; if an improvement is observed, update current with the new solution.
(e) Repeat steps (b–e) for all solutions.

Step 5: Scout Bees Phase:

(a) Determine an abandoned food source, if exists.
(b) Replace it with the randomly generated new solution.

Step 6: Memorize the ***Best Solution (g*).***
Step 7: Repeat **Steps 2–5** till the termination condition is met.
Step 8: Return *Best Solution (g*)*

Algorithm 1: Pseudo-code of the proposed OeSNN-ABC

1. **Input:**
2. Set up the ABC parameters (Population size (NP), Maximum Cycle Number (MCN), number of dimensions (D), Lower Bound (LB), Upper Bound (UB), and Maximum number of trials (*limit*).
3. Set up the OeSNN parameters (Number of Output Neurons (NOsize), Number of Input Neuron (NIsize), Similarity threshold (SIM), modulation factor (MOD), threshold factor *(C),* and Output value correction factor (ξ).
4. Initialize the population by using equation (10)
5. Evaluate the initial population and determine the ***Best Solution (g*)***
6. Set *iter = 1*
7. **while** *iter* < = *MCN* **do**
8. {————— **Employee Phase** —————}
9. **for** all employee bees **do**
10. Generate a new solution by using equation (11)
11. Calculate the objective function using OeSNN and evaluate the fitness value for the new solution
12. Apply greedy selection between the new solution and the current solution
13. **end for**
14. Calculate the probability *P* for all solutions by using equations (13) and (14)
15. {————— **OnLooker Phase** —————}
16. **for** all onlooker bees **do**
17. Generate random values for *rand* between (0,1)
18. **if** *rand* < *Pi* **then**
19. Generate a new solution by using equation (11)
20. Calculate the objective function using OeSNN and evaluate the fitness function for the new solution
21. Apply greedy selection between the new solution and the current solution
22. **end if**
23. **end for**
24. {————— **Scout Phase** —————}
25. Set *Max_Trial_Index* = 0
26. Set *i = 1*
27. **while** *i* < *NP* **do**
28. **if** *trial(i)* > *trial (Max_Trial_Index)* **then**
29. *Max_Trial_Index* = i
30. **End if**
31. *i* = *i* + 1
32. **end while**
33. **if** *trials(Max_Trial_Index)* > *Limit* **then**
34. replace the solution with a new randomly generated solution by using equation (10)
35. **end if**
36. Memories ***Best Solution (g*)***
37. *iter = iter + 1*
38. **end while**
39. **Output:**
40. **Return *Best Solution*** (*g**)

The evaluation of the proposed method is conducted through the analysis of two popular benchmarking datasets. The evolutionary progression of the method under consideration is examined within this experimental setup, and its effectiveness is evaluated correspondingly. The primary objective of this evaluation is to ascertain the efficacy of the suggested method in developing an OeSNN network. This process encompasses the training of both the OeSNN network and the refined OeSNN-ABC network. Moreover, the abilities of the suggested method have been assessed through a comparison with other detectors in the literature and with five well-known optimization algorithms.

5 Experiment and result

This section shows the experimental evaluation of the proposed OeSNN-ABC method on different datasets (420 time series) and compares it with 15 state-of-the-art anomaly detection approaches. Our experiments will be divided into two parts based on the NAB and Yahoo Webscope datasets. It is crucial to select an appropriate measurement for evaluating the algorithm of anomaly detection. The objective is to choose a metric that is invariant in the presence of an imbalance between normal and anomalous data points and offers an effective basis for comparing the model to other methods. The popular measures of detection quality used to assess our method are precision, recall, $F1$-score, BA, and Matthew's correlation coefficient (MCC). Due to the Numenta and Yahoo Webscope datasets comprising time series from several domains, we have presented the average $F1$ score for each part. The mathematical calculation of these assessment measurements is depicted in the following equations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{15}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{16}$$

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{17}$$

$$\text{BA} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right), \tag{18}$$

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}}, \tag{19}$$

where TP, TN, FP, and FN denote the True Positives, True Negatives, False Positives, and False Negatives, respectively.

5.1 Datasets description

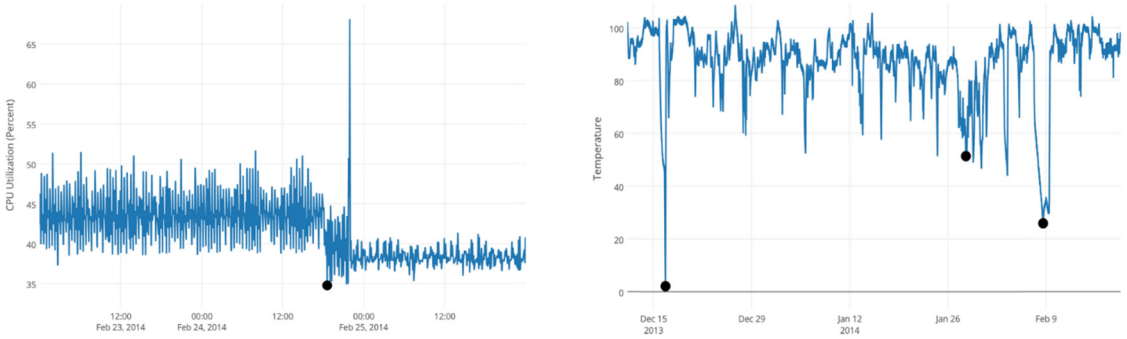
The experimental datasets used in this research were obtained from the literature that is publicly available as anomaly detection benchmarks, namely the NAB [49] and Yahoo Webscope [50]. These two main datasets are separated into various categories, and each one includes numerous time series with labeled anomalies, which vary in length and anomalies contained. Both datasets are commonly used to evaluate the performance and accuracy of unsupervised anomaly detection methods. [14]. Table 1 shows the properties of these two benchmark datasets, which illustrate the total number of data files, the total number of observations, and the number of anomalies in each subgroup of the main datasets. Additionally, it identifies whether the anomalies in the dataset are single "point" or "collective" anomalies.

The NAB dataset is a univariate time series divided into seven labeled sub-datasets, both artificial and real, with six including anomalies and one containing only normal data, which was not considered in the experiments. The number of input values in data files varies between 1,000 and 22,000 instances. Overall, there are 58

Table 1: The NAB and Yahoo Webscope dataset properties

Dataset	Time series	# Data Files	# Instances	# Point anomalies	# Collective anomalies
NAB	ArtificialNoAnomaly	5	20,160	0	0
	ArtificialWithAnomaly	6	24,192	0	6
	RealAdExchange	6	9,610	0	11
	RealAWSCloudWatch	17	67,740	0	19
	RealKnownCause	7	69,561	0	30
	RealTraffic	7	15,664	0	14
	RealTweets	10	158,631	0	33
Yahoo Webscope	A1 benchmark	67	94,866	68	110
	A2 benchmark	100	142,100	33	167
	A3 benchmark	100	168,000	935	4
	A4 benchmark	100	168,000	833	2

time series, and each one includes time stamps and data values files, and the anomaly labels for each data file are defined within a separate set of files. Each time series exhibits a distinct set of characteristics, including temporal noise, short- and long-term periodicities, and concept drift. The data comes from a diverse of sources, including recordings of online advertisement clicks, metrics obtained from AWS servers, real data files such as hourly registered taxi schedules in New York City or CPU utilization, as well as freeway traffic recordings detailing speed or travel time and Twitter volume statistics. Additionally, NAB includes artificially generated data files to assess anomalous behaviors not yet observed in real data. Notably, all-time series within the NAB are imbalanced, with the proportion of anomalies in a time series being less than 10% on average. Figure 5 shows two sample files included in the NAB dataset.



(a) (b)

Figure 5: Examples plots for sample files from NAB: Normal time series shown in blue lines, anomalies are labeled with red circles [8]. (a) AWS CloudWatch CPU utilization data. (b) Machine temperature sensor data.

The Yahoo Webscope datasets consist of four sub-benchmarks, A1, A2, A3, and A4 including 367 real and synthetic time series. Each subgroup contains 1,420–1,680 occurrences. The timestamps of the dataset are observed hourly, and anomalous values are indicated by humans according to the Yahoo S5 dataset's guidelines. In synthetic time series, the anomalous values have appeared in random locations. The A1 benchmark includes 67 data files with real input values reflecting login activities on the Yahoo network. Each file is structured with three main time series: timestamps, input values, and labels indicating whether each input value is anomalous or not. The A2 benchmark comprises 100 synthetic data files, each containing anomalous values in the form of a single instance. Each file in this benchmark includes three time series: timestamp, input values, and labels. Besides, the A3 benchmark consists of 100 synthetic data files with anomalies in the form of single anomalous values. Beyond the three standard time series (timestamps, input values, and anomalies

labels), these data files also include additional time series (trend, noise, seasonality, and changepoint). On the other hand, the A4 benchmark comprises 100 synthetic data files with anomalies. Most anomalies in this benchmark correspond to sudden shifts from one considerably distinct input data trend to another [2,14]. Although both A3 and A4 benchmarks share the same time series types, we utilize only the input values in all the experiments, discarding additional time series. Here, within the Yahoo Webscope dataset, all-time series are imbalanced, with the average proportion of anomalous input values typically falling below 1%. Figure 6 depicts the time series samples from the four sub-benchmarks.

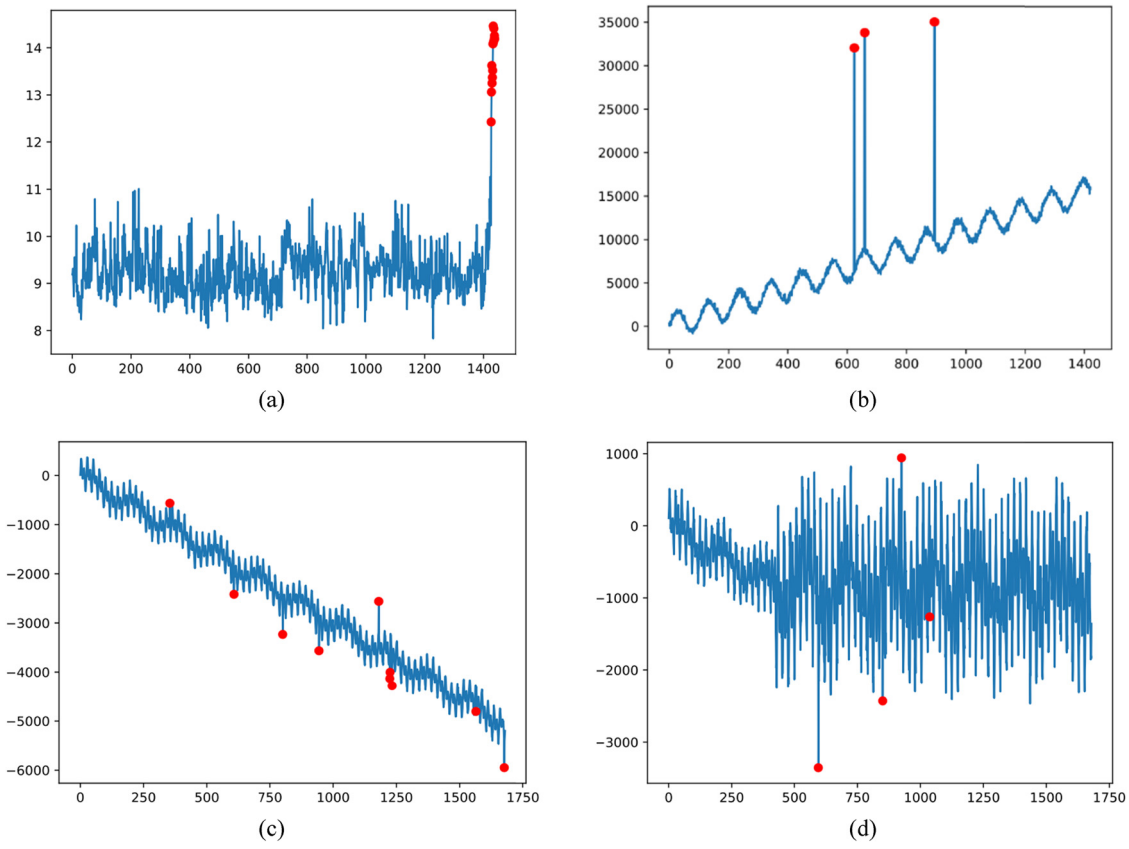


Figure 6: Examples plots for the Yahoo dataset. Normal time series are shown in blue lines, and anomalies are labeled with red circles [51]. (a) A1 benchmark. (b) A2 benchmark. (c) A3 benchmark. (d) A4 benchmark.

5.2 Parameters setting

The ABC parameters were defined before the experimental setup, as listed in Table 2, with colony size value determined according to El-Abd [52]. In Table 3, the values of the hyperparameters of OeSNN are provided based on the literature [2]. The window size *(Wsize)* and anomaly classification factor (ε) have the most significant effect on anomaly detection [8,14], so their values are initialized following guidelines from [2,53,54].

5.3 Results on NAB dataset

Table 4 presents the obtained result for Precision, Recall, average $F1$-score, BA, and MCC produced by OeSNN-ABC for all types of time series in the NAB dataset.

Table 2: Parameters setting of ABC

Parameter	Initialization values
Colony size (CS)	40
Employee bees	20
Onlooker bees	20
Scout bee	1
Trials limit	20
Max iteration	100

Table 3: Parameter settings of OeSNN

Parameter	Description	Initialization values
NIsize	Count of input neurons	10
NOsize	The greatest number of output neurons	50
sim	Similarity factor	0.17
mod	Modulation factor	0.6
C	Fire threshold fraction	0.6
ξ	Error correction factor	0.9
Wsize (for NAB)	Input window size	[100–600]
Wsize (for Yahoo)	Input window size	[50–400]
ε (for NAB)	Anomaly classification factor	[2–7]
ε (for Yahoo)	Anomaly classification factor	[5–17]

Table 4: Evaluation measures of the proposed OeSNN-ABC on the NAB dataset

Dataset	Time series	Prec.	Rec.	*F*1	BA	MCC
NAB	ArtificialWithAnomaly	0.758	0.877	0.800	0.921	0.790
	RealAdExchange	0.414	0.529	0.408	0.706	0.370
	RealAWSCloudwatch	0.523	0.674	0.540	0.783	0.508
	RealKnownCause	0.443	0.534	0.455	0.727	0.413
	RealTraffic	0.541	0.616	0.545	0.776	0.513
	RealTweets	0.321	0.587	0.395	0.712	0.334

Next, we continue to evaluate the proposed OeSNN-ABC method by conducting a comparison against some of the state-of-the-art methods presented by Maciąg et al. [2]. It should be noted from Table 5 that OeSNN-ABC betters the results obtained by the other detectors with a significant margin regarding the average *F*1-score for all NAB data files. The performance on the *ArtificialWithAnomaly* dataset is especially impressive, with a score of 0.800 that significantly outperforms the next top performer, OeSNN-UAD, at 0.427. The superiority trend is consistent across real-world datasets, with OeSNN-ABC maintaining a significant lead. For example, in the *RealTraffic* dataset, OeSNN-ABC scores 0.545, surpassing DeepAnT of 0.223 and OeSNN-UAD of 0.340. This finding confirms that good parameter selection enables the network to ensure the best output by integrating OeSNN with the metaheuristic optimization algorithm.

In Table 6, we provide the obtained result for recall and precision metrics compared with benchmarking anomaly detectors for selected data files from the NAB dataset. A high precision indicates that few normal samples are misclassified as anomalous, whereas a low recall means that many anomalies are missed. In addition, a high recall reflects that the model can identify a significant proportion of positive instances, which is important for detecting as many positives as possible, even if it means producing more false positives [55]. It can be observed that almost all of the detectors achieved high precision and low recall. The primary cause of

Table 5: Average *F*1-score of OeSNN-ABC in comparison against the different state-of-the-art algorithms on the NAB dataset

Time series	Bayesian changepoint	Context OSE	EXPoSE	HTM JAVA	KNN CAD	Numenta	NumentaTM	Relative entropy	Skyline	Twitter ADVec	Windowed Gaussian	DeepAnT	OeSNN-UAD	OeSNN-ABC
ArtificialWithAnomaly	0.009	0.004	0.004	0.017	0.003	0.012	0.017	0.021	0.043	0.017	0.013	0.156	0.427	**0.800**
RealAdExchange	0.018	0.022	0.005	0.034	0.024	0.040	0.035	0.024	0.005	0.018	0.026	0.132	0.234	**0.408**
RealAWSCloudwatch	0.006	0.007	0.015	0.018	0.006	0.017	0.018	0.018	0.053	0.013	0.060	0.146	0.342	**0.540**
RealKnownCause	0.007	0.005	0.005	0.013	0.008	0.015	0.012	0.013	0.008	0.017	0.006	0.200	0.324	**0.455**
RealTraffic	0.012	0.020	0.011	0.032	0.013	0.033	0.036	0.033	0.091	0.020	0.045	0.223	0.340	**0.545**
RealTweets	0.003	0.003	0.003	0.010	0.004	0.009	0.010	0.006	0.035	0.018	0.026	0.075	0.310	**0.395**

The bolded numbers indicate the best results.

Table 6: The result of precision and recall obtained by OeSNN-ABC compared with the selected unsupervised anomaly detectors on the NAB dataset

Time series	Data files	ContextOSE*		NumentaTM*		Skyline*		ADVec*		DeepAnT*		OeSNN-UAD*		OeSNN-ABC	
		Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
RealAWSCloudWatch	ec2-cpu-utilization-5f5533	1.000	0.005	1.000	0.010	1.000	0.002	1.000	0.002	1.000	0.010	0.346	**0.833**	0.237	0.627
	rds-cpu-utilization-cc0c53	1.000	0.005	1.000	0.002	1.000	0.100	0.620	0.012	1.000	0.030	0.501	0.749	0.502	**0.754**
RealAdExchange	exchange-2-cpc-results	0.500	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.330	0.176	0.190	**0.512**	**0.258**
	exchange-3-cpc-results	0.750	0.020	1.000	0.007	0.000	0.000	1.000	0.020	0.710	0.030	0.791	0.222	0.636	**0.458**
RealKnownCause	ambient-temperature- system-failure	0.330	0.001	0.500	0.006	0.000	0.000	0.000	0.000	0.260	0.060	0.288	0.427	0.310	**0.446**
	cpu-utilization-asg- misconfiguration	0.120	0.001	0.520	0.010	0.000	0.000	0.740	0.010	0.630	0.360	0.288	0.652	0.325	**0.719**
	ec2-request-latency-	1.000	0.009	1.000	0.009	1.000	0.014	1.000	0.020	1.000	0.040	0.510	0.225	0.494	**0.341**
	machine-temperature- system-failure	1.000	0.001	0.270	0.004	0.970	0.010	1.000	0.020	0.800	0.001	0.153	**0.892**	0.444	0.414
	nyc-taxi	1.000	0.002	0.850	0.006	0.000	0.000	0.000	0.000	1.000	0.002	0.199	**0.428**	0.351	0.362
	rouge-agent-key-hold	0.330	0.005	0.500	0.005	0.000	0.000	0.000	0.000	0.340	0.050	0.207	**0.642**	0.731	0.458
	rouge-agent-key-updown	0.000	0.000	0.000	0.000	0.000	0.000	0.110	0.002	0.110	0.010	0.334	0.640	**0.451**	**1.000**
RealTraffic	occupancy-6005	0.500	0.004	0.200	0.004	0.500	0.004	0.500	0.004	0.500	0.004	0.235	0.485	0.447	**0.552**
	occupancy-t4013	1.000	0.008	0.660	0.008	1.000	0.040	1.000	0.020	1.000	0.036	0.467	0.508	0.494	**0.652**
	speed-6005	0.500	0.004	0.250	0.008	1.000	0.010	1.000	0.010	1.000	0.008	0.497	0.372	0.710	**0.481**
	speed-7578	0.570	0.030	0.600	0.020	0.860	0.160	1.000	0.010	1.000	0.070	0.511	0.388	0.569	**0.534**
	speed-t4013	1.000	0.008	0.800	0.010	1.000	0.060	1.000	0.010	1.000	0.080	0.545	0.780	0.591	**0.948**
	TravelTime-387	0.600	0.010	0.330	0.004	0.620	0.070	0.200	0.004	1.000	0.004	0.199	**0.606**	0.566	0.241
	TravelTime-451	1.000	0.005	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.009	0.365	0.373	0.414	**0.903**
RealTweets	Twitter-volume-GOOG	0.750	0.002	0.380	0.005	0.590	0.020	0.810	0.010	0.750	0.010	0.142	0.448	0.182	**0.602**
	Twitter-volume-IBM	0.370	0.002	0.220	0.005	0.220	0.010	0.500	0.009	0.500	0.005	0.221	**0.743**	0.424	0.466

The bolded numbers indicate the best results.
*Result presented by Maciąg et al. [2].

this poor recall is the labeling method used in the NAB dataset [14]. The recall value below 0.01 reveals how limited the detector's ability is to detect anomalies. Conversely, the OeSNN-ABC, in most cases, is characterized by much larger values of the recall. Moreover, OeSNN-ABC exhibits notably greater precision and recall for certain data files; thus, it is more efficient in detecting anomalies than the compared approaches.

Table 7 describes evaluation measures for the proposed OeSNN-ABC in contrast to the method introduced by Zhang et al. [56] and the OeSNN-UAD [2] methods for the *RealKnownCause* and *RealTweets* time series from the NAB dataset. Based on the results, it can be observed that the method of Zhang et al. has high precision values but low recall values, which implies that this method has a minimal ability to detect anomalies. That is because high precision values mean that the detector considers a few cases anomalous even if not labeled as anomalies in the data file. On the other hand, low recall values indicate that the OeSNN-UAD detector missed many anomalies. For the data files *machine-temperature system-failure*, *TV-CRM,* and *TV-GOOG*, the result of OeSNN-ABC is second best. Together, the present findings confirm that OeSNN-ABC performs noticeably better than other approaches.

Figure 7 illustrates the performance of the OeSNN-ABC compared to several deep-learning approaches for anomaly detection across multiple time series. It is worth noting that the proposed method consistently outperforms other techniques such as *MadGan, MS Azure (CNN), Dense AE, and RE-ADTS (CV).* The OeSNN-ABC performs notably well on the ArtificialWwithAnomaly scoring 0.80, which is much higher than the next best performer. This reveals that the approach may effectively detect artificially created anomalies. However, the performance on *RealTweets* is relatively modest compared to its success in other datasets, where its efficiency drops to 0.39, demonstrating that the approach may struggle to handle less organized or more dynamic data. In comparison, models like *MadGan* perform better in the *RealTweets* time series, achieving 0.44. For instance, in the *RealAWSCloudWatch*, the OeSNN-ABC lags behind the *Dense AE* model, which outperforms the proposed method by a margin of 0.10.

5.3.1 Comparative performance of OeSNN-ABC with other classifiers

In this subsection, we compare the OeSNN-ABC against three benchmarked classifiers: random forced (RF), support vector machine (SVM), and k-nearest neighbors (kNN). The results in Table 8 demonstrate a comparable pattern of enhanced performance by OeSNN-ABC across a range of evaluation metrics for diverse time series within the NAB dataset. In the *ArtificialWithAnomaly* time series, OeSNN-ABC demonstrated superior performance to SVM and kNN with a value of 0.11, and RF of 0.08, with an $F1$ score of 0.80. This trend of higher precision was observed consistently across all-time series, with OeSNN-ABC consistently outperforming other classifiers. For instance, in the *realAWSCloudWatch* dataset, OeSNN-ABC exhibits a precision of 0.52 and an $F1$ score of 0.54, while the next-best performing classifier, SVM has a precision of 0.12 and an $F1$ score of 0.21. It is important to note, however, that while OeSNN-ABC excels at precision, other classifiers frequently have superior recall rates. This phenomenon is most evident in time series such as the *realAWSCloudWatch*, where all three classifiers achieve a recall of 0.93, in comparison to OeSNN-ABC's 0.67. On the other hand, the consistent increase in $F1$ scores across all datasets indicates that OeSNN-ABC provides a more balanced performance, which is particularly beneficial in situations where false positives and false negatives have comparable weights.

5.3.2 Comparative analysis with other optimization algorithms

Table 9 presents the comparison between OeSNN-ABC and the optimization algorithms: PSO, GWO, FPA, WOA, and GS in terms of $F1$-score. It should be noted that all these algorithms are tested under identical settings to assure fairness. The maximum number of iterations is 100, and the population size is 40. Observation showed that the ABC algorithm showed a notable and promising result when used as an optimizer for the hyperparameters of OeSNN.

Table 7: Result of evaluation measures of the proposed OeSNN-ABC against unsupervised detector and OeSNN-UAD for selected data files in the NAB dataset

Time series	Data files	Unsupervised anomaly detector[a]					OeSNN-UAD[b]					OeSNN-ABC				
		Prec.	Rec.	*F*1	BA	MCC	Prec.	Rec.	*F*1	BA	MCC	Prec.	Rec.	*F*1	BA	MCC
RealKnownCause	ambient-temperature system-failure	0.833	0.007	0.014	0.503	0.070	0.207	0.752	0.325	0.716	0.270	0.310	0.446	**0.366**	0.668	0.287
	cpu-utilization-asgmisconfiguration	0.500	0.001	0.003	0.501	0.022	0.318	0.494	0.387	0.699	0.328	0.325	0.719	**0.448**	0.792	0.416
	ec2-request-latency system-failure	1.000	0.006	0.011	0.503	0.073	0.380	0.402	0.390	0.670	0.332	0.494	0.341	**0.403**	0.654	0.366
	machine-temperature system-failure	0.294	0.002	0.004	0.501	0.018	0.395	0.500	**0.441**	0.707	0.374	0.444	0.414	0.428	0.678	0.368
	nyc-taxi	0.722	0.013	0.025	0.506	0.087	0.166	0.471	0.245	0.604	0.138	0.351	0.362	**0.357**	0.644	0.284
	rouge-agent-key-hold	0.667	0.011	0.021	0.505	0.075	0.126	0.232	0.164	0.526	0.040	0.731	0.458	**0.563**	0.719	0.543
	rouge-agent-key-updown	0.667	0.008	0.015	0.504	0.064	0.251	0.432	0.317	0.645	0.230	0.451	1.000	**0.622**	0.933	0.625
RealTweets	TV-AAPL	0.333	0.004	0.007	0.501	0.026	0.454	0.491	0.472	0.713	0.411	0.493	0.490	**0.491**	0.717	0.435
	TV-AMZN	0.233	0.004	0.009	0.501	0.019	0.170	0.410	0.240	0.593	0.132	0.211	0.322	**0.255**	0.594	0.157
	TV-CRM	0.615	0.005	0.010	0.502	0.049	0.313	0.551	**0.399**	0.708	0.328	0.245	0.846	0.379	0.777	0.350
	TV-CVS	0.600	0.002	0.004	0.501	0.030	0.201	0.543	0.293	0.656	0.210	0.292	0.647	**0.402**	0.740	0.345
	TV-FB	0.375	0.002	0.004	0.501	0.021	0.261	0.180	0.213	0.562	0.146	0.377	0.653	**0.478**	0.767	0.423
	TV-GOOG	0.400	0.003	0.006	0.501	0.027	0.248	0.429	**0.314**	0.650	0.236	0.182	0.602	0.279	0.666	0.208
	TV-IBM	0.429	0.002	0.004	0.501	0.023	0.241	0.284	0.261	0.592	0.172	0.424	0.466	**0.444**	0.698	0.379
	TV-KO	0.286	0.004	0.007	0.501	0.023	0.132	0.339	0.190	0.546	0.063	0.258	0.457	**0.330**	0.655	0.244
	TV-PFE	0.250	0.007	0.013	0.502	0.026	0.188	0.463	0.267	0.620	0.168	0.218	0.780	**0.340**	0.734	0.293
	TV-UPS	0.389	0.009	0.017	0.504	0.046	0.463	0.428	0.445	0.687	0.387	0.508	0.609	**0.554**	0.772	0.502

The bolded numbers indicate the best results.

[a,b]Result reported by Maciąg et al. [2].

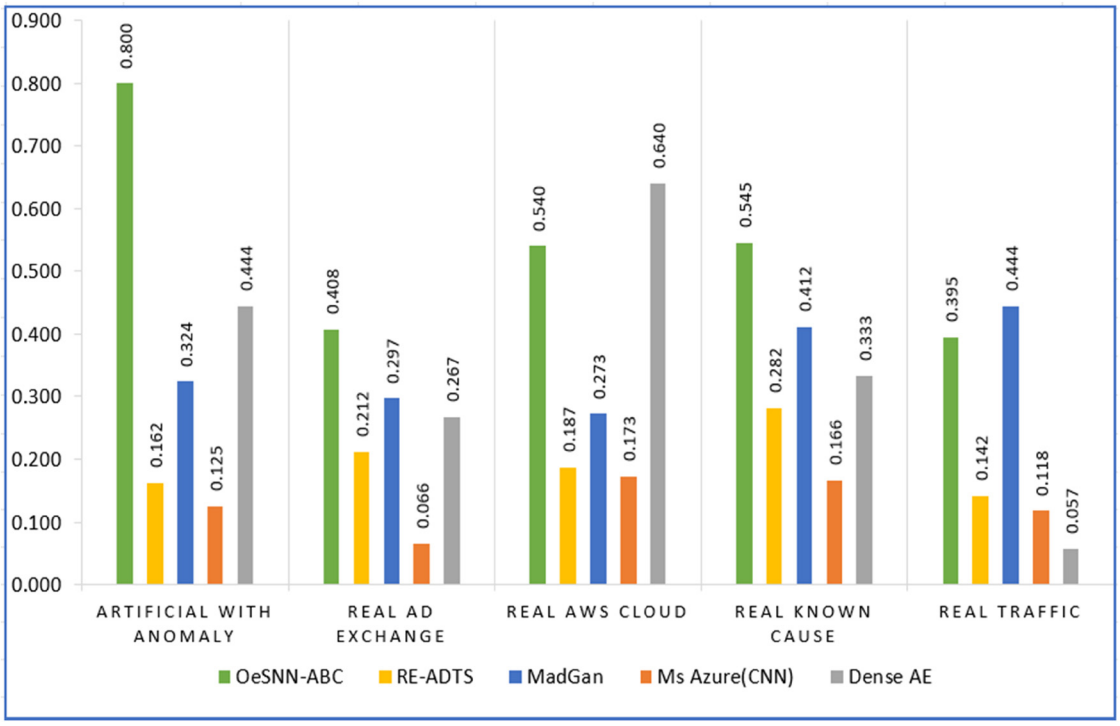


Figure 7: *F*1-score measure of OeSNN-ABC against deep learning approaches MadGan, MS Azure, Dense AE are presented in [57], RE-ADTS [58], and AWRM [59], Source: Created by the authors.

Table 8: Comparison of the proposed method (OeSNN-ABC) against competitive classifiers on the NAB dataset

Time series	Measure	RF*	SVM*	kNN*	OeSNN-ABC
ArtificialWithAnomaly	Precision	0.04	0.06	0.06	**0.76**
	Recall	0.83	**1**	**1**	0.88
	*F*1 score	0.08	0.11	0.11	**0.80**
realAdExchange	Precision	0.26	0.28	0.26	**0.41**
	Recall	**0.71**	**0.71**	**0.71**	0.53
	*F*1 score	0.38	0.4	0.38	**0.41**
realAWSCloudwatch	Precision	0.11	0.1	0.12	**0.52**
	Recall	**0.93**	**0.93**	**0.93**	0.67
	*F*1 score	0.19	0.19	0.21	**0.54**
realKnownCause	Precision	0.06	0.07	0.05	**0.44**
	Recall	**0.63**	**0.63**	0.58	0.53
	*F*1 score	0.11	0.13	0.1	**0.46**
realTraffic	Precision	0.22	0.24	0.28	**0.54**
	Recall	**0.79**	**0.79**	**0.79**	0.62
	*F*1 score	0.34	0.37	0.41	**0.55**
realTweets	Precision	0.05	0.05	0.05	**0.32**
	Recall	**1**	0.94	0.97	0.59
	*F*1 score	0.1	0.1	0.1	**0.40**

The bolded numbers indicate the best results.
*Result presented by Sina and Thomas [60].

Furthermore, the experimental results presented in Figure 8 offer valuable insights into the performance of the proposed method across a range of diverse datasets, when compared to other optimization algorithms. OeSNN-ABC demonstrates consistent superiority over other algorithms, exhibiting exceptional efficacy in key

Table 9: Result of average *F*1 score evaluation of OeSNN integrated with ABC, PSO, GWO, FPA, WOA, and GS for the NAB dataset

Time series	PSO	GWO	FPA	WOA	GS	ABC
ArtificialWithAnomaly	0.787	0.781	0.773	0.714	0.427	**0.800**
RealAdExchange	0.379	0.349	0.343	0.203	0.234	**0.408**
RealAWSCloudwatch	0.512	0.536	0.500	0.470	0.342	**0.540**
RealKnownCause	0.420	0.438	0.414	0.409	0.324	**0.455**
RealTraffic	0.529	0.510	0.505	0.477	0.340	**0.545**
RealTweets	0.381	0.367	0.339	0.325	0.310	**0.395**

The bolded numbers indicate the best results.

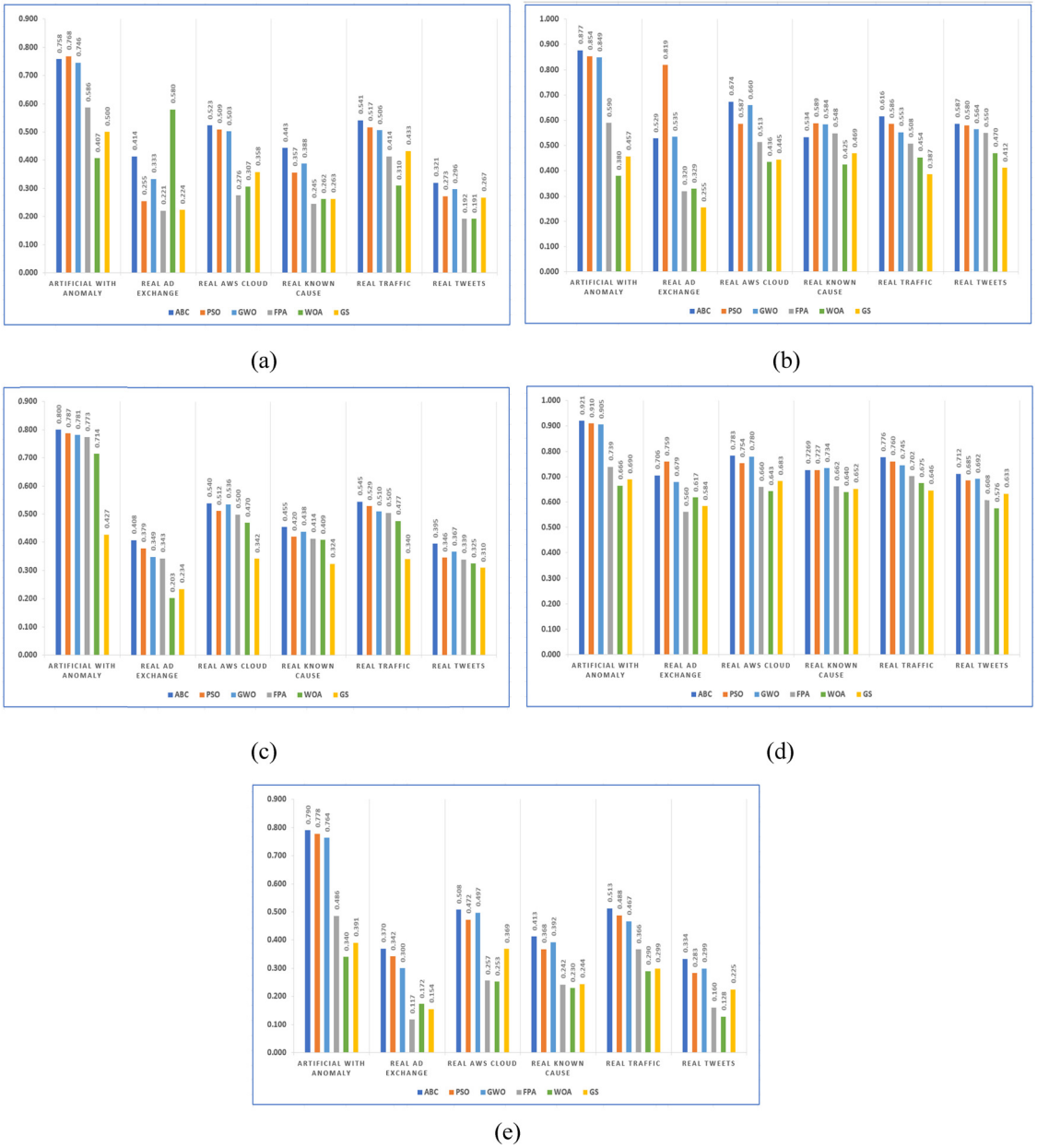


Figure 8: Evaluation measures of OeSNN-ABC against PSO, GWO, FPA, WOA, and GS optimization algorithms: (a) Precision. (b) Recall. (c) *F*1-score. (d) BA. (e) MCC. Source: Created by the authors.

metrics including precision, recall, *F*1-score, BA, and MCC. It is noteworthy that while algorithms like PSO and GWO exhibit competitive performance on *RealAdExchange* and *RealKnownCause,* respectively, they lack the consistent performance of OeSNN-ABC across a wide range of datasets.

In Table 10, we present the optimal values of *Wsize* and anomaly classification factor ε obtained for the OeSNN-ABC as compared to other optimization algorithms across different time series that were used in the experiments of the results, which are presented in Table 6. It can be observed from Table 10 that for every time series, the combination of hyperparameter values is different. Thus, to achieve good performance by OeSNN, a correct combination of hyperparameters is required. Once the algorithm finds a correct combination of values for the hyperparameters, the OeSNN will classify at the best rate of accuracy.

Table 10: Optimal window sizes and anomaly classification factors used by ABC, PSO, GWO, FPA, WOA, and GS for data files from Table 6

Time series	Data files	ABC		PSO		FPA		GWO		WOA		GS*	
		Wsize	ε	Wsize	ε	Wsize	ε	Wsize	ε	Wsize	ε	Wsize	ε
RealAWSCloudWatch	ec2-cpu-utilization-5f5533	220	3	310	2	150	3	150	2	210	6	100	2
	rds-cpu-utilization-cc0c53	600	9	600	11	600	11	600	13	600	11	100	4
RealAdExchange	exchange-2-cpc-results	70	7	100	2	100	2	110	2	100	2	300	2
	exchange-3-cpc-results	100	4	100	2	100	4	100	4	110	4	600	7
RealKnownCause	ambient-temperature- system-failure	340	8	530	2	320	4	290	4	230	3	500	6
	cpu-utilization-asg-misconfiguration	600	3	510	2	590	3	600	3	600	3	600	3
	ec2-request-latency-	100	4	390	6	600	4	100	4	100	4	400	5
	machine-temperature- system-failure	120	15	450	9	100	14	120	16	430	10	300	4
	nyc-taxi	120	3	590	2	270	3	120	3	100	3	100	3
	rouge-agent-key-hold	70	17	260	2	190	17	190	19	190	4	100	5
	rouge-agent-key-updown	380	17	380	2	370	16	450	5	380	15	300	6
RealTraffic	occupancy-6005	50	6	170	2	600	5	160	2	310	6	300	2
	occupancy-t4013	600	2	590	2	600	3	500	3	100	4	600	2
	speed-6005	260	3	460	2	290	3	260	3	190	4	600	2
	speed-7578	30	6	600	2	150	6	110	3	110	4	100	4
	speed-t4013	250	3	470	2	260	3	250	3	470	3	400	3
	TravelTime-387	490	6	100	2	340	7	100	4	370	7	100	2
	TravelTime-451	120	4	130	4	140	6	100	8	130	6	100	5
RealTweets	Twitter-volume-GOOG	330	3	340	2	240	5	320	3	600	2	400	3
	Twitter-volume-IBM	80	9	280	2	170	4	130	6	130	7	150	5

*Result presented by Maciąg et al. [2].

5.4 Results on the Yahoo dataset

The average values for the precision, recall, average *F*1-score, BA, and MCC for all types of time series in the Yahoo Webscope dataset obtained by the proposed method are presented in Table 11.

Table 11: Result of evaluation measures of proposed OeSNN-ABC on the Yahoo Webscope dataset

Time series	Prec.	Rec.	*F*1	BA	MCC
A1 benchmark	0.876	0.828	0.832	0.911	0.838
A2 benchmark	0.910	0.934	0.881	0.966	0.895
A3 benchmark	0.826	0.477	0.571	0.738	0.606
A4 benchmark	0.668	0.450	0.469	0.724	0.505

In Table 12, we show the comparison of the obtained average *F*1-score values for each of the Yahoo Webscope categories for the OeSNN-ABC against state-of-the-art approaches: DeepAnT, EGADS, and Twitter presented in [14]; OeSNN-D presented in [28]; and OeSNN-UAD presented in [2]. The suggested OeSNN-ABC method posed beneficial outcomes in some cases, particularly in the A1 and A2 benchmarks, with scores of 0.832 and 0.881, respectively, greatly exceeding most approaches and exhibiting competitive results to the outcome provided by DeepAnT. However, the findings emphasize the context-dependent nature of anomaly detection techniques. For example, DeepAnT performs well throughout the synthesis time series A2, A3, and A4 standards, but is downgraded in the real time series in A1 with scores of 0.460.

Table 12: Result of average *F*1-score evaluation of OeSNN-ABC compared with the selected unsupervised anomaly detectors on the Yahoo Webscope dataset

Time series	Yahoo EGADS	Twitter anomaly	DeepAnT[a]	OeSNN-UAD	OeSNN-D[b]	OeSNN-ABC
A1 benchmark	0.470	0.470	0.460	0.700	0.405	**0.832**
A2 benchmark	0.580	0.000	**0.940**	0.690	0.451	0.881
A3 benchmark	0.480	0.300	**0.870**	0.410	0.110	0.571
A4 benchmark	0.290	0.340	**0.680**	0.340	0.147	0.469

The bolded numbers indicate the best results.
[a]DeepAnT presented by Munir et al. [14].
[b]OeSNN-D presented by Bäßler et al. [28].

Figure 9 depicts the results of a comparative performance analysis of various deep learning anomaly detection models over four-time series (A1–A4). It is worth noting that the OeSNN-ABC received the highest scores in both A1 benchmark and A2 benchmark, with scores of 0.83 and 0.88, respectively, indicating remarkable performance. It outperformed competitor models such as MadGan and MS Azure (CNN). These results indicate that the approach works well for processing certain forms of time series data. However, the OeSNN-

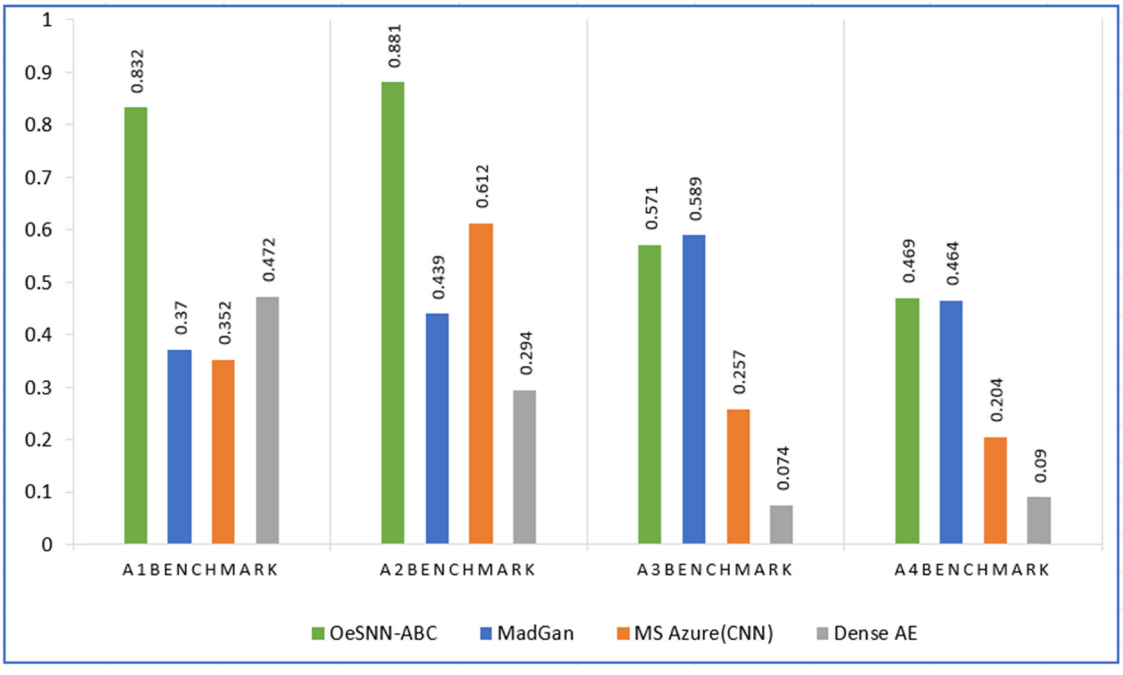


Figure 9: *F*1-score measure of OeSNN-ABC against deep learning approaches: MadGan [57], MS Azure (CNN) [58], Dense AE [58], RE-ADTS (CV) [59], and AWRM [60]. Source: Created by the authors.

ABC performs less well in the A3 and A4 benchmarks, scoring 0.57 and 0.46, respectively. In these cases, the proposed method either matches or falls behind MadGan, which has a higher A3 score (0.58).

5.4.1 Comparative performance of OeSNN-ABC with other classifiers

Table 13 describes the result of comparing the OeSNN-ABC with the three well-known classifiers: RF, SVM, and kNN. It can be observed, in the context of the A1 and A2 benchmarks, that the OeSNN-ABC consistently outperforms other classifiers, exhibiting superior precision, recall, and *F*1 scores. Notably, OeSNN-ABC in the A1 benchmark achieves the highest *F*1 score of 0.83. Similarly, for the A2 benchmark, OeSNN-ABC maintains its lead with an *F*1 score of 0.88, followed closely by SVM with a score of 0.90. However, performance conditions change in the A3 and A4 benchmarks, and traditional classifiers show competitive or slightly better results. For example, in A3, kNN achieved the highest *F*1 score of 0.76 compared with OeSNN-ABC of 0.57. Even though the OeSNN-ABC maintains a high precision of 0.83, but struggles with recall of 0.48, indicating it may be less precise when detecting anomalies in this dataset. For the A4 benchmark, all classifiers surpass OeSNN-ABC in terms of *F*1 scores. SVM performs best overall, with an *F*1 score of 0.66, followed by RF with 0.65 and kNN with 0.63. The precision of OeSNN-ABC with a value of 0.67 is comparable to other approaches, but its recall of 0.45 is lower, implying that it may miss some anomalies in this time series.

Table 13: Comparison of the proposed method (OeSNN-ABC) against competitive classifiers on the Yahoo Webscope dataset

Time series	Measure	RF*	SVM*	kNN*	OeSNN-ABC
A1 benchmark	Precision	0.4	0.46	0.44	**0.88**
	Recall	0.77	0.76	0.76	**0.83**
	*F*1 score	0.52	0.58	0.56	**0.83**
A2 benchmark	Precision	0.48	0.82	0.48	**0.91**
	Recall	**0.98**	**0.98**	**0.98**	0.93
	*F*1 score	0.65	**0.90**	0.65	0.88
A3 benchmark	Precision	0.84	0.83	**0.87**	0.83
	Recall	**0.69**	0.64	0.68	0.48
	*F*1 score	0.75	0.72	**0.76**	0.57
A4 benchmark	Precision	0.69	**0.71**	0.68	0.67
	Recall	**0.61**	**0.61**	0.59	0.45
	*F*1 score	0.65	**0.66**	0.63	0.47

The bolded numbers indicate the best results.
*Result presented by Jensen [61].

5.4.2 Comparative analysis with other optimization algorithms

In this subsection, OeSNN-ABC is compared with optimization algorithms: PSO, GWO, FPA, WOA, and GS. It can be viewed from the reported result in Table 14 that OeSNN-ABC outperforms other methods in terms of average *F*1-score for A1 benchmark and A2 benchmark categories. For A3 benchmarks and A4 benchmarks, the FPA algorithm exhibits a better average *F*1 score than other methods. This is due to FPA's higher recall value compared to OeSNN-ABC, which positively impacts its *F*1-score performance.

Figure 10 depicts the evaluation metrics for the proposed method OeSNN-ABC against other optimization algorithms utilized for comparison, including Precision, Recall, average *F*1-score, BA, and MCC. It is worth noting that OeSNN-ABC produced the best results in terms of the average *F*1-score for the A1 and A2 benchmarks time series, demonstrating its proficiency in balancing precision and recall across diverse anomaly characteristics. It is notable that FPA outperforms other methods for the A3 and A4 benchmarks time series. This superiority can be due to FPA's greater recall values, which greatly impact the accuracy.

Table 14: Result of average *F*1 score evaluation of OeSNN integrated with ABC, PSO, GWO, FPA, WOA, and GS for the YAHOO dataset

Time series	PSO	GWO	FPA	WOA	GS	ABC
A1 benchmark	0.565	0.781	0.766	0.771	0.697	**0.832**
A2 benchmark	0.418	0.867	0.820	0.835	0.69	**0.881**
A3 benchmark	0.325	0.534	**0.753**	0.502	0.409	0.571
A4 benchmark	0.244	0.424	**0.765**	0.396	0.342	0.469

The bolded numbers indicate the best results.

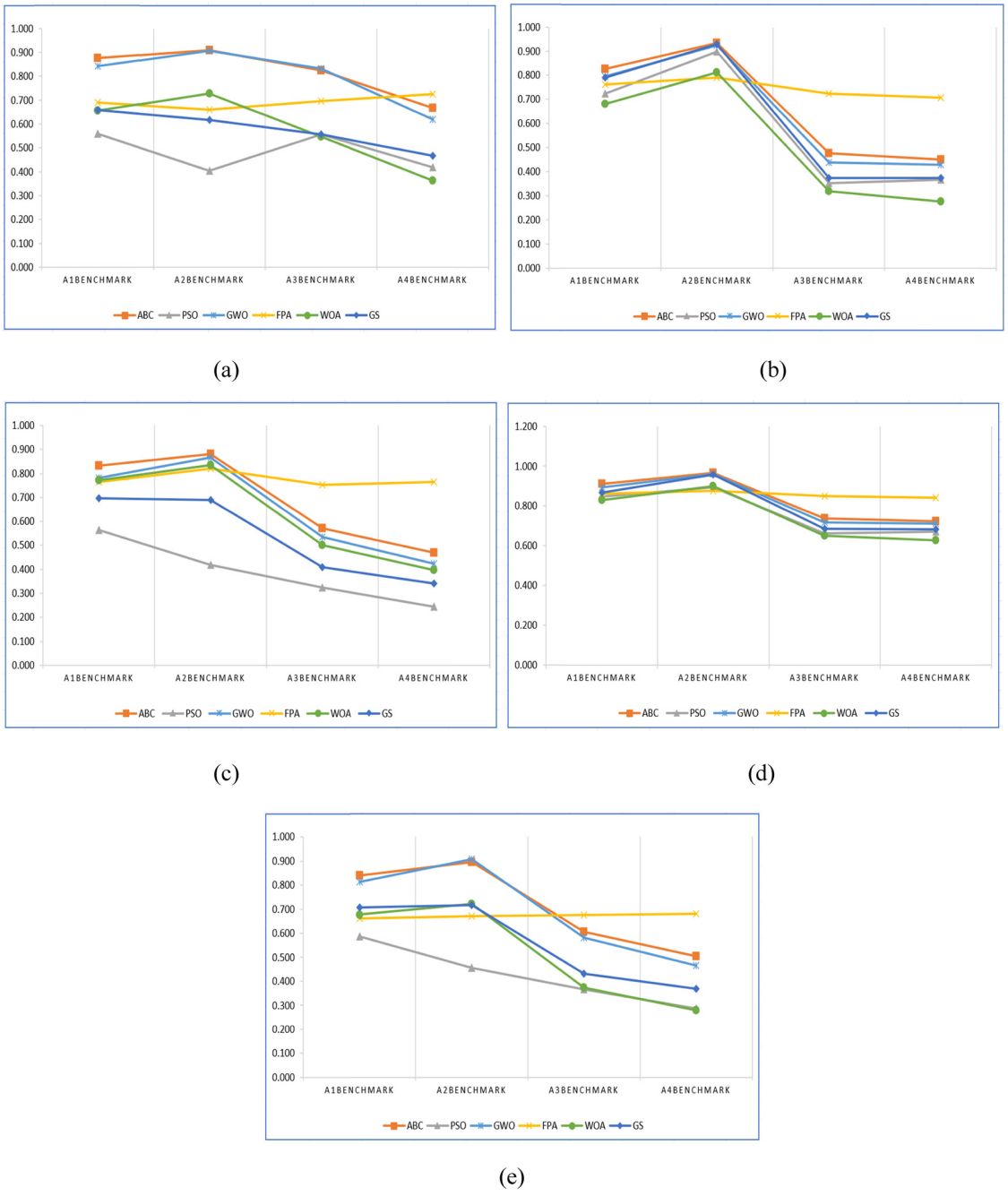


Figure 10: Evaluation measures of OeSNN-ABC against PSO, GWO, FPA, WOA, and GS optimization algorithms: (a) Precision. (b) Recall. (c) *F*1-score. (d) BA. (e) MCC. Source: Created by the authors.

5.5 Significance analysis

To further validate the outcomes provided by the proposed OeSNN-ABC method, the *t-test* is used in this study to verify if a statistically significant difference exists between OeSNN-ABC and other state-of-the-art. The null hypothesis shows that the proposed model and other models' accuracy do not significantly differ. The null hypothesis is not rejected when the state level is more than 0.05, and it is rejected when the state level is less than 0.05. The p values in Table 15 indicate that the proposed OeSNN-ABC outperforms alternative techniques significantly. As a result, the null hypothesis is rejected at the default 5% significance level.

Table 15: Statistical results of the OeSNN-ABC concerning T-TEST against other detectors for the NAB dataset

Dataset	Methods	*P*_Value	OeSNN-ABC
NAB	Bayesian Changepoint	<0.00001	Significant
	Context OSE	<0.00001	Significant
	EXPoSE	<0.00001	Significant
	HTM JAVA	<0.00001	Significant
	KNN CAD	<0.00001	Significant
	Numenta	<0.00001	Significant
	NumentaTM	<0.00001	Significant
	Relative Entropy	<0.00001	Significant
	Skyline	<0.00002	Significant
	Twitter ADVec	<0.00001	Significant
	Windowed Gaussian	<0.00001	Significant
	DeepAnt	<0.00020	Significant
	OeSNN-UAD	0.01474	Significant

For additional verification of the obtained results, the Wilcoxon signed-rank test was carried out to assess if the proposed OeSNN-ABC significantly outperformed the PSO, GWO, FPA, WOA, and GS optimization algorithms. Table 16 presents the outcome of the test, which is conducted at a significant level of 5%. From this table, we can conclude that the gaps in the performance are statistically significant for OeSNN-ABC against other optimization algorithms; so, the null hypothesis is rejected at the 5% significance level.

Table 16: Statistical results of the proposed OeSNN-ABC with respect to the Wilcoxon signed-rank test for the NAB dataset

Dataset	Algorithm	*P*_Value	OeSNN-ABC
NAB	PSO	<0.00001	Significant
	GWO	<0.00001	Significant
	FPA	<0.00001	Significant
	WOA	<0.00001	Significant
	GS	<0.00001	Significant

The study emphasizes that the combination of optimizers, in particular the ABC, significantly improves the results of the OeSNN model for unsupervised anomaly detection. By optimizing the OeSNN hyperparameters, window size ($Wsize$), and anomaly classification factor (ε), the ABC algorithm helps to find the best structure of the model, resulting in a higher $F1$ score compared to the OeSNN-UAD (unoptimized OeSNN). Moreover, the results show that OeSNN-ABC improves the precision and recall and achieves a balance between accuracy and recall compared to other existing anomaly detection methods, especially on the NAB dataset and specific categories of the Yahoo Webscope dataset (A1 and A2 benchmarks). This implies that the optimization process enables the model to identify a greater number of true anomalies (higher recall) while reducing the number of

false positives (higher precision). Furthermore, the results of the statistical tests (*t*-test and Wilcoxon signed-rank test) demonstrated that the performance differences between the proposed method and unoptimized methods, as well as OeSNN when optimized by other metaheuristic algorithms, were statistically significant.

Although the proposed method has shown a good performance, there are some limitations, which are provided as follows:

- The current study has considered only two hyperparameters: window size (*Wsize*) and anomaly classification factor (ε).
- The study only used benchmark datasets (NAB and Yahoo Webscope) to validate the performance of the proposed method.
- The evaluations of the proposed method did not consider the measurement of execution time.

6 Conclusion and future work

In this research, we proposed the OeSNN-ABC method to optimize the OeSNN hyperparameters named window size (*Wsize*) and anomaly classification factor (ε) rather than manually selecting these hyperparameter values. The proposed OeSNN-ABC was adopted in unsupervised anomaly detection for streaming data. The OeSNN-ABC was compared with fifteen detectors in the literature to evaluate the quality of the proposed method. The results of experiments indicated that the OeSNN-ABC offers a noteworthy result in overall performance compared with other detectors for the NAB dataset and for most time series in the Yahoo Webscope dataset. Meanwhile, we evaluate the performance of OeSNN-ABC in comparison with popular optimization algorithms, including PSO, FPA, GWO, WOA, and GS, alongside other classifiers such as RF, SVM, and kNN. The result demonstrated that OeSNN-ABC consistently outperforms these alternative optimization algorithms and classifiers, particularly on the NAB dataset and specific categories of the Yahoo Webscope dataset (A1 and A2 benchmarks), in terms of the average *F*1-score. In general, the outcomes of experiments reveal that integrating OeSNN with swarm intelligence-based optimization algorithms, particularly the ABC algorithm, may improve the performance of anomaly detection systems and might be applicable in different fields.

Regarding future work, there is potential for further enhancement of the suggested method through the refinement of the OeSNN structure alongside the optimization of hyperparameters: window size (*Wsize*) and anomaly classification factor (ε). This could involve optimizing the pre-synaptic neurons and hyperparameters of OeSNN simultaneously by using a multi-objective method to achieve an optimized trade-off between accuracy and network complexity.

Further, more investigation on diverse real-world datasets from various domains is needed to ascertain the generalization claims of the proposed method. Besides that, additional work could be directed towards the integration of hybridized OeSNN-ABC with alternative metaheuristic algorithms, to leverage the strengths of other optimization techniques, aiming to enhance the efficiency and accuracy of anomaly detection. Additionally, the execution time needs to be explored to establish a fair comparison with other methodologies.

Acknowledgments: We thank the anonymous reviewers for their comments and suggestions.

Funding information: This work was funded by the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Selangor, Malaysia.

Author contributions: Rabie Rehan: conceptualization, data curation, investigation, methodology and writing original draft; Shahnorbanun Sahran: conceptualization and funding; Zaid Abdi Alkareem Alyasseri: formal analysis; Nor Samsiah Sani: supervision; Mohammed Azmi Al-Betar: review and editing; All authors have read and agreed to the published version of the manuscript.

Conflict of interest: Authors state no conflict of interest.

Data availability statement: The dataset is publicly available for download from: https://github.com/numenta/NAB (Figure 5); https://webscope.sandbox.yahoo.com/.

References

[1] Belay MA, Blakseth SS, Rasheed A, Rossi PS. Unsupervised anomaly detection for IoT-based multivariate time series: Existing solutions, performance analysis, and future directions. Sensors. 2023;23(5):2844. doi: 10.3390/s23052844.

[2] Maciąg PS, Kryszkiewicz M, Bembenik R, Lobo JL, Del Ser J. Unsupervised anomaly detection in stream data with online evolving spiking neural networks. Neural Netw. 2021;139:118–39. doi: 10.1016/j.neunet.2021.02.017.

[3] Alomari ES, Nuiaa RR, Alyasseri ZAA, Mohammed HJ, Sani NS, Esa MI, et al. Malware Detection Using Deep Learning and Correlation-Based Feature Selection. Symmetry (Basel). 2023;15:123. doi: 10.3390/sym15010123.

[4] Ahmad Z, Shahid Khan A, Nisar K, Haider I, Hassan R, Haque MR, et al. Anomaly detection using deep neural network for IoT architecture. Appl Sci. 2021;11(15):7050. doi: 10.3390/app11157050.

[5] Tarish HA, Hassan R, Jaber MM. Network security framework for Internet of medical things applications: A survey. J Intell Syst. 2024;33(1):20230220. doi: 10.1515/jisys-2023-0220.

[6] Tareq M, Sundararajan EA, Mohd M, Sani NS. Online clustering of evolving data streams using a density grid-based method. IEEE Access. 2020;8:166472–90. doi: 10.1109/ACCESS.2020.3021684.

[7] Xiaolan W, Manjur Ahmed M, Nizam Husen M, Qian Z, Belhaouari SB. Evolving anomaly detection for network streaming data. Inf Sci (Ny). 2022;608:757–77. doi: 10.1016/j.ins.2022.06.064.

[8] Ahmad S, Lavin A, Purdy S, Agha Z. Unsupervised real-time anomaly detection for streaming data. Neurocomputing. 2017;262:134–47. doi: 10.1016/j.neucom.2017.04.070.

[9] Lobo JL, Del Ser J, Bifet A, Kasabov N. Spiking neural networks and online learning: an overview and perspectives. Neural Netw. 2020;121:88–100. doi: 10.1016/j.neunet.2019.09.004.

[10] Almasri AH, Sahran S. Time window, spike time and threshold boundary for spiking neural network applications. J Appl Sci. 2014;14(4):317–24. doi: 10.3923/jas.2014.317.324.

[11] Javanshir A, Nguyen TT, Mahmud MAP, Kouzani AZ. Training spiking neural networks with metaheuristic algorithms. Appl Sci. 2023;13(8):4809. doi: 10.3390/app13084809.

[12] Schliebs S, Kasabov N. Evolving spiking neural network-a survey. Evol Syst. 2013;4(2):87–98. doi: 10.1007/s12530-013-9074-9.

[13] Ibad T, Kadir A, Binti N, Aziz AB. Evolving spiking neural network: a comprehensive survey of its variants and their results. J Theor Appl Inf Technol. 2020;31:24.

[14] Munir M, Siddiqui SA, Dengel A, Ahmed S. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. IEEE Access. 2019;7:1991–2005. doi: 10.1109/ACCESS.2018.2886457.

[15] Clark J, Liu Z, Japkowicz N. Adaptive threshold for outlier detection on data streams. Proc. - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics DSAA 2018; 2018. p. 41–9. doi: 10.1109/DSAA.2018.00014.

[16] Ibad T, Abdulkadir SJ, Aziz N, Ragab MG, Al-Tashi Q. Hyperparameter optimization of evolving spiking neural network for time-series classification. N Gener Comput. 2022;40(1):377–97. doi: 10.1007/s00354-022-00165-3.

[17] Roslan F, Hamed HN, Isa MA. The enhancement of evolving spiking neural network with firefly algorithm. J Telecommun Electron Comput Eng. 2017;9(3–3):63–6.

[18] Saleh AY, Shamsuddin SM, Nuzly Abdull Hamed H. A hybrid differential evolution algorithm for parameter tuning of evolving spiking neural network. Int J Comput Vis Robot. 2017;7(2):20–34. doi: 10.1504/IJCVR.2017.081231.

[19] Akay B, Karaboga D, Gorkemli B, Kaya E. A survey on the artificial bee colony algorithm variants for binary, integer and mixed integer programming problems. Appl Soft Comput. 2021;106:107351. doi: 10.1016/j.asoc.2021.107351.

[20] Ismail AM, Remli MA, Choon YW, Nasarudin NA, Ismail NS, Ismail MA, et al. Artificial Bee Colony algorithm in estimating kinetic parameters for yeast fermentation pathway. J Integr Bioinform. 2023;20(2):20220051. doi: 10.1515/jib-2022-0051.

[21] Hajisalem V, Babaie S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. Comput Netw. 2018;136:37–50. doi: 10.1016/j.comnet.2018.02.028.

[22] Ghanem WAHM, Jantan A, Ghaleb SAA, Nasser AB. An efficient intrusion detection model based on hybridization of artificial bee colony and dragonfly algorithms for training multilayer perceptrons. IEEE Access. 2020;8:130452–75. doi: 10.1109/ACCESS.2020.3009533.

[23] Khanna M, Chaudhary A, Toofani A, Pawar A. Performance comparison of multi-objective algorithms for test case prioritization during web application testing. Arab J Sci Eng. 2019;44(11):9599–625. doi: 10.1007/S13369-019-03817-7/METRICS.

[24] Yin C, Zhang S, Wang J, Xiong NN. Anomaly detection based on convolutional recurrent autoencoder for IoT time series. IEEE Trans Syst Man, Cybern Syst. 2022;52(1):112–22. doi: 10.1109/TSMC.2020.2968516.

[25] Reid D, Hussain AJ, Tawfik H. Financial time series prediction using spiking neural networks. PLoS One. 2014;9(8):1–13. doi: 10.1371/journal.pone.0103656.

[26] Demertzis K, Iliadis L, Bougoudis I. Gryphon: a semi-supervised anomaly detection system based on one-class evolving spiking neural network. Neural Comput Appl. 2019;32:4303–14. doi: 10.1007/s00521-019-04363-x.
[27] Lobo JL, Laña I, Del Ser J, Bilbao MN, Kasabov N. Evolving spiking neural networks for online learning over drifting data streams. Neural Netw. 2018;108:1–19. doi: 10.1016/J.NEUNET.2018.07.014.
[28] Bäßler D, Kortus T, Gühring G. Unsupervised anomaly detection in multivariate time series with online evolving spiking neural networks. Mach Learn. 2022;111(4):1377–408. doi: 10.1007/s10994-022-06129-4.
[29] Khanna M, Singh LK, Shrivastava K, Singh R. An enhanced and efficient approach for feature selection for chronic human disease prediction: A breast cancer study. Heliyon. 2024;10:1–21. doi: 10.1016/j.heliyon.2024.e26799.
[30] Khanna M, Singh LK, Garg H. A novel approach for human diseases prediction using nature inspired computing & machine learning approach. Multimed Tools Appl. 2024;83(6):17773–809. doi: 10.1007/S11042-023-16236-6/METRICS.
[31] Md. Said NN, Hamed HNA, Abdullah A. The enhancement of evolving spiking neural network with dynamic population particle swarm optimization. In Communications in computer and information science. Vol. 752, Singapore: Springer; 2017. p. 95–103, doi: 10.1007/978-981-10-6502-6_8.
[32] Yusuf ZM, Hamed HN, Yusuf LM, Isa MA. Evolving spiking neural network (ESNN) and harmony search algorithm (HSA) for parameter optimization. In 2017 6th International Conference on Electrical Engineering and Informatics; 2017, p. 1–6. doi: 10.1109/ICEEI.2017.8312365.
[33] Braei M, Wagner S. Anomaly detection in univariate time-series: A survey on the state-of-the-art. ArXiv, Prepr 2020. doi: 10.48550/arXiv.2004.00433.
[34] Sadeghi Z, Matwin S. Anomaly detection for maritime navigation based on probability density function of error of reconstruction. J Intell Syst. 2023;32(1):1–14. doi: 10.1515/jisys-2022-0270.
[35] Roel B, Zaharah B, Heskes T. Unsupervised anomaly detection algorithms on real-world data: how many do we need? J Mach Learn Res. 2023;25:5199–232. doi: 10.48550/arXiv.2305.00735.
[36] Mustafa HMJ, Ayob M. Enhanced connectivity validity measure based on outlier detection for multi-objective metaheuristic data clustering algorithms. Appl Comput Intell Soft Comput. 2022;2022:1–10. doi: 10.1155/2022/1036293.
[37] Wysoski SG, Benuskova L, Kasabov N. On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Vol. 4131 LNCS; 2006. p. 61–70. doi: 10.1007/11840817_7.
[38] Kasabov NK. Time-space, spiking neural networks and brain-inspired artificial intelligence. vol. 7, Berlin, Heidelberg: Springer Berlin Heidelberg; 2019. doi: 10.1007/978-3-662-57715-8.
[39] Sabri N, Hamed HNA, Ibrahim Z, Ibrahim K, Isa MA. Integrated evolving spiking neural network and feature extraction methods for scoliosis classification. Comput Mater Contin. 2022;73(3):5559–73. doi: 10.32604/cmc.2022.029221.
[40] Maciąg PS, Bembenik R, Piekarzewicz A, Del Ser J, Lobo JL, Kasabov NK. Effective air pollution prediction by combining time series decomposition with stacking and bagging ensembles of evolving spiking neural networks. Env Model Softw. 2023;170:105851. doi: 10.1016/j.envsoft.2023.105851.
[41] Schliebs S, Defoin-Platel M, Kasabov N. Integrated feature and parameter optimization for an evolving spiking neural network. ICONIP 2008 Lect Notes Comput Sci. 2009;5506:1229–36. doi: 10.1007/978-3-642-02490-0_149.
[42] AL-Gburi AFJ, Nazri MZA, Bin Yaakub MR, Alyasseri ZAA. A systematic review of symbiotic organisms search algorithm for data clustering and predictive analysis. J Intell Syst. 2024;33:1–27. doi: 10.1515/jisys-2023-0267.
[43] Alyasseri ZA, Al-Betar MA, Awadallah MA, Makhadmeh SN, Abasi AK, Doush IA, et al. A hybrid flower pollination with β-hill climbing algorithm for global optimization. J King Saud Univ - Comput Inf Sci. 2022;34(8):4821–35. doi: 10.1016/j.jksuci.2021.06.015.
[44] Singh LK, Khanna M, Thawkar S, Singh R. Collaboration of features optimization techniques for the effective diagnosis of glaucoma in retinal fundus images. Adv Eng Softw. 2022;173:103283. doi: 10.1016/J.ADVENGSOFT.2022.103283.
[45] Hussein WA, Sahran S, Sheikh Abdullah SNH. The variants of the Bees Algorithm (BA): a survey. Artif Intell Rev. 2017;47(1):67–121. doi: 10.1007/s10462-016-9476-8.
[46] Karaboga D. An idea based on honey bee swarm for numerical optimization. Tech Report-TR06, Computer Engineering Department, Engineering Faculty, Erciyes University; 2005. p. 10.
[47] Yang J, Cui J, Xia X, Gao X, Yang B, Zhang Y-D. An artificial bee colony algorithm with an adaptive search strategy selection mechanism and its application on workload prediction. Comput Ind Eng. Mar. 2024;189:109982. doi: 10.1016/J.CIE.2024.109982.
[48] John GH, Kohavi R, Pfleger K. Irrelevant features and the subset selection problem. Proc. 11th Int. Conf. Mach. Learn. ICML 1994; 1994. p. 121–9. doi: 10.1016/B978-1-55860-335-6.50023-4.
[49] Lavin A, Ahmad S. Evaluating real-time anomaly detection algorithms - The Numenta anomaly benchmark. Proc. - 2015 IEEE 14th Int. Conf. Mach. Learn. Appl. ICMLA 2015; Mar. 2016. p. 38–44. doi: 10.1109/ICMLA.2015.141.
[50] Laptev S, Amizadeh N. A labeled anomaly detection dataset S5 Yahoo Research, v1. [Online]. Available: https://webscope.sandbox.yahoo.com/.
[51] Qiu J, Du Q, Qian C. KPI-TSAD: A time-series anomaly detector for KPI monitoring in cloud applications. Symmetry. 2019;11:1–20. doi: 10.3390/sym11111350.
[52] El-Abd M. Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf Sci (Ny). 2012;182(1):243–63. doi: 10.1016/j.ins.2011.09.005.

[53] Kasabov N, Feigin V, Hou ZG, Chen Y, Liang L, Krishnamurthi R, et al. Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. Neurocomputing. Jun. 2014;134:269–79. doi: 10.1016/j.neucom.2013.09.049.
[54] Tu E, Kasabov N, Yang J. Mapping temporal variables into the neucube for improved pattern recognition, predictive modeling, and understanding of stream data. IEEE Trans Neural Netw Learn Syst. 2017;28(6):1305–17. doi: 10.1109/TNNLS.2016.2536742.
[55] Tharwat A. Classification assessment methods. Appl Comput Inform. 2018;17(1):168–92. doi: 10.1016/j.aci.2018.08.003.
[56] Zhang L, Zhao J, Li W. Online and unsupervised anomaly detection for streaming data using an array of sliding windows and PDDs. IEEE Trans Cybern. 2019;51(4):2284–9. doi: 10.1109/TCYB.2019.2935066.
[57] Xu L, Zheng L, Li W, Chen Z, Song W, Deng Y, et al. NVAE-GAN Based Approach for Unsupervised Time Series Anomaly Detection. ArXiv, Prepr; 2021. doi: 10.48550/arXiv.2101.02908.
[58] Amarbayasgalan T, Pham VH, Theera-Umpon N, Ryu KH. Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error. Symmetry. 2020;12:1251. doi: 10.3390/sym12081251.
[59] Dimoudis D, Vafeiadis T, Nizamis A, Ioannidis D, Tzovaras D. Utilizing an adaptive window rolling median methodology for time series anomaly detection. Procedia Comput Sci. 2022;217:584–93. doi: 10.1016/j.procs.2022.12.254.
[60] Sina D, Thomas B. Large anomaly detection in univariate time series - an empirical comparison of machine learning algorithms. 19th Ind. Conf. Data Min. ICDM 2019. Icdm; 2019. p. 1–15. https://www.honda-ri.de/pubs/pdf/4015.pdf.
[61] Jensen L. Minimalism in Deep Learning [PhD dissertation]. Boston University, 2021. Available from: https://hdl.handle.net/2144/43932.