JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4281-4308
https://publishoa.com
ISSN: 1309-3452

Convolution Neural Network based Overhead Reduced Intrusion Detection System

[1]Harihara Krishnan. R, [2]Dr. Ananthi Sheshasaayee
[1]Ph.D. Research Scholar, PG & Research Department of Computer Science,
Quaid-E-Millath Government College for Women (Autonomous), Chennai – 600 002

[2]Associate Professor & Head, PG & Research Department of Computer Science,
Quaid-E-Millath Government College for Women (Autonomous), Chennai – 600 002

Received: 2022 March 15; **Revised:** 2022 April 20; **Accepted:** 2022 May 10

ABSTRACT

Attacks in wireless sensor networks (WSNs) aim to prevent or eradicate the network's ability to perform its anticipated functions. Intrusion detection is a defense used in wireless sensor networks that can detect unknown attacks. Due to the incredible development in computer-related applications and massive Internet usage, it is indispensable to provide host and network security. The development of hacking technology tries to compromise computer security through intrusion. Intrusion detection system (IDS) was employed with the help of machine learning (ML) Algorithms to detect intrusions in the network. In recent work Pattern Matching aware Replicated Neural Network based Intrusion Detection System (PM-RNN-IDS) is introduced for the accurate and faster IDS rate. In this work, Data pre-processing is performed based on Firefly algorithm to eliminate redundant record set and enhanced KNN based imputation is utilized to handle missing value. Feature selection using Modified Particle Swarm Optimization. Finally, Intrusion detection is carried out using Replicator Neural Networks.used. However, dataset which is used in this work has more dimensions and it consumes more storage space. Feature reduction will solve this problem and it is not focused in recent work. Also, Replicator Neural Networks does not produce sufficient accuracy results. To avoid these problems in this work Convolution Neural Network based Overhead Reduced Intrusion Detection System (CNN-OR-IDS) is proposed for the optimal and accurate intrusion detection rate. In this work, modified Firefly algorithm is introduced for redundant data detection and Bagging based KNN Imputation technique is used for the Missing value imputation process.

Once the dataset is preprocessed, feature reduction is performed by introducing the method namely Gradually Feature Removal (GFR) method. Once the irrelevant features are removed from the dataset, optimal feature selection is done by using Hybrid of Cuckoo Search with fish swarm algorithm. Finally, intrusion detection process is carried out using Convolutional Neural Network algorithm. Experimental results demonstrate the effectiveness of the proposed work in terms of false positive rate and false alarm rate.

Keywords: Pattern Matching, Missing value, Attacks, feature reduction and intrusion detection.

1. INTRODUCTION

Wireless Sensor Networks is regarded as one of the prominent research topics. The technology is an ideal solution for numerous applications in various fields like telecommunication, military, healthcare, research, and agriculture, amongst others. The application of wireless sensor networks in detecting natural disasters such as earthquakes, flooding, or volcanoes. The widespread WSNs usage has introduced many security threats in the implementation and deployment phase [1,2].

Wireless sensor networks are susceptible to different attacks due to unique constraints like storage capacity, restricted processing power, and battery power capacity. People worldwide rely on networking systems to bring new ideas and answers to their issues and help them meet their basic requirements [3]. New and most often used technological innovations include sensors that allow users to receive remote data and utilize it for their specific purpose [4,5].

Despite the advantages of WSN, several security loopholes can be exploited to receive attacks. While using WSN applications, users can face several types of security threats that can cause data breaches [6]. Researchers have been attempting to develop new security solutions in order to prevent attacks from succeeding in their endeavors. Several technological advancements have helped develop novel approaches to infiltrate and prevent such attacks. Still, deep learning has brought about the most effective approaches for preventing such security risks and attacks [7,8].

In recent work Pattern Matching aware Replicated Neural Network based Intrusion Detection System (PM-RNN-IDS) is introduced for the accurate and faster IDS rate. In this work, Data pre-processing is performed based on Firefly algorithm to eliminate redundant record set and enhanced KNN based imputation is utilized to handle missing value. Feature selection using Modified Particle Swarm Optimization. Finally Intrusion detection is carried out using Replicator Neural Networks.used. However dataset which is used in this work has more dimensions and it consumes more storage space. Feature reduction will solve this problem and it is not focused in recent work. Also Replicator Neural Networks does not produces sufficient accuracy results.

To avoid these problems in this work Convolution Neural Network based Overhead Reduced Intrusion Detection System (CNN-OR-IDS) is proposed for the optimal and accurate intrusion detection rate. In this work, modified Firefly algorithm is introduced for redundant data detection and Bagging based KNN Imputation technique is used for the Missing value imputation process. Once the dataset is preprocessed, feature reduction is performed by introducing the method namely Gradually Feature Removal (GFR) method. Once the irrelevant features are removed from the dataset, optimal feature selection is done by using Hybrid of Cuckoo Search with fish swarm algorithm. Finally intrusion detection process is carried out using Convolutional Neural Network algorithm.

2. LITERATURE REVIEW

Riecker et al (2015) [9] proposed a lightweight, energy-efficient system, which makes use of mobile agents to detect intrusions based on the energy consumption of the sensor nodes as a metric. A linear regression model is applied to predict the energy consumption. Simulation results indicate that denial-of-service attacks, such as flooding, can be detected with high accuracy, while keeping the number of false-positives very low.

Maleh et al (2015) [10] proposed a hybrid, lightweight intrusion detection system for sensor networks. Our intrusion detection model takes advantage of cluster-based architecture to reduce energy consumption. This model uses anomaly detection based on support vector machine (SVM) algorithm and a set of signature rules to detect malicious behaviors and provide global lightweight IDS. Simulation results show that the proposed model can detect abnormal events efficiently and has a high detection rate with lower false alarm.

Hu et al (2016) [11] performed intrusion detection based on Kernel Fisher Discriminant and SVM. According to the principle that the classifiers' sensitivity is different when different types of data is processed, the data is assigned to Kernel Fisher Discriminant and SVM. So that Data can be processed by the corresponding optimal classifier, and detection efficiency can be raised. Theoretical analysis and simulation results show that the proposed schemes not only can detect intrusions effectively, but also lower energy consumption than others.

Otoum et al (2019) [12] presented a comprehensive analysis of the use of machine and deep learning (DL) solutions for IDS systems in wireless sensor networks (WSNs). To accomplish this, we introduce restricted Boltzmann machine-based clustered IDS (RBC-IDS), a potential DL-based IDS methodology for monitoring critical infrastructures by WSNs. We study the performance of RBC-IDS, and compare it to the previously proposed adaptive machine learning-based IDS: the adaptively supervised and clustered hybrid IDS (ASCH-IDS).

Almomani et al (2016) [13] considered the use of LEACH protocol which is one of the most popular hierarchical routing protocols in WSNs. A scheme has been defined to collect data from Network Simulator 2 (NS-2) and then processed to produce 23 features. The collected dataset is called WSN-DS. Artificial Neural Network (ANN) has been trained on the dataset to detect and classify different DoS attacks. The results show that WSN-DS improved the ability of IDS to achieve higher classification accuracy rate. WEKA toolbox was used with holdout and 10-Fold Cross Validation methods.

Amaran and Mohan (2021) [14] presented a new optimal Support Vector Machine (OSVM) based IDS in WSN. The presented OSVM model involves the proficient selection of optimal kernels in the SVM model using whale optimization algorithm (WOA) for intrusion detection. Since the SVM kernel gets altered using

WOA, the application of OSVM model can be used for the detection of intrusions with proficient results. The performance of the OSVM model has been investigated on the benchmark NSL KDDCup 99 dataset. The resultant simulation values portrayed the effectual results of the OSVM model by obtaining a superior accuracy of 94.09% and detection rate of 95.02%.

Safaldin et al (2021) [15] proposed an enhanced intrusion detection system (IDS) by using the modified binary grey wolf optimizer with support vector machine (GWOSVM-IDS). The GWOSVM-IDS used 3 wolves, 5 wolves and 7 wolves to find the best number of wolves. The proposed method aims to increase intrusion detection accuracy and detection rate and reduce processing time in the WSN environment through decrease false alarms rates, and the number of features resulted from the IDSs in the WSN environment. Indeed, the NSL KDD'99 dataset is used to demonstrate the performance of the proposed method and compare it with other existing methods. The proposed methods are evaluated in terms of accuracy, the number of features, execution time, false alarm rate, and detection rate. The results showed that the proposed GWOSVM-IDS with seven wolves overwhelm the other proposed and comparative algorithms.

3. PROPOSED METHODOLOGY

This section discuses the proposed Convolution Neural Network based Overhead Reduced Intrusion Detection System (CNN-OR-IDS).Which consist of six phases first one is redundant data detection using modified Firefly algorithm, Second one is KNN based Missing value imputation, thirdly feature reduction is performed by introducing the method namely Gradually Feature Removal (GFR) method, fourth one is feature selection using Hybrid of Cuckoo Search with fish swarm algorithm, fifth one is intrusion detection process is carried out using Convolutional Neural Network. Overall architecture of the proposed model is shown in figure 1.

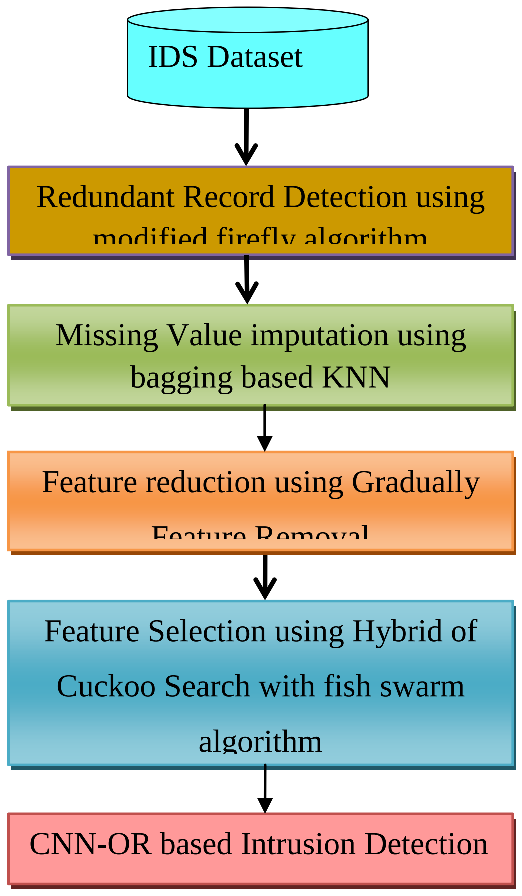
IDS Dataset
Redundant Record Detection using modified firefly algorithm
Missing Value imputation using bagging based KNN
Feature reduction using Gradually Feature Removal
Feature Selection using Hybrid of Cuckoo Search with fish swarm algorithm
CNN-OR based Intrusion Detection

Figure 1.Overall architecture of the proposed system

3.1. DATA PRE-PROCESSING

Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results. In this work, modified Firefly algorithm is utilized for redundant data detection and Bagging based KNN Imputation technique is utilized for the Missing value replacement process.

3.1.1. MODIFIED FIREFLY BASED REDUNDANT DATA DETECTION

Firefly algorithm is one of the new metaheuristic algorithms for optimization problems. The algorithm is inspired by the flashing behavior of fireflies. In the algorithm, randomly generated solutions will be considered as fireflies, and brightness is assigned depending on their performance on the objective function. One of the rules used to construct the algorithm is, a firefly will be attracted to a brighter firefly, and if there is no brighter firefly, it will move randomly. Advantages of this algorithm were: “Easy implementation, Easy to understand”. Disadvantages of this algorithm were, “Gets trapped into several local optima, No Memorizing Capability”[16].

This algorithm eliminates the weaknesses of the firefly algorithm by enhancing its exploitation and exploration ability by reducing the randomness of the algorithm and improving the collective movement of the fireflies. In the SFA, the fireflies move regardless of the global optima which decreases the ability of the firefly algorithm to find global best. Therefore in the MoFA, while comparing the brightness of two fireflies the global optimum affects the movement of fireflies. The firefly with either the maximum or minimum value is allowed to influence other fireflies leading to better results iteratively. Also this algorithm enhances the exploitation quality of the algorithm by gradually reducing the randomness and by adding a social dimension to each firefly (the global best). This increases the global exploration chances of fireflies. In the MoFA implementation, randomization parameter # was not kept fixed and was linearly decreased from #0 to #$ with iterations, where #0 was the initial value and #$ was the final value. This strategy could keep balance between the exploration and exploitation abilities of the proposed algorithm. In the early stage, larger # provided better global searching ability and in the later stage, smaller # offered better convergence. The distance function (ri) was determined by the equation (1)

$$r_{i,best} = \sqrt{(x_i - x_{gbest})^2 + (y_j - y_{best})^2} \quad (1)$$

The movement of ith firefly is determined by equation (2)

$$x_i = x_i + \left(\beta_0 e^{-\gamma r_{i,j}^2}(x_j - x_i) + \beta_0 e^{-\gamma r_{i,best}^2}(x_{gbest} - x_i)\right) \quad (2)$$

Where

$$\varepsilon = rand^{=1/2} \quad (3)$$

Here, the ith firefly got attracted to the best solution if no local best solution existed in the neighbourhood. Also, the current global best (gbest) is explicitly used redefine the distance and movement of fireflies along with decoding the final best solutions. Thus, the proposed algorithm reduces the randomness of the algorithm to obtain convergence quickly and influences the movement of fireflies towards global optima for reducing the probability of the algorithm in getting trapped into several local optima.

In this work modified this random movement of the brighter firefly by generating random directions in order to determine the best direction in which the brightness increases. If such a direction is not generated, it will remain in its current position. Furthermore the assignment of attractiveness is modified in such a way that the effect of the objective function is magnified. This algorithm is resultant with the effective and optimal detection of redundant data from the input dataset.

Algorithm:

Inputs: objective function f(x); variable boundary; population size n; maximum attractiveness β_0; absorption coefficient γ; randomization parameter α

Outputs: best solution

```
1. Initialization;
2. For variable i = 1:n
3. Randomly produce xi within the variable ranges;
4. End for i;
5. Evaluate the function values of the firefly population;
6. Do while (Termination Criterion Are Not Met)
7. For i = 1:n
8. For variable j = 1:n
9. If f (xj) < f(xi)
10. If (Ij > Ii), move firefly i towards j;
11. End if
12. Evaluate new solutions and update light intensity;
13. End for j;
14. End for i;
15. Evaluate the new firefly populations;
16. Record the best solution achieved so far;
17. End while;
```

3.1.2. MISSING VALUE REPLACEMENT USING BAGGING BASED KNN IMPUTATION

Once the redundant data is found it will be eliminated from the dataset, and then resultant data will be given as input to the enhanced KNN based imputation method. The use of a KNN model to predict or fill missing values is referred to as "Nearest Neighbor Imputation" or "KNN imputation." [17] Configuration of KNN imputation often involves selecting the distance measure (e.g. Euclidean) and the number of contributing neighbors for each prediction, the k hyperparameter of the KNN algorithm. The resultant data from KNN imputation algorithm will be the dataset with complete information.

The expansion of Bagging is "bootstrap aggregating" it a unique method for integration decision tree or other classifiers. It makes the base learning algorithm to run iteratively in successive rounds. During each round with the help of bootstrap replicate the base learner gets trained from the original training set. If the training set consists of n examples. Then the bootstrap replicate is a new training set that also consists of n examples, and which is formed by repeatedly selecting uniformly at random and with replacement of n examples from the original training set. It means that the same example may appear multiple times in the bootstrap replicate, or it may appear not at all. Thus, on each of M rounds of bagging, a bootstrap replicate is created from the original training set. A base classifier is then trained on this replicate, and the process continues. After M rounds, a final combined classifier is formed which simply predicts with the majority choice of all of the base classifiers. In the k-Nearest Neighbour based imputation an attribute-att with missing value is imputed by finding its k-Nearest Neighbour and assigning its value to the attribute att[18].

```
Input: Unique Dataset of KDDCUP obtained using the resultant of Peudocode 1.
Output: Complete Dataset
27.  Given training data (x1, y1), . . . , (xn, yn)
28. For m=1, . . . , M
29. Form bootstrap replicate dataset BSt by selecting n random examples from the training set with replacement
30. let km be the result of k-NN algorithm on BSt
31.  output combined classifier:
32. K(x) = majority(k1(x), . . . , kM(x))
```

The dataset for preprocessing is collected from KDD Cup'99 dataset. Before performing any attack detection, the duplication in dataset is removed using weighted minkowski based firefly algorithm and in the second stage the presence of missing value is handled using three techniques namely replace using mean value, replace using k-NN imputation and replace using proposed bagging k-NN. The performance of each technique is validated using Rule Induction Classifier. From the result obtained the complete dataset generated by proposed bagging k-NN shows a better result than the two former techniques and the complete dataset will be used for feature extraction.

In the proposed work the firefly algorithm a variant of the Minkowski function, the weighted Minkowski distance function, has also been applied to measure similarity in the dataset. The basic idea is to introduce weighting to identify important features. Assigning each feature, a weighting coefficient wi (i = 1. . . p), the weighted Minkowski distance function is defined in equation 4.

$$r = \left(\sum_{i=1}^{p} w_i |x_i - y_i|^n\right)^{1/n} \quad (4)$$

By applying a static weighting vector for measuring similarity, the weighted Minkowski distance function assumes that similar records will resemble same record in the same features. The original data set is first portioned in to groups. The records having missing values in their attributes are in one set and the records without any missing values are placed in a separate group. The k-NN classifier is trained with the complete data sets, and afterward the imperfect data is agreed to the bagging model for predicting the missing feature values. The scheme is recurrent for the whole set of attributes that have missing values. At the last part of training, this training dataset and absent value imputed datasets are combined to formulate the absolute data. The concluding dataset is then fed to the chosen k-NN classifier for classification.

3.2. FEATURE REDUCTION PROCESS USING GRADUALLY FEATURE REMOVAL

(GFR) METHOD

Feature Reduction also known as Dimensionality Reduction helps us in reducing the processing time allowing us to perform more complex algorithms. Another benefit of having a low number of features is that it frees storage space. These benefits, however, are all hardware related. The most important benefit of Feature Reduction is that it takes care of the problem of multicollinearity. Thus, Feature Reduction helps in making the process of data analysis faster and more accurate. The techniques of feature reduction can be divided into two types- Feature Selection and Feature Extraction. Under Feature Selection we check the worthiness of a feature by using different techniques and once the features that are less useful to our model are identified, they can be dropped from the dataset. This also acts as the basic difference between Feature Selection and the next type of Feature Reduction- Feature Extraction.

Once the dataset is preprocessed, feature reduction is performed by introducing the method namely Gradually Feature Removal (GFR) method. Direct observation shows that the idea of the sole feature method is based on the reverse idea of the feature removal method. From the point view of statistics, the intersection of the features in above both methods is more reliable than mere one method. Henceforth, a hybrid method is formed by using the same features with high importance order both in the feature removal method and the sole feature method. This method will reduce the number of features from the input dataset.

Each network visit behaviour in KDD99 data-base maps to a mathematical vector with 41 features. For the sake of efficiency, streamline the mathematical vector is vital for Machine Learning method in IDS.

Algorithm (Gradually feature removal method, GFR method).

Step 1: Let N = 41, j = 1, where N denote as the dimension of the feature scale, j is used to count the chosen critical feature.

Step 2: Assume X =(x1, x2, ..., xN), stands for the current feature of the sample data.

Step 3: Delete xi(i =1,2,..., N) from X and update $X^{(1,i)}$=(x1, x2, ..., xi1, xi+1, ..., x41) as the new feature vector.

Step 4: A classifier is undertaken to evaluate the effective of the features combination, $X^{(1)}$, $X^{(2)}$,…..,$X^{(N).}$ Record the ith $x_i$with the best performance of$X^{(j)}$.x_j= x_i.

Step 5: Delete xi from X =(x1, x2, ..., xN), N--, j + +. Go to step 2.

Step 6: Repeat step 2 to step 5, until a series ofx_j is obtained.

Step 7: Evaluate the accuracy and efficiency performance of classifier with feature subset of $\overline{x_j}$. Choose the most balancedone.

3.3. FEATURE SELECTION USING HYBRID CUCKOO SEARCH

Once the irrelevant features are removed from the dataset, optimal feature selection is done by using Hybrid Cuckoo Search algorithm. In this work cuckoo search algorithm is hybridized with the fish swarm algorithm for resulting with the optimal set of features. Those optimally selected features will be given as input to the classifier for ensuring the increased accuracy rate. Cuckoo Search algorithm (CS) is swarm intelligence-based algorithm motivated by nature. This algorithm is based on brood parasitism of some cuckoo species and has high capability of global search. Therefore, the global optimum can be figured out with higher probability. The fish swarm algorithm (FSA) is a new population-based/swarm intelligent evolutionary computation technique that was inspired by the natural schooling behaviour of fish. FSA presents a strong ability to avoid local minimums in order to achieve global optimization. These two algorithms are hybridized together to avoid the local convergence issue.

3.3.1. CUCKOO SEARCH ALGORITHM

Cuckoo search is an optimization algorithm developed by Xin-She Yang and Suash Deb in 2009. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of host birds of other species. Some host birds can engage direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in colors and pattern of the eggs of a few chosen host species. Cuckoo search idealized such breeding behavior, and thus can be applied for various optimization problems. It has been shown that cuckoo search is a special case of the well-known ($\mu + \lambda$)-evolution strategy[19].

Cuckoo search (CS) uses the following representations:

Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. In the simplest form, each nest has one egg. The algorithm can be extended to more complicated cases in which each nest has multiple eggs representing a set of solutions. CS is based on three idealized rules:

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest;
- The best nests with high quality of eggs will carry over to the next generation;
- The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability. In this case, the host bird can throw the egg away/abandon the nest, and build a completely new nest.

In addition, Yang and Deb discovered that the random-walk style search is better performed by Lévy flights rather than simple random walk.

The pseudo-code can be summarized as:

```
Objective function: $f(x)$, $x=(x_1, x_2,…,x_d)$
 Generate an initial population of n host nests;
While (t<MaxGeneration) or (stop criterion)
  Get a cuckoo randomly (say, i) and replace its solution by performing Lévy flights;
  Evaluate its quality/fitness $F_1$
       [For maximization, $F_i \propto f(x_i)$)
  Choose a nest among n (say, j) randomly;
  if ($F_i > F_j$),
      Replace j by the new solution;
  end if
  A fraction (pa) of the worse nests is abandoned and new ones are built;
  Keep the best solutions/nests;
  Rank the solutions/nests and find the current best;
  Pass the current best solutions to the next generation;
end while
```

3.3.2. FISH SWARM ALGORITHM

Fish swarm algorithm, first proposed in 2002, is a new population-based optimization technique inspired by the natural feeding behavior of fish. A fish is represented by its D-dimensional position X_i

= (x_1, x_2, . . ., x_k, . . ., x_D), and food satisfaction for the fish is represented as FSi. This paper targets FS minimization. The relationship between two fish is denoted by their Euclidean distance $d_{ij} = \|X_i - X_j\|$. Another parameters include: visual (representing the visual distances of fish), step (maximum step length), and ı (a crowd factor). n is used to represent the size of the fish population. All fish try to identify locations able to satisfy their food needs using three distinct behaviours[20,21]. These include:

(1) Searching behavior: Searching is a basic biological behavior adopted by fish looking for food. It is based on a random search, with a tendency toward food concentration. It is expressed mathematically as:

$$\vec{x}_{i+1}^{k} = \vec{x}_i^{k} + R(S1)\,\frac{\vec{x}_j^{k} - \vec{x}_i^{k}}{\|x_j - x_i\|},\ FS_j < FS_i \quad (5)$$

$$\vec{x}_{i+1}^{k} = \vec{x}_i^{k} + R(S2)\vec{1} \quad (6)$$

where x_{ik} represents the kth element of fish position X_i. We randomly select for fish X_i a new position X_j within its visual. If the corresponding j is satisfied, Eq. (5) is then employed at the next position X_{i+1}. If FS_j is not satisfied after try number trials, a random position within the step range will be directly adopted as Eq. (6). In the above equations, R(S1) and R(S2) represent random variables within [0, step] and [−step, step], respectively.

(2) Swarming behavior Fish assemble in several swarms to minimize danger. Objectives common to all swarms include satisfying food intake needs, entertaining swarm members and attracting new swarm members. Mathematically,

$$\vec{x}_{i+1}^{k} = \vec{x}_i^{k} + R(S1)\,\frac{\vec{x}_C^{k} - \vec{x}_i^{k}}{\|x_C - x_i\|},\ FS_j < FS_i \text{ and}$$

$$\left(\frac{n_s}{n}\right) < \delta \quad (7)$$

A fish located at Xi has neighbors within its visual. Xc identifies the center position of those neighbors and is used to describe the attributes of the entire neighboring swarm. If the swarm center has a greater concentration of food than is available at the fish's current position Xi (i.e., FSc < FSi), and if the swarm (Xc) is not overly crowded (ns/n < ı), the fish will move from Xi to next Xi+1, toward Xc. Here, ns represents number of individuals within the Xc's visual. Swarming behavior is executed for a fish based on its associated Xc; otherwise, searching behavior guarantees a next position for the fish.

(3) Following behavior When a fish locates food, neighboring individuals follow. Mathematically,

$$\vec{x}_{i+1}^{k} = \vec{x}_{i}^{k} + R(S1)\,\frac{\vec{x}_{min}^{k} - \vec{x}_{i}^{k}}{\|x_{min} - x_i\|},\ FS_j < FS_i \text{ and } \left(\frac{n_f}{n}\right) < \delta \quad (8)$$

Within a fish's visual, certain fish will be perceived as finding a greater amount of food than others, and this fish will naturally try to follow the best one (Xmin) in order to increase satisfaction (i.e., gain relatively more food [FSmin < FSi] and less crowding [nf/n <δ]). nf represents number of fish within the visual of Xmin. Searching behavior commences if following behavior is unable to determine a fish's next position. Besides, FSA should provide a bulletin that records the optimal state and current performance of fish during iterations.

3.3.3. HYBRIDIZED ALGORITHM

The FSA visual provides local search attributes. A small visual restricts a fish to interaction with a relatively small number of companions. The FSA step limits maximum step length, with a small step limiting fish to searching a small area and increasing the risk of wasting time. step values are set based on Euclidean distance calculations and are sensitive to FSA performance. As settings are difficult, this work employs CS formulation to minimize the impact of the step factor. As a result, artificial fish are able to swim like a cuckoos in CS, subject to the visual factor, but not the step. All the original FSA equations have been modified:

1) Searching behavior:

$$\vec{x}_{i+1}^{k} = \vec{x}_{i}^{k} + \varphi_{3k}(\vec{x}_{j}^{k} - \vec{x}_{i}^{k}),\ \ FS_j < FS_i \quad (9)$$

$$\vec{x}_{i+1}^{k} = \vec{x}_{i}^{k} + R(V1)\vec{1} \quad (10)$$

$$\varphi_3^k = c_3 r_3^k \quad (11)$$

Where φ_3^k is a uniform random number within [0, 2] with a mean value of one. c3 is 2, and a uniform random number within [0, 1]. Xj is still a new position within Xi's visual (same as FSA). Therefore, Eq. (9) uses the CS formulae to releases step settings in Eq. (5). Since Eq. (9) is free to step, this paper further modified Eq. (6) as Eq. (10), providing such with a visual range. R(V1) is a random variable within [−visual, visual]. When step is smaller than visual in FSA, the modified FSA allows fish to swim for greater lengths than permitted by the original FSA. Modified searching behavior is totally free to step.

2) Swarming behavior:

$$\vec{x}_{i+1}^{k} = \vec{x}_{i}^{k} + \varphi_4^k \left(\vec{x}_{C}^{k} - \vec{x}_{i}^{k}\right), FS_j < FS_i \text{ and } \left(\frac{n_s}{n}\right) < \delta \quad (12)$$

Where the definition of φ_4^k is the same as φ_3^k and definition of Xc is same as that in FSA. Modified swarming behavior is therefore free to step.

3) Following behavior:

$$\vec{x}_{i+1}^{k} = \vec{x}_{i}^{k} + \varphi_5^k \left(\vec{x}_{min}^{k} - \vec{x}_{i}^{k}\right), FS_{min} < FS_i \text{ and } \left(\frac{n_f}{n}\right) < \delta \quad (13)$$

where the definition of φ_5^k is the same as φ_3^k and the definition of Xmin is same as that used in FSA. Modified following behavior is therefore free to step. This modified FSA, named “MFSA”, is free of the step parameter of the original FSA. All positions of Xj, Xc, Xmin, and the next position Xj are dependent upon visual only. The major advantage of this is to release a step parameter. Certainly, the execution mechanism still follows the same process for FSA.

3.4. INTRUSION DETECTION USING CONVOLUTIONAL NEURAL NETWORK ALGORITHM

In this work, optimal and accurate detection of intrusion activities is ensured by adapting the convolutional neural network algorithm. The one-dimensional convolution neural network (1D-CNN) method can not only diagnose bearing faults accurately, but also overcome shortcomings of the traditional methods. Different from machine learning and other deep learning models, the 1D-CNN method does not need pre-processing one-dimensional data of rolling bearing’s vibration [22,23].

CNN is usually composed of two parts. In part 1, convolution and pooling operations are alternatively used to generate deep features of the raw data. And in part 2, the features are connected to an MLP for classification. Here are some details for each layer:

(i) Input layer. Input layer has N × k neurons, where k denotes the variate number of input time series and N denotes the length of each univariate series

(ii) Convolutional layer. Perform convolution operations on the time series of preceding layer with convolution filters. Here are some filter parameters to be determined previously according to domain knowledge or just depending on experiments, such as, filter numbers m, convolution stride s and the size of filter k × l, where k denotes the variate number of the time series in the preceding layer and l denotes the length of filter. A nonlinear transformation function f also needs to be determined in this layer. For instance, if the preceding layer contains k-variate time series and the length of each univariate is N, after the Convolutional operation, we get m-variate time series and the length of each univariate is $\left\lfloor \frac{N-1}{s} + 1 \right\rfloor$, where $\lfloor . \rfloor$ denotes rounding down.

(iii) Auto encoder Layer: An autoencoder is a type of artificial neural network used to learn efficient data coding in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Several variants exist to the basic model, with the aim of forcing the learned representations of the input to assume useful properties. he simplest form of an autoencoder is a feedforward, non-recurrent neural network similar to single layer perceptrons that participate in multilayer perceptrons (MLP) – having an input layer, an output layer and one or more hidden layers connecting them – where the output layer has the same number of nodes (neurons) as the input layer, and with the purpose of reconstructing its inputs (minimizing the difference between the input and the output) instead of predicting the target value Y given inputs X. Therefore, autoencoders are unsupervised learning models (do not require labeled inputs to enable learning). An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions Φ and ψ such that:

$$\Phi : X \rightarrow F \quad (14)$$

$$\Psi : F \rightarrow X \quad (15)$$

$$\Phi, \Psi = \underset{\Phi, \Psi}{\arg\min} \|X - (\Psi \circ \Phi)X\|^2$$

(16)

In the simplest case, given one hidden layer, the encoder stage of an autoencoder takes the input $xx \in R^d = X$ and maps it to $h \in R^p = F$:

$$h = \sigma(Wx + b) \quad (17)$$

(iv) Softmax layer: A Softmax function is a type of squashing function. Squashing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability. Similarly, softmax functions are multi-class sigmoids, meaning they are used in determining probability of multiple classes at once. Since the outputs of a softmax function can be interpreted as a probability (i.e.they must sum to 1), a softmax layer is typically the final layer used in neural network functions. It is important to note that a softmax layer must have the same number of nodes as the output later. A neural network may be attempting to determine if there is a dog in an image. It may be able to produce a probability that a dog is, or is not, in the image, but it would do so individually, for each input. A softmax layer, allows the neural network to run a multi-class function. In short, the neural network will now be able to determine the probability that the dog is in the image, as well as the probability that additional objects are included as well. Softmax layers are great at determining multi-class probabilities, however there are limits. Softmax can become costly as the number of classes grows. In those situations, candidate sampling can be an effective workaround. With candidate sampling, a softmax layer will limit the scope of its calculations to a specific set of classes. For example, when determining if an image of a bowl of fruit has apples, the probability does not need to be calculated for every type of fruit, just the apples. Additionally, a softmax layer assumes that there is only one member per class, and in situations where an object belongs to multiple classes, a softmax layer will not work. In that case, the alternative is to use multiple logistic regressions instead.

(v) Output layer. The output layer has n neurons, corresponding to n classes of time series. It is fully connected to the feature layer. The most popular method is taking the maximum output neuron as the class label of the input emotion in classification task.

Training of auto encoder CNN

The CNN is trained via a sequence of training examples $((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N))$ with

$x_t \in R^{N\times k}$, $y_t \in R^n$ for $1 \le t \le N$. The high-order features x_t is given as input to the network, while the vector yt denotes the target output. The network is trained according to the following several steps:

Step 1 Initialize the network. Determine the CNN architecture, composed of two Convolutional layers and two pooling layers, as shown in Fig. 1. Fix the neuron number of input layer and output layer according to the classification task. Set all the CNN parameters. Initialize the weights and bias with a small random number. Select a learning rate η, and an activation function f, and the commonly used example is the sigmoid function:

$$f(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}} \quad (18)$$

Step 2 Choose a training sample from the training set randomly.

Step 3 Calculate the output of each layer.

The output of the Convolutional layer can be written as

$$C_r(t) = f(\textstyle\sum_{i=1}^{l} \sum_{j=1}^{k} x(i+s \quad (19)$$

Where $x \in R^{N\times k}$ denotes the input higher order semantic features or the output of the preceding layer, s denotes the convolution stride, $C_r(t) =$ refers to the tth component of the rth feature map, ωr ∈ Rl×k and b(r) refer to the weights and bias of the rth convolution filter.

The output of the pooling layer can be written as

$$P_r(t) = fg(C_r((t-1)l+1), C_r((t-1)l \quad (20)$$

Where the function g represents the pooling strategy, the most popular used is averaging or max pooling. It is obvious that pooling operation achieves reducing the point data, while not changing the number of feature maps.

The output of the output layer can be written as

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4281-4308
https://publishoa.com
ISSN: 1309-3452

$$O(j) = f(\sum_{i=1}^{M} z(i)\omega_f(i,j$$

(21)

Where z denotes the final feature map in the feature layer, b_f is the bias of the output layer and $\omega_f \in R^{M\times n}$ refers to the connection weights between the feature layer and the output layer.

So the mean-square error can be written as

$$E = \frac{1}{2}\sum_{k=1}^{n} e(k)^2 = \frac{1}{2}\sum_{k=}^{n}$$

(22)

Step 4 Update the weights and bias by the gradient descent method.

$$p = p - \eta\frac{\partial E}{\partial p}$$

(23)

Where p is the value of the parameter, and p refers to ω_r, ω_f ,b, or b_f in this CNN.

Step 5 Choose another training sample and go to Step 3 until all the samples in the training set have been trained.

Step 6 Increase the iteration number. If the iteration number is equal to the maximum value which is set previously, terminate the algorithm. Otherwise, go to Step 2. Based on the above steps the intrusion is classified.

4. RESULT AND DISCUSSION

In this section, the NS-2 simulator is used to evaluate the performance of the proposed HNWT. This simulation model network consisting of 100 nodes placed randomly within a 100 × 100 meters area. The two types of nodes in the simulations are defined as: well-behaved nodes and malicious nodes. The malicious nodes can launch DOS flooding attacks in the simulated scenarios. The proposed system performance is evaluated by comparing the proposed system CNN-OR-IDSwith the previous systems namely Tree-CNN, PM-RNN-IDS. The parameters used in this research for evaluating the trust system are given in the Table 1. The performance of CNN-OR-IDS model was evaluated using the following metrics such as false alarm rate, false positive rate, success rate, and DoS flooding attack probability.

Table 1. Simulation parameters

Simulation Parameters	Values
Channel	Wireless Channel
Mac	802.11
Antenna Type	Omni antenna
Routing Protocol	AODV
Initial Energy	100 joules
Traffic type	CBR
Agent	UDP
Simulation area	100X100 meters
Number of nodes	100

4.1. FALSE ALARM RATE

A false alarm ratio, generally abbreviated FAR, is the number of false alarms per the total number of warnings or alarms in a Sybil attack detection. In the following figure false alarm rate comparison is shown against different number of attacker nodes presence in the environment.

Table 2. False Alarm Rate Values

Percentage of Attacker Nodes	False Alarm Rate		
	Tree-CNN	PM-RNN-IDS	CNN-OR-IDS
10	78	71	68
20	79	73	69
30	83	77	71
40	85	79	73
50	87	83	75

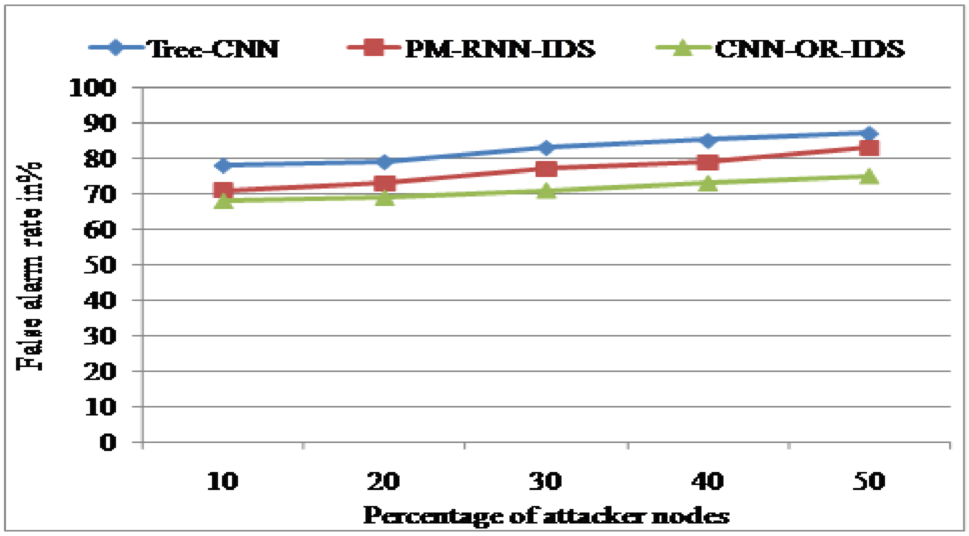

Figure 2. False alarm rate vs number of attacker nodes

In figure 2, comparison evaluation of the false alarm rate for the proposed and existing methodologies are given. From this comparison it can be proved that the proposed method CNN-OR-IDS tends to have better performance than the previous methodologies with lesser wrong detection of attacker nodes.

4.2. FALSE POSITIVE RATE

The false positive rate is calculated as the ratio between the number of negative events wrongly categorized as positive (false positives) and the total number of actual negative events (regardless of classification). False-positive rate, which is the percentage that our algorithm incorrectly identifies the attacker node.

Table 3. False Positive Rate

Percentage of attacker nodes	False Positive Rate		
	Tree-CNN	PM-RNN-IDS	CNN-OR-IDS
10	12	19	22

20	11	17	21
30	7	13	19
40	5	11	17
50	3	7	15

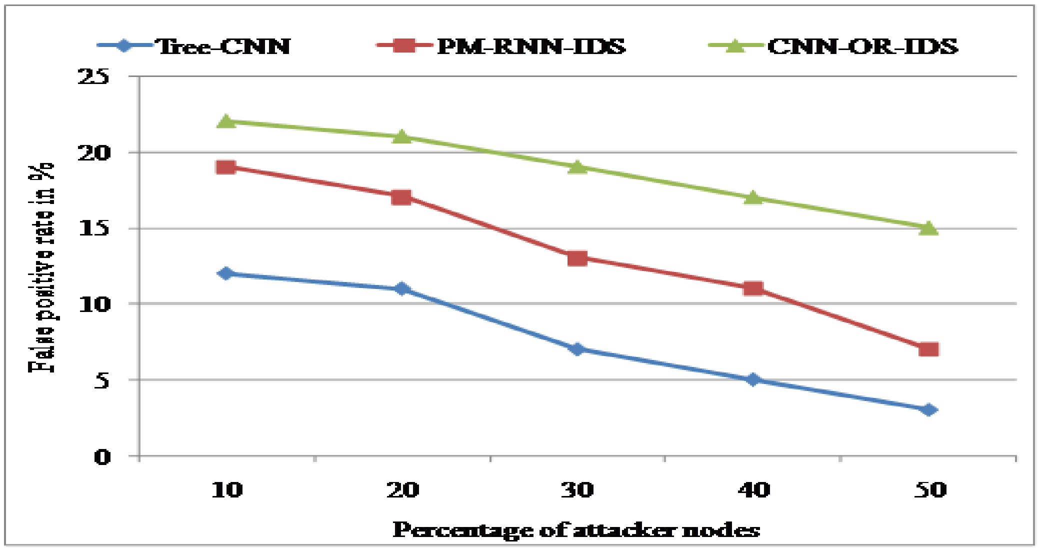

Figure 3. False positive rate vs percentage of attacker nodes

In figure 3, comparison evaluation of the false positive rate for the proposed and existing methodologies are given. From this comparison it can be proved that the proposed method CNN-OR-IDS tends to have better performance than the previous methodologies with lesser wrong detection of attacker nodes.

4.3. SUCCESS RATE VS RATIO OF COLLUDING NODE

Success rate, which is the percentage that our algorithm can correctly identify the attacker nodes.

Table 4. Success Rate

Percentage of Attacker Nodes	Success Rate		
	Tree-CNN	PM-RNN-IDS	CNN-OR-IDS
10	22	29	32
20	21	27	31
30	17	23	29
40	15	21	27
50	13	17	25

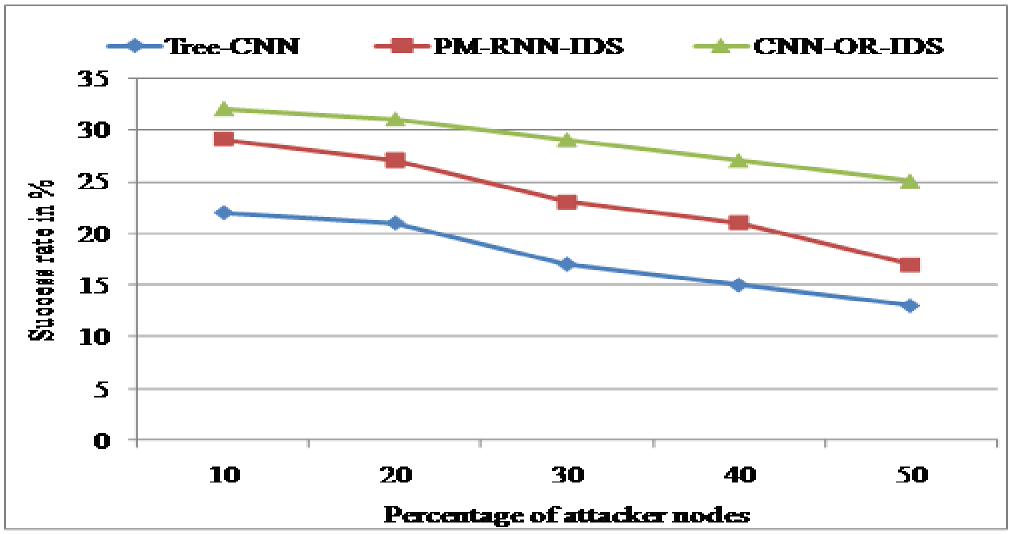

Figure 4. Success rate comparison

In figure 4, comparison evaluation of the success rate for the proposed and existing methodologies are given. From this comparison it can be proved that the proposed method CNN-OR-IDStends to have better performance than the previous methodologies with accurate detection of attacker nodes.

4.4. FLOODING ATTACK PROBABILITY VS METHOD

DoS flooding attack probability is defined as the effectiveness of proposed algorithm to capture the present of flooding attacks from the total number of DoS flooding attacks. That is, it is the ratio between the number of captured flooding attacks to the total number of flooding attacks present in the environment.

Table 5. Attack Success Probability

Methods	Attack success probability
Tree-CNN	0.2
PM-RNN-IDS	0.4
CNN-OR-IDS	0.53

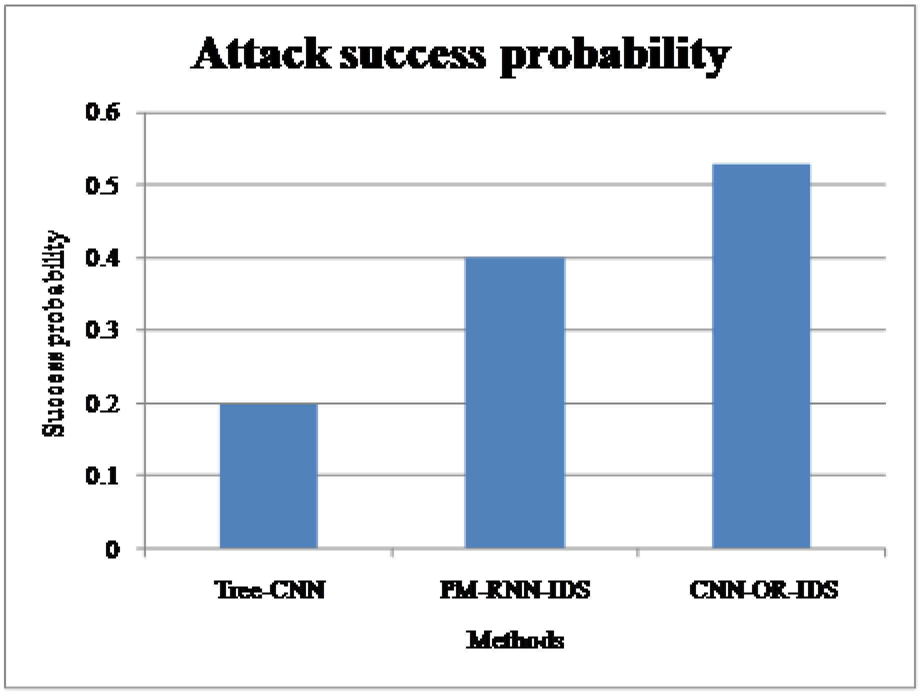

Figure 5. Flooding attack probability comparison

In figure 5, comparison evaluation of the success probability for the proposed and existing methodologies is given. From this comparison it can be proved that the proposed method CNN-OR-IDS tends to have better performance than the previous methodologies with accurate detection of attacker nodes.

5. Conclusion and Future Work

In this research work, Convolution Neural Network based Overhead Reduced Intrusion Detection System (CNN-OR-IDS) is introduced for the optimal and accurate intrusion detection rate. In this work, modified Firefly algorithm is utilized for redundant data detection and Bagging based KNN Imputation technique is utilized for the Missing value replacement process. Once the dataset is preprocessed, feature reduction is performed by introducing the method namely Gradually Feature Removal (GFR) method. Once the irrelevant features are removed from the dataset, optimal feature selection is done by using Hybrid of Cuckoo Search with fish swarm algorithm. Finally, intrusion detection process is carried out using Convolutional Neural Network algorithm. Experimental results shows that this proposed model produces better results with lower false positive rate of 15% than other existing models. However, proposed model does not test with real time data and this could be focused in the future research.

REFERENCES

1. Begli, M., Derakhshan, F. and Karimipour, H. (2019, August). A layered intrusion detection system for critical infrastructure using machine learning. In 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE) (pp. 120-124). IEEE.
2. Duhan, S. and Khandnor, P. (2016, March). Intrusion detection system in wireless sensor networks: A comprehensive review. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 2707-2713). IEEE.
3. Can, O. and Sahingoz, O.K. (2015, May). A survey of intrusion detection systems in wireless sensor networks. In 2015 6th international conference on modeling, simulation, and applied optimization (ICMSAO) (pp. 1-6). IEEE.
4. Maleh, Y., Ezzati, A., Qasmaoui, Y. and Mbida, M. (2015). A global hybrid intrusion detection system for wireless sensor networks. Procedia Computer Science, 52, pp.1047-1052.
5. Yin, C., Zhu, Y., Fei, J. and He, X. (2017). A deep learning approach for intrusion detection using recurrent neural

networks. Ieee Access, 5, pp.21954-21961.

6. Umba, S.M.W., Abu-Mahfouz, A.M., Ramotsoela, T.D. and Hancke, G.P. (2019, June). A review of artificial intelligence-based intrusion detection for software-defined wireless sensor networks. In 2019 IEEE 28th International symposium on industrial electronics (ISIE) (pp. 1277-1282). IEEE.
7. Manvith, V.S., Saraswathi, R.V. and Vasavi, R. (2021, February). A performance comparison of machine learning approaches on intrusion detection dataset. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV) (pp. 782-788). IEEE.
8. Subba, B., Biswas, S. and Karmakar, S. (2016). November. Enhancing performance of anomaly based intrusion detection systems through dimensionality reduction using principal component analysis. In 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) (pp. 1-6). IEEE.
9. Riecker, M., Biedermann, S., El Bansarkhani, R., & Hollick, M. (2015). Lightweight energy consumption-based intrusion detection system for wireless sensor networks. International Journal of Information Security, 14(2), 155-167.
10. Maleh, Y., Ezzati, A., Qasmaoui, Y., & Mbida, M. (2015). A global hybrid intrusion detection system for wireless sensor networks. Procedia Computer Science, 52, 1047-1052.
11. Hu, Z., Zhang, J., & Wang, X. A. (2016, November). Intrusion detection for wsn based on kernel fisher discriminant and svm. In International Conference on P2p, Parallel, Grid, Cloud and Internet Computing (pp. 197-208). Springer, Cham.
12. Otoum, S., Kantarci, B., & Mouftah, H. T. (2019). On the feasibility of deep learning in sensor network intrusion detection. IEEE Networking Letters, 1(2), 68-71.
13. Almomani, I., Al-Kasasbeh, B., & Al-Akhras, M. (2016). WSN-DS: A dataset for intrusion detection systems in wireless sensor networks. Journal of Sensors, 2016.
14. Amaran, S. and Mohan, R.M. (2021, March). Intrusion detection system using optimal support vector machine for wireless sensor networks. In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) (pp. 1100-1104). IEEE.
15. Safaldin, M., Otair, M. and Abualigah, L. (2021). Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. Journal of ambient intelligence and humanized computing, 12(2), pp.1559-1576.
16. Zivkovic, M., Bacanin, N., Tuba, E., Strumberger, I., Bezdan, T. and Tuba, M. (2020, June). Wireless sensor networks life time optimization based on the improved firefly algorithm. In 2020

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4281-4308
https://publishoa.com
ISSN: 1309-3452

International Wireless Communications and Mobile Computing (IWCMC) (pp. 1176-1181). IEEE.

17. Taunk, K., De, S., Verma, S. and Swetapadma, A. (2019, May). A brief review of nearest neighbor algorithm for learning and classification. In 2019 International Conference on Intelligent Computing and Control Systems (ICCS) (pp. 1255-1260). IEEE.
18. Xing, W. and Bei, Y. (2019). Medical health big data classification based on KNN classification algorithm. IEEE Access, 8, pp.28808-28819.
19. Naik, M., Nath, M.R., Wunnava, A., Sahany, S. and Panda, R. (2015, July). A new adaptive Cuckoo search algorithm. In 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS) (pp. 1-5). IEEE.
20. Feng, Y., Zhao, S. and Liu, H. (2020). Analysis of network coverage optimization based on feedback K-means clustering and artificial fish swarm algorithm. IEEE Access, 8, pp.42864-42876.
21. Fang, N., Zhou, J., Zhang, R., Liu, Y. and Zhang, Y. (2014). A hybrid of real coded genetic algorithm and artificial fish swarm algorithm for short-term optimal hydrothermal scheduling. International Journal of Electrical Power & Energy Systems, 62, pp.617-629.
22. Albawi, S., Mohammed, T.A. and Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). IEEE.
23. Lin, Y.J. and Chang, T.S. (2017). Data and hardware efficient design for convolutional neural network. IEEE Transactions on Circuits and Systems I: Regular Papers, 65(5), pp.1642-1651.