Expert Systems With Applications 211 (2023) 118588

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A deep learning approach to predict the number of k-barriers for intrusion detection over a circular region using wireless sensor networks

Abhilash Singh [a], J. Amutha [b], Jaiprakash Nagar [c], Sandeep Sharma [d,*]

[a] Fluvial Geomorphology and Remote Sensing Laboratory, Indian Institute of Science Education and Research Bhopal, 462066, India
[b] University School of ICT, Gautam Buddha University, Greater Noida, 201312, India
[c] Subir Chowdhury School of Quality and Reliability, Indian Institute of Technology Kharagpur, 721302, India
[d] Department of Electronics Engineering, Madhav Institute of Technology and Science, Gwalior, Madhya Pradesh, 474005, India

ARTICLE INFO

Keywords:
WSNs
Binary sensing model
Gaussian distribution
Uniform distribution
Barrier coverage
Deep learning

ABSTRACT

Wireless Sensor Networks (WSNs) is a promising technology with enormous applications in almost every walk of life. One of the crucial applications of WSNs is intrusion detection and surveillance at border areas and in the defence establishments. The border areas are stretched over hundreds to thousands of miles, hence, it is not possible to patrol the entire border region. As a result, an enemy may enter from any point absence of surveillance and cause the loss of lives or destroy the military establishments. WSNs can be a feasible solution for the problem of intrusion detection and surveillance at the border areas. Detection of an enemy at the border areas and nearby critical areas such as military cantonments is a time-sensitive task as a delay of a few seconds may have disastrous consequences. Therefore, it becomes imperative to design systems that can identify and detect the enemy as soon as it comes within the range of the deployed system. In this paper, we have proposed a deep learning architecture based on a fully connected feed-forward Artificial Neural Network (ANN) for the accurate prediction of the number of k-barriers for fast intrusion detection and prevention. We have trained and evaluated the feed-forward ANN model using four potential features, namely area of the circular region, sensing range of sensors, transmission range of sensors, and number of sensor for Gaussian and uniform sensor distribution. These features are extracted through Monte Carlo simulation. In doing so, we found that the model accurately predicts the number of k-barriers for both Gaussian and uniform sensor distribution with correlation coefficient (R = 0.78) and Root Mean Square Error (RMSE = 41.15) for the former and R = 0.79 and RMSE = 48.36 for the latter. Further, the proposed approach outperforms the other benchmark algorithms in terms of accuracy and computational time complexity.

1. Introduction

The unquenchable thirst of people for political and military power is compelling them to extend their boundaries and grab other people's natural resources. In order to achieve this goal, they may try different techniques such as gaining information about the military establishments, the number of military personnel at a given place, regions of natural resources, and the vulnerabilities of authorities that they can exploit. In addition, illegal immigration, smuggling of drugs, and other banned commodities across the boundaries are immediate concerns that must be dealt with immediately. Therefore, it is crucial to identify an intruder or an unauthorised activity accurately in a timely manner as they all are time-sensitive issues that may result in havoc if not prevented in time. Furthermore, encroachment in the border areas and unauthorised entry into the prohibited regions is a serious issue making border surveillance mandatory.

It is a well-known fact that most real-life problems can be solved with the help of suitable technologies. Fortunately, WSNs are widely used and is a popular technology that can resolve the concerned issue (Huang et al., 2018; Keung, Li, & Zhang, 2012; Singh, Sharma and Singh, 2021; Singh, Sharma, Singh, & Kumar, 2019). WSNs are widely deployed for various military applications such as intrusion detection in border areas, combat monitoring, an unauthorised access to prohibited areas, land mines detection, battlefield surveillance, reconnaissance, and so on (Amutha, Sharma, & Nagar, 2020; Kandris, Nakas, Vomvas, & Koulouras, 2020; Sharma & Nagar, 2020; Si, Ma, Tao, Fu, & Shu, 2020).

* Corresponding author.
E-mail addresses: sabhilash@iiserb.ac.in (A. Singh), roniamutha@gmail.com (J. Amutha), jpnagar@iitkgp.ac.in (J. Nagar), sandeepsvce@mitsgwalior.in (S. Sharma).

https://doi.org/10.1016/j.eswa.2022.118588
Received 1 July 2021; Received in revised form 7 August 2022; Accepted 13 August 2022
Available online 19 August 2022
0957-4174/© 2022 Elsevier Ltd. All rights reserved.

Researchers have proposed various border surveillance and intrusion detection techniques using WSNs (Amutha, Nagar and Sharma, 2021; Arjun, Indukala, & Menon, 2019; Aseeri, Ahmed, Shakib, Ghorbel, & Shaman, 2017; Benahmed & Benahmed, 2019; Gavel, Raghuvanshi, & Tiwari, 2021; Karanja & Badru, 2021; Karthick, Prabaharan, & Selvaprasanth, 2019; Mostafaei, Chowdhury, & Obaidat, 2018). The proposed techniques use either simulations or Internet of Things (IoT) methods to validate their techniques. However, simulation methods for the validation of proposed techniques have high time complexity, *i.e.*, the time taken to obtain a single output at a given value of sensor and sensing range is in several hours. Also, IoT devices are very expensive and require a huge financial investment. The high time complexity and financial issues can be minimised to a negligible level using machine learning approaches to validate and predict the performance of WSNs before their actual deployment in a given region (Kotiyal, Singh, Sharma, Nagar, & Lee, 2021; Mishra, Varadharajan, Tupakula, & Pilli, 2018; Singh, Nagar, Sharma and Kotiyal, 2021). However, the accurate and timely detection and prevention of intrusion through machine learning approaches is still an ill-posed problem that has been insufficiently investigated. To address this issue, we propose a deep learning architecture for accurate and timely intrusion detection and prevention.

In this paper, we proposed a fully connected feed-forward ANN architecture to predict the number of k-barriers for accurate intrusion detection and prevention in WSNs using potential features. We have extracted four features, namely area of the circular region, sensing range of sensors, the transmission range of sensors, and the number of sensors through the Monte Carlo simulation approach. Afterward, we used these features to train and evaluate the performance of the feed-forward ANN model using R, RMSE, bias, and computational time complexity as performance metrics.

Further, the rest of the paper is divided into seven sections. In Section 2, we have discussed the related works. In Section 3, we have presented the system model. In this section, we have discussed the sensor distribution models and the sensing model. In Section 4, we have discussed the simulation experiment. In Section 5, we have discussed the machine learning model. In this section, we have discussed the feature importance, feature sensitivity, and model setup. In Section 6, we have presented the results. Lastly, in Sections 7 and 8, we have presented the discussion and conclusion of this study, respectively.

2. Related works

Deep learning is a subset of machine learning algorithms which has been applied for intrusion detection using WSNs (Amutha, Sharma and Sharma, 2021; Lee et al., 2021; Singh, Amutha, Nagar, Sharma, & Lee, 2022a, 2022b; Sood, Prakash, Sharma, Singh, & Choubey, 2022). It is also employed for pattern matching and network security where it identifies the malicious activities occurring in the network and is termed as Network Intrusion Detection System (NIDS). The accuracy of the NIDS can be improved with the help of Recurrent Neural Network IDS (RNNIDS) (Sohi, Seifert, & Ganji, 2021) which is capable of identifying the complex patterns resulting in an enhanced intrusion detection rate. Authors in Yin, Zhu, Fei, and He (2017) have proposed an RNNs approach which examines the system behaviour, type of intrusion, and the impact of intrusion on the intrusion detection accuracy with the help of learning rate and the number of neurons. The major limitation of RNNIDS is that it fails to minimise the false positives; thus, not able to achieve the maximum detection rate. This issue has been dealt with the help of a deep learning architecture that combines classifiers with Convolution Neural Networks (CNN) having Long Short Term Memory (LSTM), thus, offering maximum detection rate (Pektaş & Acarman, 2019). Here, the CNN acquires spatial information, and the LSTM acquires temporal features from the received packets in the network. The optimal parameters in the feature space are obtained by employing an estimator known as tree-structured Parzen. This method focuses on generating flow-based statistical features rather than dataset features. Although the abnormal traffic can be detected with an accuracy rate of 99.09% and a false alarm rate of 0.0227, it fails to compute the computational time complexity of flow-based intrusion detection method.

A deep learning approach that uses an auto-encoder strategy exhibits an improvement in the time complexity by 18.12% (Abbasi, Bashir, Qureshi, ul Islam, & Jeon, 2021). This strategy uses multilayer perception to replace the detection of hierarchical features and unsupervised feature learning. Thus, it enhances the pattern matching mechanism for intrusion detection with the help of Deep Learning-based Feature Extraction (DLFE), which is an Optimisation of Pattern Matching (OPM) approaches. Another approach called Scale-Hybrid-IDS-AlertNet (SHIA) (Vinayakumar et al., 2019) performs data computation at the network and the host-level for the intrusion detection and delivers the relevant alert notifications to the controller automatically. A Deep Neural Network (DNN) can render an effective IDS that can detect and classify the intruders crossing the Region of Interest (RoI). Here, the performance of SHIA is measured in terms of multi-class classification of the DNN, accuracy, True Positive Rate (TPR), and False-Positive Rate (FPR). Although SHIA is scalable and shows improved performance in handling large data-sets of real-time systems, it does not compute the number of k-barriers and incurs high computational costs.

Despite having a high computational complexity, ensemble-based techniques have a high level of accuracy as compared to the base models. An ensemble based DNN framework for the continuous analysis of intrusion detection along with the ability to learn hierarchical data-sets automatically has been proposed in Folino, Folino, Guarascio, Pisani, and Pontieri (2021). Here, a log-stream of an intrusion detection system maintains an ensemble that contains classifiers trained on discrete chunks of the data-set instance and a combiner model. The combiner model performs reasoning on the instance parameters and classifier predictions; after that, the efficacy is computed in terms of data scarcity and accuracy that allows to analyse the diverse ensemble aggregation strategies. This ensemble approach utilises the unstructured data that does not comply with the transfer learning strategies and is found in the log-stream of NIDS.

Barrier coverage using WSNs plays a crucial role in the detection of an unauthorised personnel or object trying to enter the prohibited region. Barriers are formed by the sensors deployed over the entire given RoI. The early depletion of sensor's energy leads to the failure of the sensors causing blind spots in the barriers. To overcome this issue, authors in Saraereh, Ali, Al-Tarawneh, and Khan (2021) have proposed a set-based max-flow procedure for the re-deployment of mobile sensors. This method determines the vulnerable locations and deploys the mobile sensors which strengthens and prolongs the longevity of the barrier. Although the algorithm exhibits higher efficiency in terms of the computation time, it fails to predict the number of k-barriers for the IDS. In Nagar and Sharma (2018), the authors have derived an analytical closed-form expression using mobile sensors for the k-barrier coverage probability of a WSN. Here, they have calculated the total area covered by an intruder travelling at a given angle to cross the RoI, Then, this total area is utilised to obtain the closed-form expression for the k-barrier coverage probability of the WSN. In another work presented in Singh, Nagar et al. (2021), the authors have employed three machine learning approaches, namely Gaussian Process Regression (GPR), Scaling GPR (S-GPR), and Center mean GPR (C-GPR) to predict the k-barrier coverage probability of a WSN. The proposed GPR technique quickly detects and prevents any intrusion from taking place at any location in the RoI. A three-level hierarchy scheme to detect a mobile intruder in distributed WSNs is proposed in Sharma and Chauhan (2020) improving the precision of intrusion detection. To maximise the probability of intrusion detection, the authors have used diverse sensing techniques, k-mean clustering, and the Likelihood Ratio Test (LRT) methods. The LRT fusion rule used in this scheme is efficient

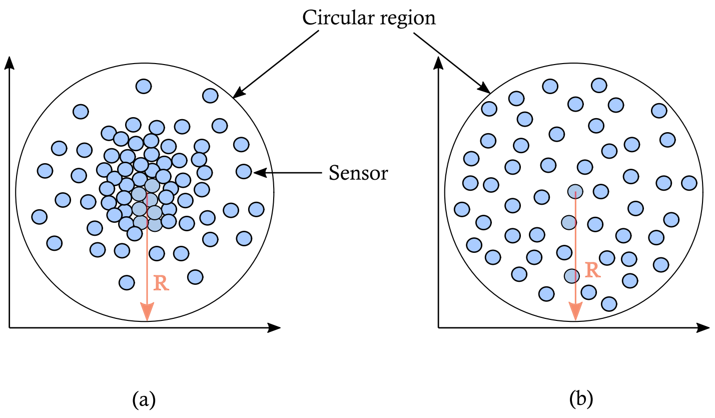

Fig. 1. Illustration of (a) Gaussian sensor distribution and (b) Uniform sensor distribution. The blue-filled circles represent sensors.

in terms of metrics like detection probability and false alarm rate at a given number of sensors and the speed of the mobile intruder.

The algorithms and strategies proposed in the literature fail to accurately predict the number of k-barriers for intrusion detection. Hence, the overall aim of this study is to overcome the limitation of the previous studies in terms of accuracy and computational time complexity using a deep learning approach.

3. System model

This section briefly discusses sensor distribution models, sensing range model, and the performance metric, namely the number of k-barriers.

3.1. Sensor distribution model

The choice of sensor distribution model depends on the required application. In this work, we consider two-sensor distribution models; (i) Gaussian distribution model and (ii) uniformly distribution model.

3.1.1. Gaussian sensor distribution model

In this model, a finite number of sensors are installed in a finite circular region of radius R meters following a 2D Gaussian distribution, also known as a normal distribution (Fig. 1a). Thus, the Probability Density Function (PDF) for a location (x, y) to be installed with a sensor is given by Wang, Xie, and Agrawal (2008)

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{(x-x_c)^2}{2\sigma_x^2} + \frac{(y-y_c)^2}{2\sigma_y^2}\right)} \tag{1}$$

where (x_c, y_c) represents the centre of the circular region, σ_x and σ_y are the standard deviations of x and y location coordinates respectively. Furthermore, the location of a sensor inside the circular region represented by (γ, ϕ) can also be modelled with the help of position coordinates (x, y) if

$$f(x,y) : \sqrt{(x-x_c)^2 + (y-y_c)^2} \le R = f(\gamma, \phi) : \gamma \le R \tag{2}$$

3.1.2. Uniform sensor distribution model

In this model, a finite number of sensors are distributed uniformly and randomly inside a finite circular region (denoted by $\mathfrak{R}$) of radius R meters (Fig. 1b). The position of a random sensor within the circular region is represented by $P_c = (\gamma, \phi)$, where, $\gamma \in [0, R]$, denotes the distance of the sensor from the centre of the circular region, and $\phi \in [0, 2\pi]$, denotes the angle between the x-axis and the line that passes through the sensor location. The resulting sensor distribution probability density function is given by

$$f_p(\mathfrak{R}) = \begin{cases} 1, if\, P_c \in \mathfrak{R} \\ 0, otherwise \end{cases} \tag{3}$$

Furthermore, the probability that a sensor is located at an arbitrary position $P_c = (\gamma, \phi)$ inside the circular region is given by

$$f(P_c) = \frac{1}{\pi R^2} \tag{4}$$

3.2. Binary sensing range model

The binary sensing range model is one of the most widely employed sensing range models for estimating the performance of WSNs (Laranjeira & Rodrigues, 2014; Nagar, Chaturvedi, & Soh, 2020). A point i denoted by $P_i(x_i, y_i)$ inside the 2D circular region will be covered by a sensor j located at $S_j(x_j, y_j)$, if the Euclidean distance of point i from the sensor j is less than or equal to the sensing range r_s of the sensor. Mathematically, it can be represented as

$$P(S_i) = \begin{cases} 1, if\, d(S_i, P_i) \le r_s \\ 0, otherwise \end{cases} \tag{5}$$

3.3. Coverage graph

Coverage measures how well sensors monitor the RoI in which they are deployed. The k-coverage ensures that each point in the target RoI is covered by at least k distinct sensors, where k is a positive integer having a typical value greater than one. A connected k-coverage is achieved when each point in the RoI is covered by at least k distinct sensors and each sensor is able to communicate with each other. An optimal k-coverage of the network is obtained when each point within the RoI is covered by at least k distinct sensor without any overlapping area. A Coverage Graph (CG) is denoted by *CG(N) = (V, E)*, where V denotes the number of vertices and E denotes the number of edges. Here, each vertex represents a sensor and an edge represents a link between any two sensors if and only if they fall within the coverage of each other. To construct a CG for a circular RoI, we have built sensor-disjoint cycles within the entire RoI. A WSN rendering two-barrier coverage inside a circular RoI is shown in Fig. 2a, and its respective CG is depicted in Fig. 2b.

3.4. Barrier and barrier path

A barrier along the boundaries of a circular RoI can be constructed by taking the union of distinct sensor coverage. Any arbitrary point from where an intruder may enter the RoI is known as the point of

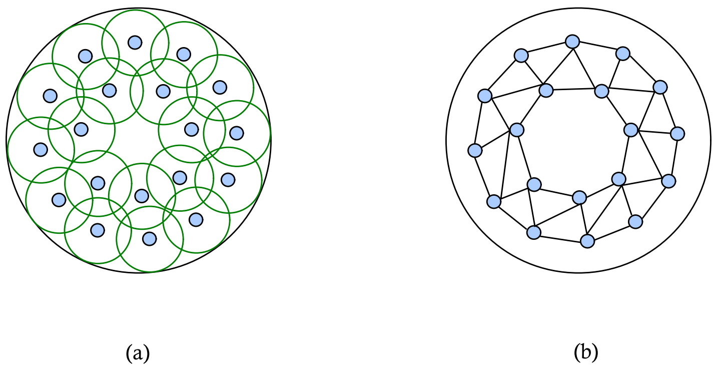

Fig. 2. Illustration of (a) 2-Barrier coverage and (b) Coverage graph.

Table 1
Simulation parameters.

Parameters	Values
Simulator	NS-2.35
Radius of the circular region (R)	40 to 127 m
Number of sensors (N)	100 to 400
Sensing range (R_s)	15 to 40 m
Transmission range (R_{tx})	30 to 80 m
Mac type	IEEE 802.11
Sensor's deployment type	(a) Gaussian Distribution
	(b) Uniform Distribution
Sensing model	Binary sensing model

intrusion and any possible path that an intruder may follow to reach the target point is called an intrusion path. In order to provide a guaranteed barrier coverage, there must exist at least one barrier for every possible intrusion path. In this way, any intrusion attempt can be identified and prevented in a timely manner. The maximum number of barrier paths BP_{max} that can be constructed for a given intrusion path without any overlapping coverage is given by Eq. (6).

$$BP_{max} = \lfloor \frac{N}{k} \rfloor \tag{6}$$

where N represents the total number of sensors and k represents the number of sensors required to ensure k barrier coverage for a possible intrusion path.

4. Simulation experiment

We have obtained the simulation results using network simulator NS-2.35, one of the widely used network simulators to obtain the performance metrics of WSNs. Table 1 shows different network parameters and their values used to get the simulation results for the number of k-barrier paths. Here, we have assumed that any two sensors can communicate with each other iff the transmission range of sensors (R_{tx}) is at least twice the sensing range of sensors (R_s), *i.e.*, $R_{tx} \geq 2R_s$.

5. Machine learning model

Machine learning algorithms are broadly classified into supervised and unsupervised learning algorithms. In supervised machine learning, we work with labelled data sets. It is mainly used to solve either regression or classification problems. In contrast, unsupervised machine learning algorithms deal with unlabelled data sets and are mainly used to perform clustering and dimension reduction tasks.

In this study, our objective is to assess the potential of fully connected feed-forward ANN to map the number of k-barriers using relevant features. To evaluate the relevancy of the selected features, we have calculated the feature importance score and performed feature sensitivity analysis.

5.1. Feature importance

This study used the area of the RoI, sensing range, transmission range, and the number of sensors as the potential features and k-barriers as the predictand. In machine learning, the selection of input features significantly affects its performance (Singh, Kotiyal, Sharma, Nagar, & Lee, 2020). Hence, before training the machine learning model, we have evaluated the relative importance of each selected feature on the predictand. In doing so, we opted regression tree ensemble technique (Torres-Barrán, Alonso, & Dorronsoro, 2019). We have first trained a regression tree ensemble model by boosting hundred regression trees using the Least Squares gradient Boosting (LSBoost) ensemble aggregation method (*i.e.*, $r = 100$), each with a learning rate of one (*i.e.*, $\alpha = 1$), and the classical decision tree (*i.e.*, decision stumps) has been considered as a weak learner. The LBoost algorithm trains one weak learner at a time and also detects its weak points. Based on such weak points, it generates a new weak learner (l_i) and evaluates its weight (*i.e.*, w_i). According to (Eq. (7)), the algorithm improves the current model (M_i) by emphasising on the prior weak learner's (M_{i-1}) weak point. After it has been trained, it integrates the weak learner into the existing model, and creates a single strong learner (M_r, *i.e.*, an ensemble of weak learners) iteratively.

$$M_i = M_{i-1} + \alpha \, . \, w_i \, . \, l_i \qquad (i = 1, 2, 3, \ldots, r) \tag{7}$$

In addition, we determine the relative feature importance score by evaluating the overall variations in the node risk (ΔR) due to the split on each feature, and then normalising it by the total number of branch nodes (R_{bn}). Mathematically, it is represented as in (Eq. (8));

$$\Delta R = \frac{R_p - (R_{ch1} + R_{ch2})}{R_{bn}} \tag{8}$$

where R_p denotes the node risk of the parent and R_{ch1} & R_{ch2} denotes the node risk of two children. The node risk at an individual node (R_i) is mathematically represented as in (Eq. (9));

$$R_i = P_i \, . \, E_i \tag{9}$$

where P_i denotes the probability of node i and E_i denotes the mean square error of the i^{th} node.

5.2. Feature sensitivity

Estimating the feature importance score only tells us about the relative importance of each feature. However, it does not convey how the features are associated with the predictand, *i.e.*, whether the predictand value increases with feature (positive impact) or decreases with features (negative impact). To evaluate this, we have performed the sensitivity analysis of the features using the Partial Dependence Plot (PDP) (Friedman, 2001; Singh, Nagar et al., 2021). PDP measures the average effect of a single or more features by marginalising the effect of all other features taken into consideration. We considered the combined

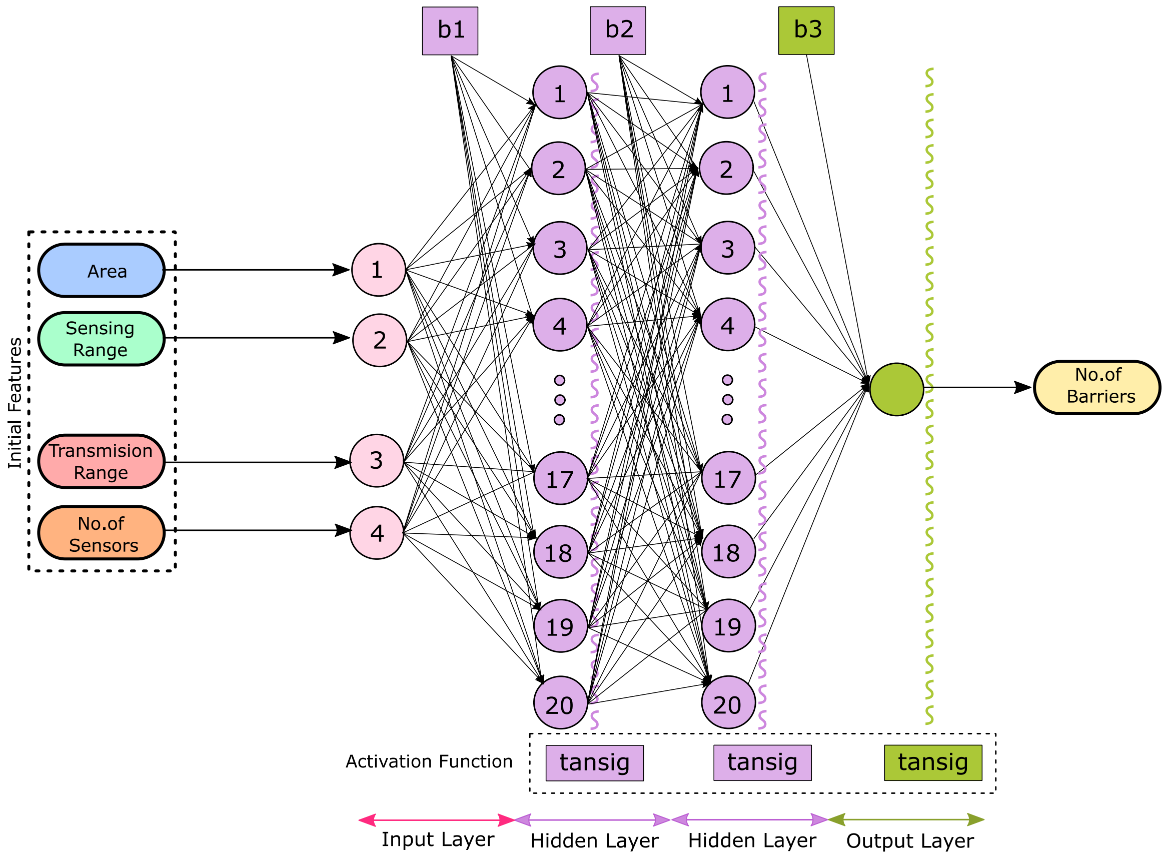

Fig. 3. Structure of the fully connected feed-forward ANN model having four inputs, two hidden layers having 20 neurons each, and one output (i.e., 4:20:20:1).

impact of two features simultaneously from the input feature set (*i.e.,* ϑ) on the predictand by marginalising the impact of the remaining features. To do so, a subset ϑ^s and a complimentary set (ϑ^c) of ϑ^s are extracted from the feature set ($\vartheta = \{k_1, k_2, \dots, k_n\}$) where, n represents the total features. Using (Eq. (10)), we can compute any prediction on ϑ.

$$f(\vartheta) = f(\vartheta^s, \vartheta^c) \tag{10}$$

The partial dependence of the feature in ϑ^s can be determined by calculating the expectation (E_c) of Eq. (11).

$$\begin{aligned} f^s(\vartheta^s) &= E_c[f(\vartheta^s, \vartheta^c)] \\ &= \int f(\vartheta^s, \vartheta^c) \;.\; \rho_c(\vartheta^c) \;.\; d\vartheta^c \end{aligned} \tag{11}$$

where $\rho_c(\vartheta^c)$ indicates the marginal probability of ϑ^c, which is represented in Eq. (12).

$$\rho_c(\vartheta^c) \approx \int p(\vartheta^s, \vartheta^c) \;.\; d\vartheta^s \tag{12}$$

Then, the partial dependency of the feature in ϑ^s can be determined by :

$$f^s(\vartheta^s) \approx \frac{1}{T} \sum_{i=1}^{T} f(\vartheta^s, \vartheta^c_i) \tag{13}$$

where T represents the total number of observations.

5.3. Model setup

In this subsection, we have discussed the architecture of the fully connected feed-forward ANN including its working, activation function, and the training algorithm.

5.3.1. Feed forward ANN

A Neural Network (NN) is a model which mimics the oversimplification of the brain performance that operates under a particular specific function of interest. The primary objective of the NN model is to discover a mapping function *(f)* that predicts a target function *(f*)* through training the NN using labelled training data sets. During the training phase, the network learns the range of parameters from the training data. Once we trained the model, we need to validate and test its performance using the unseen data.

A network can be subdivided into basic information-processing elements called neurons, which are the building blocks of ANN. Layers are groups of neurons, and the network is comprised of interconnections between these layers. There are diverse perspectives of linking layers together that lead to several other forms of NNs like feed-forward neural networks, recurrent neural networks, and convolutional neural networks. In general, the optimisation problem of a neural network can be represented by Eq. (14), where l represents the loss function, and W is the learnable parameter. The main objective is to learn W so that the variance between the output of f can be minimised, when the input x_0 and the actual output y are provided.

$$min\ l(f(x_0, W), y) \tag{14}$$

In this study, we have trained a fully connected feed-forward ANN. It is a category of NN formed by organising neurons so that all neurons in each layer are linked to every other neuron in the adjacent forward layer. The data flows mainly in one direction, *i.e.,* forward, from the input neurons to the output, passing through the hidden layers (if any). Since each neuron has one activation function, the total activation function for a layer equals the total output value. The training algorithm utilises the output findings to calibrate the sensor connection weight value (Moldovan, Caţaron, & Andonie, 2020; Novickis, Justs, Ozols, & Greitāns, 2020). Any continuous function can be approximated by a feed-forward ANN with one hidden layer. However, the desired hidden size might be high, making learning unfeasible. Feed-forward ANN is well-suited for unstructured data, such as data that is not sequential or time-dependent.

In this study, we structured a feed-forward ANN that consists of two hidden layers and one output layer, as illustrated in Fig. 3. Each hidden layers consist of twenty neurons. A common bias value is added to each neuron in the hidden layers, which is followed by an activation function.

5.3.2. Activation function

In feed-forward ANN, the activation function is one of the essential key elements of the neuron (Duch & Jankowski, 1999). It significantly impacts the performance of the neural networks by modifying the neuron output. The Universal Approximation Theorem states that a feed-forward ANN with one hidden layer and an arbitrary sigmoidal function with adequate sensors can estimate any continuous function with no restrictions on the number of sensors or the size of the weights. In this study, we have used the hyperbolic tangent sigmoid transfer function at each layer because it is a bipolar sigmoid function that has a positive response for a positive input and a negative response for a negative input. Hence, it eliminates the problem of negative responses for positive values. Further, as the complexity and non-linear of the problems increases (when we increase the number of sensors and the monitoring area in WSNs), the advantage of using the hyperbolic tangent sigmoid transfer function becomes more apparent. The mathematical model is expressed as:

$$a = \frac{2}{(1 + e^{(-2 \cdot n)}) - 1} \tag{15}$$

This expression is mathematically equivalent to tanh(n). However, the computational time complexity of Eq. (15) is lower than tanh(n).

5.3.3. Training algorithm

To minimise the error in the output, the values of weights and biases need to be updated. This is done with the help of a backpropagation training algorithm. In the backpropagation algorithm, the input is transmitted to the hidden layer, which then perpetuates back the sensitivity in order to minimise the error rate by updating the weights and the bias during the process. This algorithm results in low convergence, and in some cases, it also leads to over-fitting. To address these issues, and for quick convergence without over-fitting, approaches like Levenberg–Marquardt backpropagation (LM) and Bayesian Regularization (BR) backpropagation, and Scaled Conjugate Gradient (SCG) backpropagation have been developed. To minimise the sum of squares at every iteration, LM utilises the conjugate gradient backpropagation method. LM is used for curve fitting problems, while SCG is used for pattern recognition problems. Bayesian regularization backpropagation algorithm uses an objective function that incorporates the sum of squared weights and the residual sum of squares in order to reduce the prediction errors for attaining the desired model.

In this study, we have used the LM backpropagation algorithm. It is a non-linear optimisation-based approach for training ANNs, by which it uses second-order derivatives for improved convergence behaviour. The LM algorithm offers the features of the steepest descent approach along with the Gauss–Newton method, which provides an invertible matrix, named Hessian matrix *(H)* which is shown in Eq. (16):

$$H(x) \approx J^T(x)J(x) + \mu I \tag{16}$$

where, μ represents the combination co-efficient, J and I represents the Jacobian and identity matrix respectively. The LM modification to the Gauss–Newton algorithm (Hagan & Menhaj, 1994) is represented in Eq. (17):

$$\Delta x = [J^T(x)J(x) + \mu I]^{-1} J^T(x)e(x) \tag{17}$$

The algorithm seems to be the steepest descent when the value of μ becomes high, whereas the algorithm is Gauss–Newton when the value of μ is minimal. The LM algorithm for the weight update rule (Mathew, Griffin, Alamaniotis, Kanarachos, & Fitzpatrick, 2018) is determined as a function of Jacobian matrix and error vector (e), which is represented in Eq. (18):

$$w(t+1) = w(t) - (J_t^T J_t + \mu I)^{-1} + J_t e_t \tag{18}$$

We have randomly divided (using the Mersenne Twister generator) the complete data set (182 × 5) in a 55:15:30 ratio for training, validation, and testing of the feed-forward ANN algorithm. The complete methodology is shown in Fig. 4.

6. Results

This section presents the results of feature importance analysis, feature sensitivity analysis, and feed-forward ANN.

6.1. Feature importance of each features

Once we evaluated the feature importance of each feature for both Gaussian and uniform sensor distribution, we plotted the relative importance graph for both scenarios (Fig. 5). The higher the importance score, the more important the feature is. We found that the relative importance score of the sensing range, transmission range, and sensors is the same and the maximum. In contrast, the area has the least relative importance amongst all the features for both scenarios.

6.2. Feature sensitivity curve

To analyse the feature sensitivity, we have plotted the surface plot of PDP considering two features at a time along with the corresponding two-dimensional plot with an axis-aligned histogram representing the distribution of the features. For four features, we have six possible sensitivity plots. Again, we have plotted it for both scenarios *i.e.,* for Gaussian and uniform sensor distribution in Figs. 6 and 7, respectively.

For both scenarios, we observed that the area of the circular region has a negative impact on the number of barriers. In contrast, sensing range, transmission range, and the number of sensors positively impact the number of barriers. In a nutshell, we observed a similar trend with slight variation in the values for both scenarios.

6.3. Performance of the fully connected feed-forward ANN model

Once we trained the feed-forward ANN model for Gaussian and Uniform distribution scenarios, we evaluated its performance by plotting a linear regression curve between the predicted and observed barriers. We have used R, RMSE, and bias as the performance metrics. A high value of R represents that the predicted values are well in accord with the observed value. A low value of RMSE represents a more accurate model. A positive value of bias shows overestimation, and a negative value of bias shows underestimation. Afterward, we performed error and residual analysis for a robust conclusion.

6.3.1. For Gaussian sensor distribution

To evaluate the performance of the trained feed-forward ANN for Gaussian sensor distribution, we have reported the training, validation, testing, and overall accuracy. For training accuracy, we fed the training data set as input to the trained feed-forward ANN and evaluated its performance. We found that the trained model work reasonably well on the training data set with R = 0.82, RMSE = 43.38, and bias = −5.11 (Fig. 8a). The presence of a small negative bias indicates that the values are slightly getting underestimated. Evaluating the model performance using the test data results in bias study and hence its performance needs to be evaluated using the unseen/new data sets. In doing so, we have

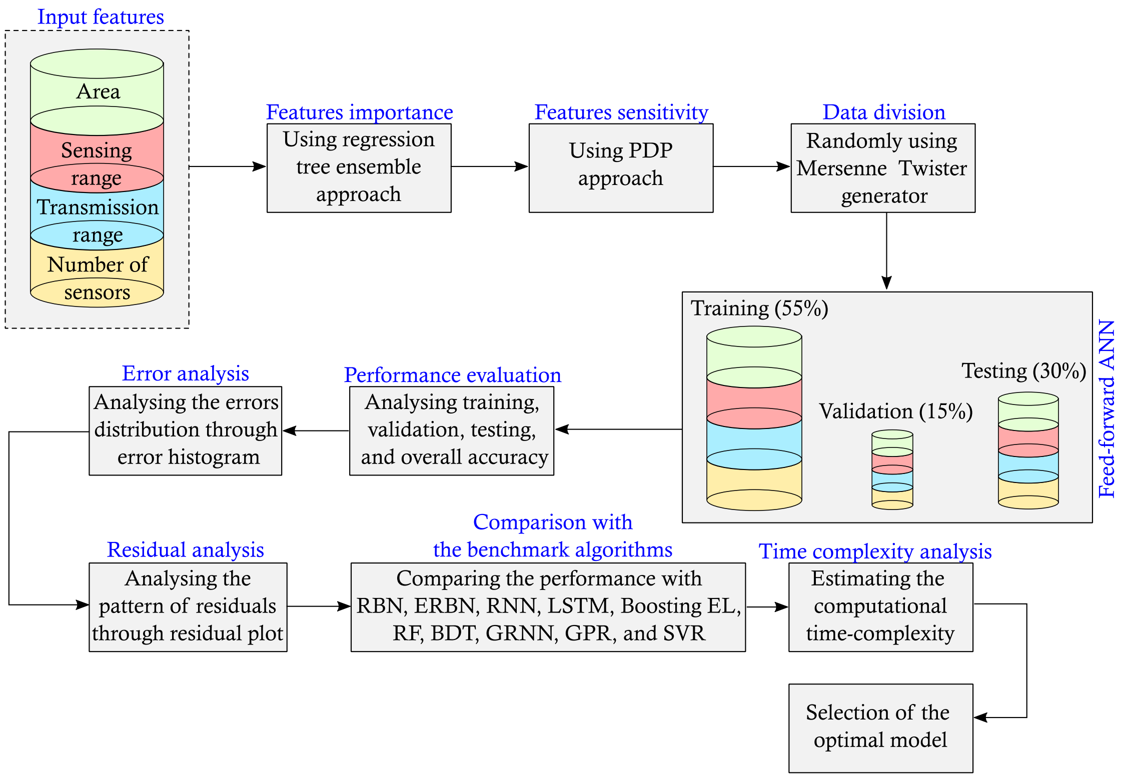
Input features
Area
Sensing range
Transmission range
Number of sensors
Features importance
Using regression tree ensemble approach
Features sensitivity
Using PDP approach
Data division
Randomly using Mersenne Twister generator
Training (55%)
Validation (15%)
Testing (30%)
Feed-forward ANN
Performance evaluation
Analysing training, validation, testing, and overall accuracy
Error analysis
Analysing the errors distribution through error histogram
Residual analysis
Analysing the pattern of residuals through residual plot
Comparison with the benchmark algorithms
Comparing the performance with RBN, ERBN, RNN, LSTM, Boosting EL, RF, BDT, GRNN, GPR, and SVR
Time complexity analysis
Estimating the computational time-complexity
Selection of the optimal model

Fig. 4. Flowchart of the methodology.

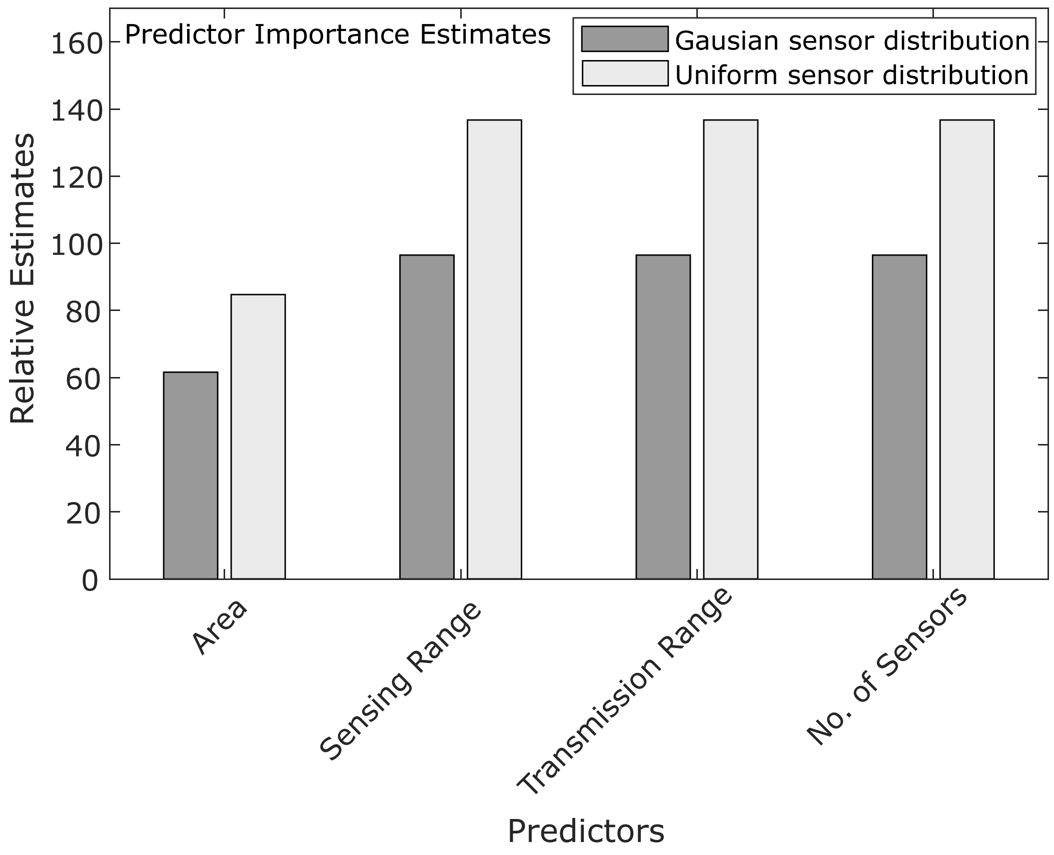

Fig. 5. Feature importance graph.

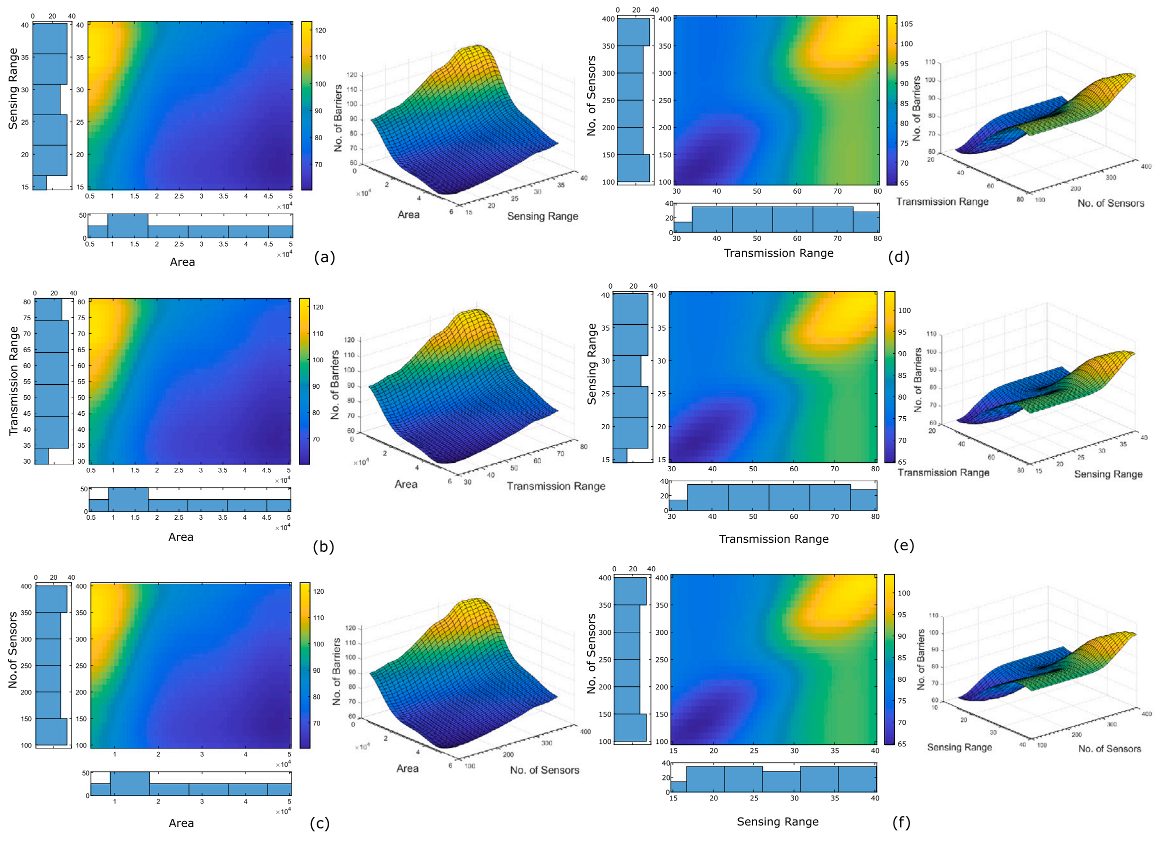

Fig. 6. Partial dependency plot for circular region considering Gaussian sensor distribution.

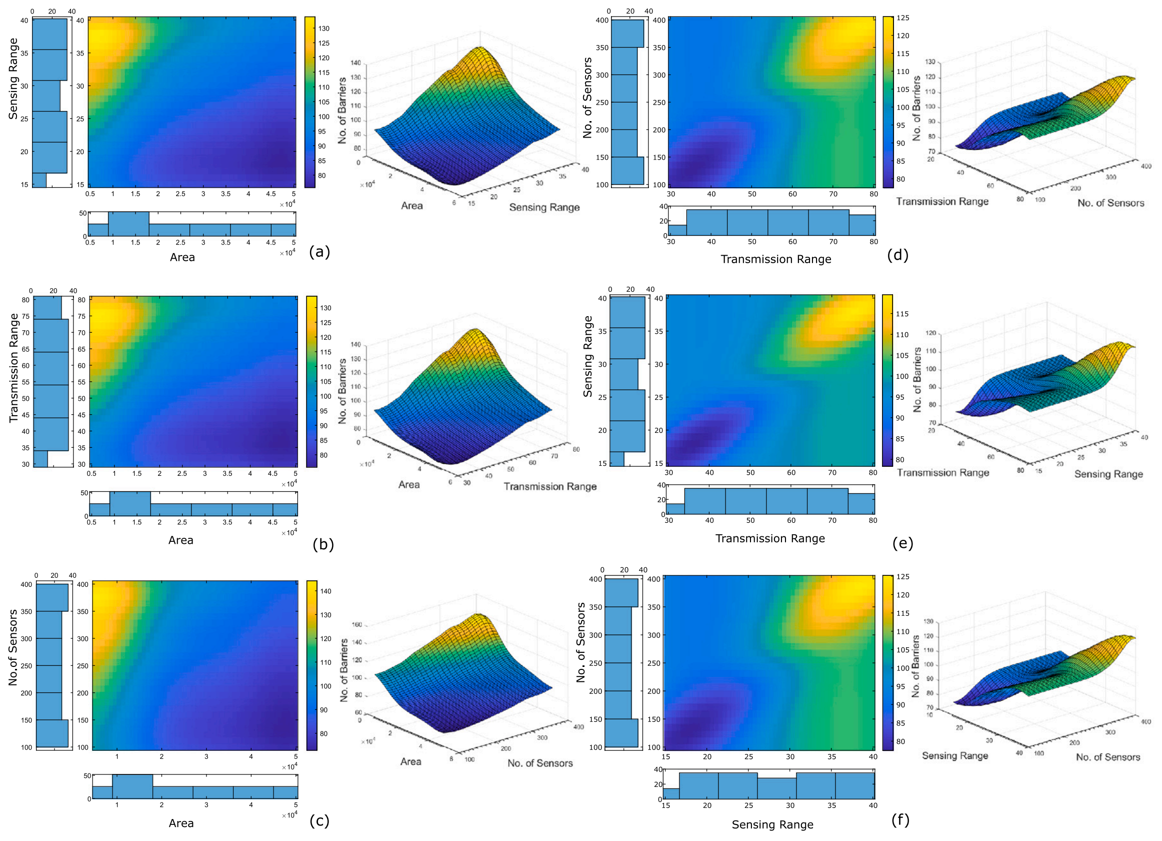

Fig. 7. Partial dependency plot for circular region considering uniform sensor distribution.

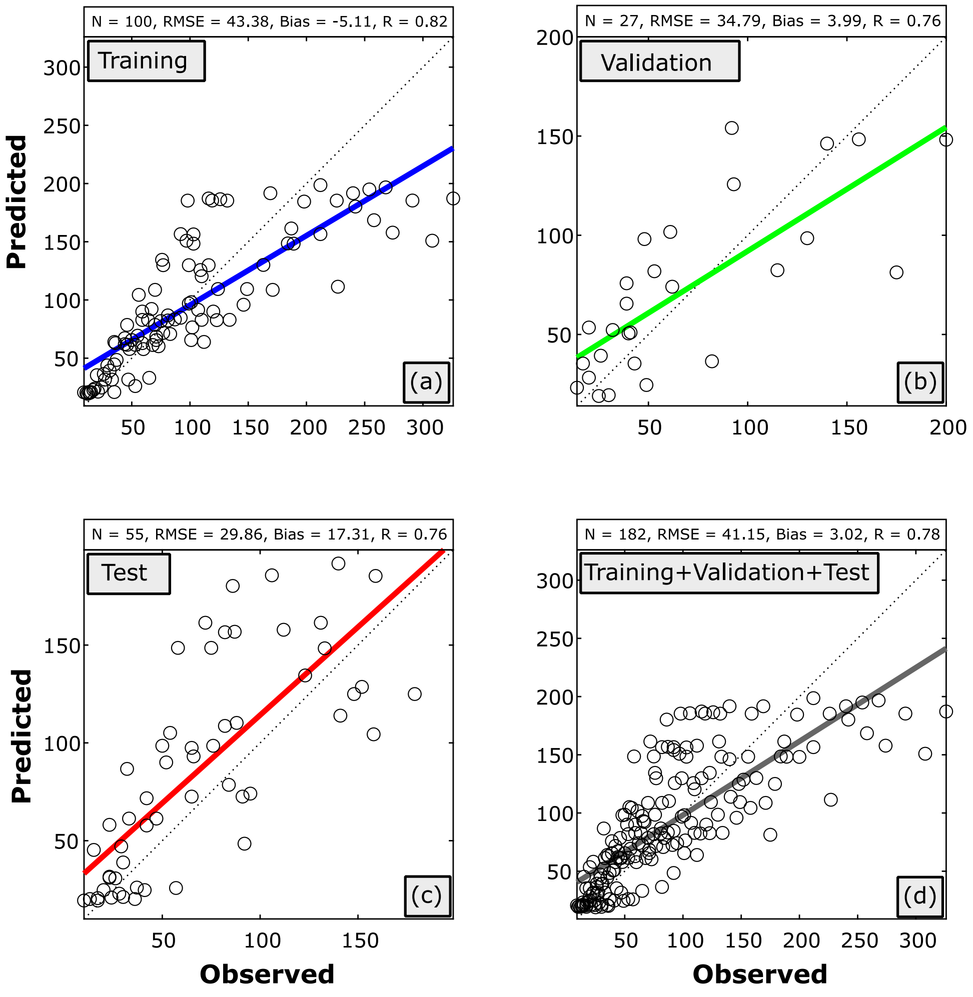

Fig. 8. Performance of the feed-forward ANN for Gaussian sensor distribution case (a) training accuracy, (b) validation accuracy, (c) testing accuracy, and (d) overall accuracy.

first validated the trained feed-forward ANN model through validation data set (Fig. 8b). We found that the trained model performs well and results in a good fit (with R = 0.76, RMSE = 34.79, and bias = 3.99) while tuning the hyper-parameters. Afterward, we used the test data for unbiased evaluation of the trained model (Fig. 8c). We observed that the trained model performed well on the test data (with R = 0.76, RMSE = 29.86, and bias = 17.31). The predicted value accord well with the observed value with slight scattering. The presence of positive bias represents that the values are slightly overestimated during the testing phase. Finally, we combined all the data sets together (training, validation, and testing) and fed it into the trained feed-forward ANN model to calculate the model's overall accuracy. We found that the trained model performs reasonably well on the complete data sets with R = 0.78, RMSE = 41.15, and bias = 3.02 (Fig. 8d).

Further, to analyse the error distribution during the training, validation, and testing phase, we have performed error analysis and plotted the combined error histogram using twenty bins (Fig. 9). The combined error from the trained feed-forward ANN model ranges from −87.94 (leftmost bin) to 150.8 (rightmost bin) and follows a slightly right-skewed Gaussian distribution. The peak of the distribution lies near the zero error line indicating a more accurate model. The region left to the zero error line indicates overestimated region, and the one on the right represents an underestimated region. Overall, the number of instances in the overestimated region is higher than the underestimated region, results in the overestimation of predicted values by the trained deep learning model. This statement is validated by the presence of a positive bias of 3.02 (Fig. 8d).

Furthermore, to evaluate the appropriateness of the trained model, we have performed the residual analysis and plotted the time series plot of the test data along with the corresponding residual plot. We have plotted the observed values (in blue) and predicted values (in red) along with their 95% Confidence Interval (C.I). The dashed line represents the RMSE value of the testing phase. The residuals are well scattered and do not follows any specific pattern indicating a good fit (Fig. 10).

6.3.2. *For uniform sensor distribution*

Similar to the Gaussian sensor distribution scenario, we have also evaluated the performance of the deep learning model for uniform sensor distribution. We reported the training, validation, testing, and overall accuracy of the feed-forward ANN model that we trained for uniform sensor distribution. For training accuracy, we have evaluated the model over the training data sets. In doing so, we observed that the model performs quite well over the training data sets (Fig. 11a). The predicted values are close to the observed values (with R = 0.79, RMSE = 55, and bias = 3.65). However, the RMSE is high, and R is slightly low as compared to the training accuracy of the Gaussian sensor distribution. Afterward, we feed the validation data sets to the model input to report the validation accuracy. The predicted values (while tuning the hyper-parameters) are in agreement with the observed values with R = 0.81, RMSE = 34.81, and bias = 10.35 (Fig. 11b). For unbiased evaluation, we have evaluated the trained model performance over the test data set. We observed that the trained model performs well over the test data with R = 0.78, RMSE = 34.58, and bias = 19.89 (Fig. 11c).

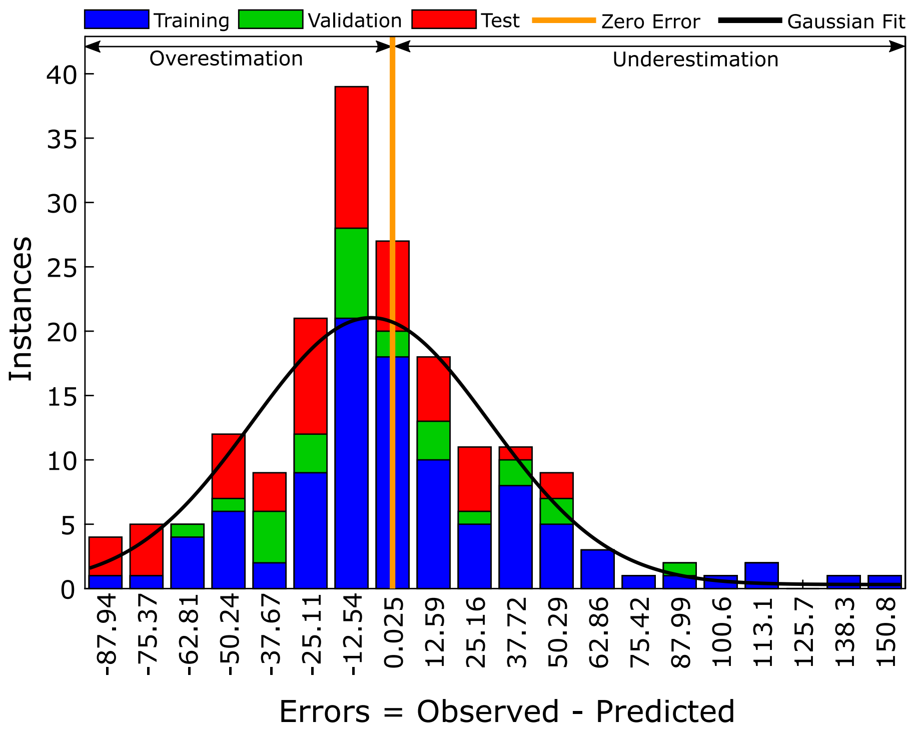

Fig. 9. Error analysis with error histogram with 20 bins for Gaussian sensor distribution case.

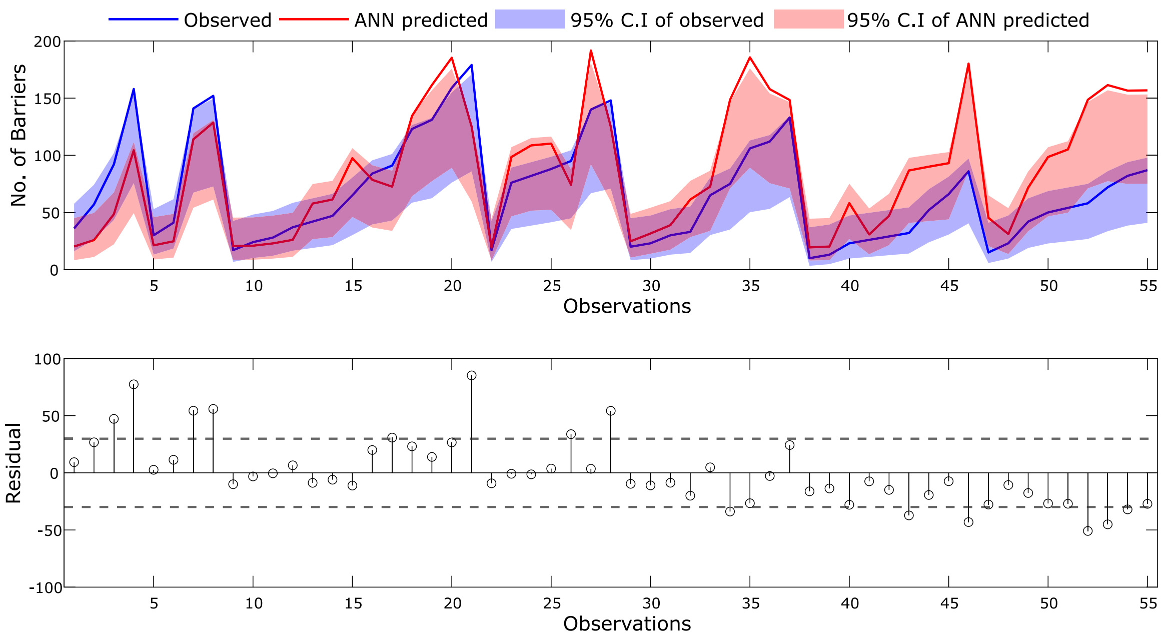

Fig. 10. Residual analysis of the feed-forward ANN output for Gaussian sensor distribution case.

Finally, we feed the combined data into the model to report the overall accuracy.

We found that the trained model also performs well over the complete data set with R = 0.79, RMSE = 48.36, and bias 9.55 (Fig. 11d).

Further, we perform the error analysis to understand the distribution of error in a uniform sensor distribution scenario. In doing so, we plotted the combined error histogram using twenty bins (Fig. 12). We observed a similar trend as for the Gaussian sensor distribution scenario despite the high error range. The total error ranges from −111.1 (leftmost bin) to 169 (rightmost bin) and follows a slightly right-skewed Gaussian distribution. Here also, the peak of the distribution lies near the zero error line indicating an accurate model. The total number of instances in the overestimated region is higher than the underestimated region. Due to this, a positive bias is present in the model.

Furthermore, we have performed the residual analysis and plotted the time series plot of observed–predicted values for the testing phase along with the corresponding residual plot (Fig. 13). Here, the residuals are well scattered and do not follow any particular path or pattern, indicating that the linear plot is a good fit.

7. Discussion

This study uses fully connected feed-forward ANN to predict the number of k-barriers for intrusion detection in WSNs. We trained two

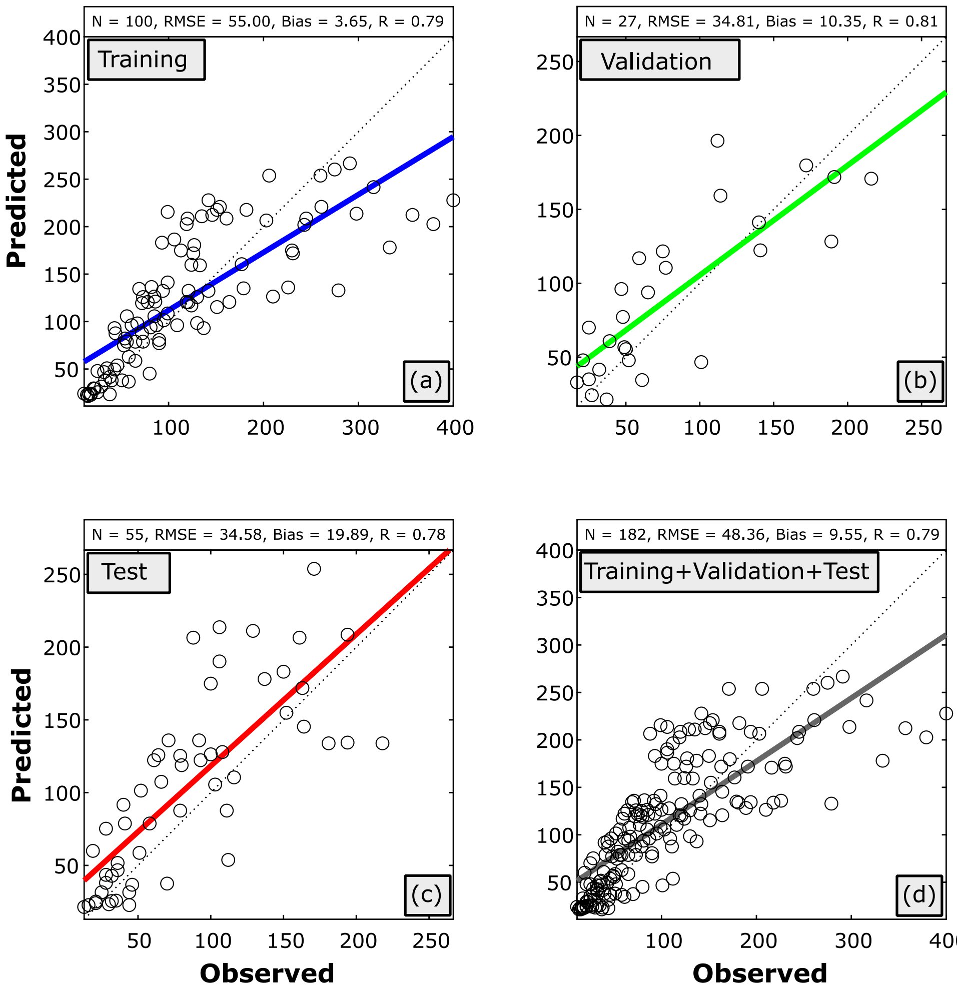

Fig. 11. Performance of the feed-forward ANN for uniform sensor distribution case (a) training accuracy, (b) validation accuracy, (c) testing accuracy, and (d) overall accuracy.

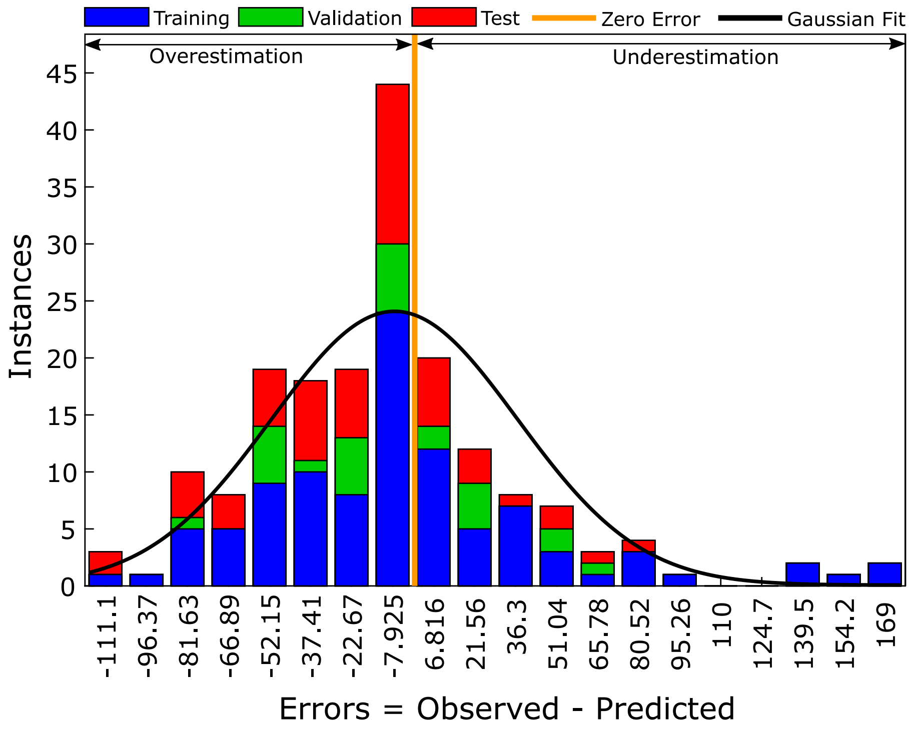

Fig. 12. Error analysis with error histogram with 20 bins for uniform sensor distribution case.

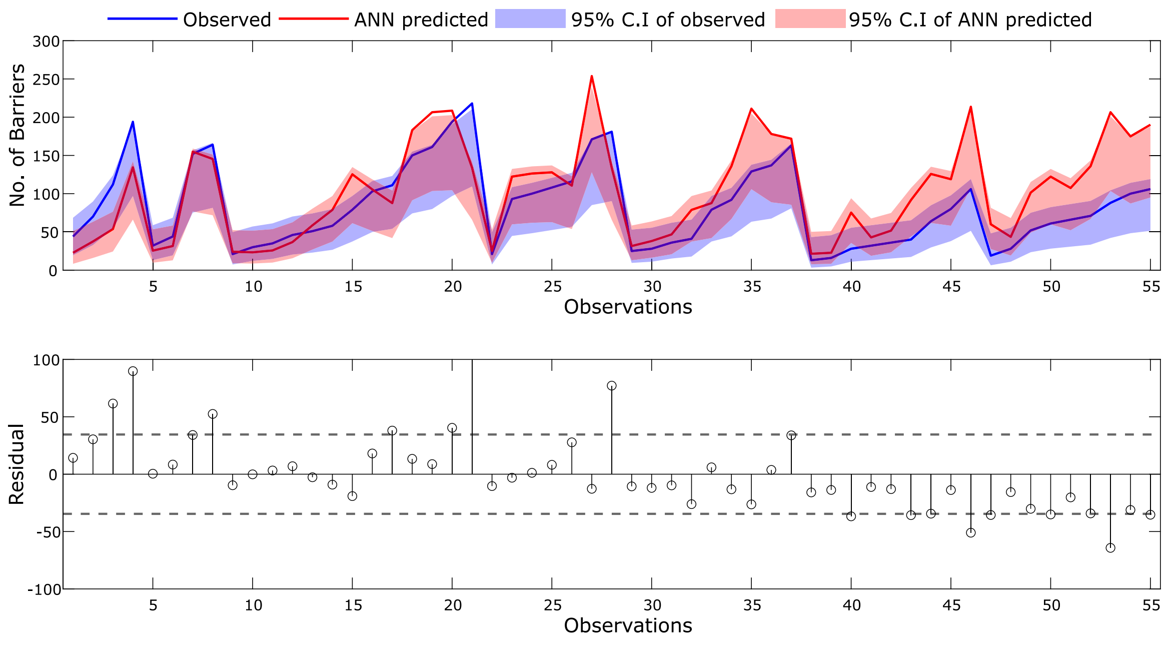

Fig. 13. Residual analysis of the feed-forward ANN output for uniform sensor distribution case.

separate feed-forward ANN models for Gaussian and uniform sensor distribution scenarios. We observed that the proposed architecture of a fully connected feed-forward ANN model gives promising results for both scenarios. Although the correlation coefficient value is nearly equal for both scenarios, the RMSE and bias value for the Gaussian sensor distribution scenario is better.

7.1. Comparing with different variant of feed-forward ANN

For an unbiased evaluation, we have generated different scenarios of the feed-forward ANN model based on the number of hidden layers used, ranging from shallow to deep feed-forward ANN model (Table 2). To do so, we have selected six different scenarios corresponding to 1, 2, 3, 4, 5, and 10 hidden layers. A feed-forward ANN model with more than ten hidden layers will result in high time complexity and eventually not an optimal solution for intrusion detection, which is a time-sensitive application. We have reported the training, validation, testing, and overall accuracy for all six scenarios. Based on the overall performance, we have categorised the performance of each scenario into poor, fair, and satisfactory. We found that the feed-forward ANN with 2, 3, and 10 layers resulted in satisfactory performance. Out of these three scenarios, feed-forward ANN with two layers (*i.e.,* 4:20:20:1) shows the best performance.

7.2. Comparison with the benchmark algorithms

Various other findings have been reported for high intrusion detection accuracy through the machine learning approach (Nancy et al., 2020; Safaldin, Otair, & Abualigah, 2021). Hence, any conclusion based on comparing different scenarios of a single algorithm may result in a biased conclusion. To ensure a fair evaluation of the proposed approach, we have compared the results of feed-forward ANN with the other benchmark algorithms using R, RMSE, and bias as the performance metrics. We have selected Radial Basis Neural Network (RBN), Exact Radial Basis Neural Network (ERBN), Recurrent Neural Network (RNN), LSTM, Boosting (Least-square boosting) Ensemble Learning (EL), Bagging EL (Random Forest), Binary Decision Tree (BDT), General Regression Neural Network (GRNN), Gaussian Process Regression (GPR), and Support Vector Regression (SVR) as potential benchmark algorithms because these algorithms are amongst best performing algorithms in the black-box and explainable based machine learning category (Belle & Papantonis, 2021; Elias et al., 2020; Lin et al., 2020; Lundberg et al., 2018; Mansor et al., 2020; Roscher, Bohn, Duarte, & Garcke, 2020; Singh, Gaurav, Rai and Beg, 2021; Zhang, Zhang, Ye, Fang, & Han, 2020). On comparing, we found that the feed-forward ANN outperforms all the benchmark algorithms in terms of R, RMSE and bias (Table 3). Other than the feed-forward ANN, SVR performs well and ranks second among the benchmark algorithms. Interestingly, we found that some benchmark algorithms show a strong correlation (*i.e.,* high R value); however, they produce biased results. The bias values are very high, indicating that these models strongly overestimate the response variable.

7.3. Comparison of the computational time complexity

Further, we have also compared the performance of the algorithms mentioned above in terms of computational efficiency. We have estimated and plotted the computational time complexity graph of all the three algorithms for both scenarios (Fig. 14). We observed that the feed-forward ANN algorithm exhibits the lowest, and the LSTM exhibits the highest computational time complexity. Apart from this, we have also plotted the time complexity of the Monte Carlo simulation for three different scenarios. We have estimated the time-complexity for sensors as 100, 200, and 300. We have kept the other features constant (area $\approx$ 5000, sensing range = 15, and transmission range = 30). We observed that the computation time-complexity of the Monte Carlo simulation increases with the number of sensors. This shows the efficacy and need of the proposed deep learning architecture to cut down the computational cost during the network setup time.

8. Conclusion

This study presented a fully connected feed-forward ANN architecture for the accurate mapping of the number of k-barriers for intrusion detection using WSNs. In doing so, we have trained two separate feed-forward ANN models for both Gaussian and uniform sensor distribution using four potential features extracted through simulations. While evaluating the feature importance and feature sensitivity for

Table 2
Comparison of the performance of different scenarios of feed-forward ANN.

Scenarios		Training			Validation			Testing			Overall			Performance
		R	RMSE	Bias	R	RMSE	Bias	R	RMSE	Bias	R	RMSE	Bias	
4:20:1 (Single layer)	Gaussian	0.40	60.66	−11.69	0.28	53.27	−17.26	0.55	61.46	19.05	0.44	59.69	−3.23	Fair
	Uniform	0.51	67.08	6.44	0.53	52.25	8.81	0.53	52.25	8.81	0.56	65.06	17.31	Fair
4:20:20:1 (Two layers)	Gaussian	0.82	43.38	−5.11	0.76	34.79	3.99	0.76	29.86	17.31	0.78	41.15	3.02	Satisfactory
	Uniform	0.79	55.00	3.65	0.81	34.81	10.35	0.78	34.58	19.89	0.79	48.36	9.55	Satisfactory
4:20:20:20:1 (Three layers)	Gaussian	0.81	42.94	−2.91	0.75	44.94	13.66	0.72	37.51	2.08	0.76	42.93	1.06	Satisfactory
	Uniform	0.82	49.54	14.69	0.80	48.62	18.31	0.66	47.90	22.24	0.77	50.17	17.51	Satisfactory
4:20:20:20:20:1 (Four layers)	Gaussian	0.44	59.93	−10.18	0.30	58.59	−20.99	0.54	57.61	−12.55	0.47	58.74	−12.50	Fair
	Uniform	0.08	78.32	−55.49	0.01	70.94	−43.85	0.12	81.12	−84.73	0.09	78.08	−62.60	Poor
4:20:20:20:20:20:1 (Five layers)	Gaussian	0.46	62.44	−24.53	0.28	67.11	−17.13	0.42	50.32	−11.17	0.43	60.00	−19.40	Fair
	Uniform	0.11	83.11	−102.25	0.06	84.13	−97.75	0.08	62.41	−70.25	0.07	78.22	−91.91	Poor
4:20:20: ... :20:20:1 (Ten layers)	Gaussian	0.64	48.35	25.09	0.82	46.20	9.78	0.75	45.01	22.25	0.69	47.72	21.96	Satisfactory
	Uniform	0.75	47.90	0.44	0.78	59.69	−3.47	0.70	59.17	5.20	0.73	53.23	1.30	Satisfactory

Table 3
Comparison with the benchmark algorithms.

Performance metrics	Methods																					
	Feed forward ANN		RBN		ERBN		RNN		LSTM		Boosting EL		Bagging EL (RF)		BDT		GRNN		GPR		SVR	
	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform	Gaussian	Uniform
R	0.78	0.79	0.68	0.34	0.69	0.68	0.97	0.96	0.08	0.07	0.90	0.89	0.99	0.99	0.93	0.93	0.89	0.88	0.98	0.98	0.63	0.66
RMSE	41.15	48.36	65.67	168.51	75.12	89.68	15.71	22.27	1.16	1.42	37.31	46.00	11.37	12.63	29.29	32.90	39.66	48.80	11.49	13.72	48.44	52.22
Bias	3.02	9.55	39.19	137.51	60.11	71.11	61.45	65.36	−17.10	−20.9	59.06	69.96	53.90	62.35	57.42	63.22	60.70	71.69	53.18	61.07	18.24	20.26

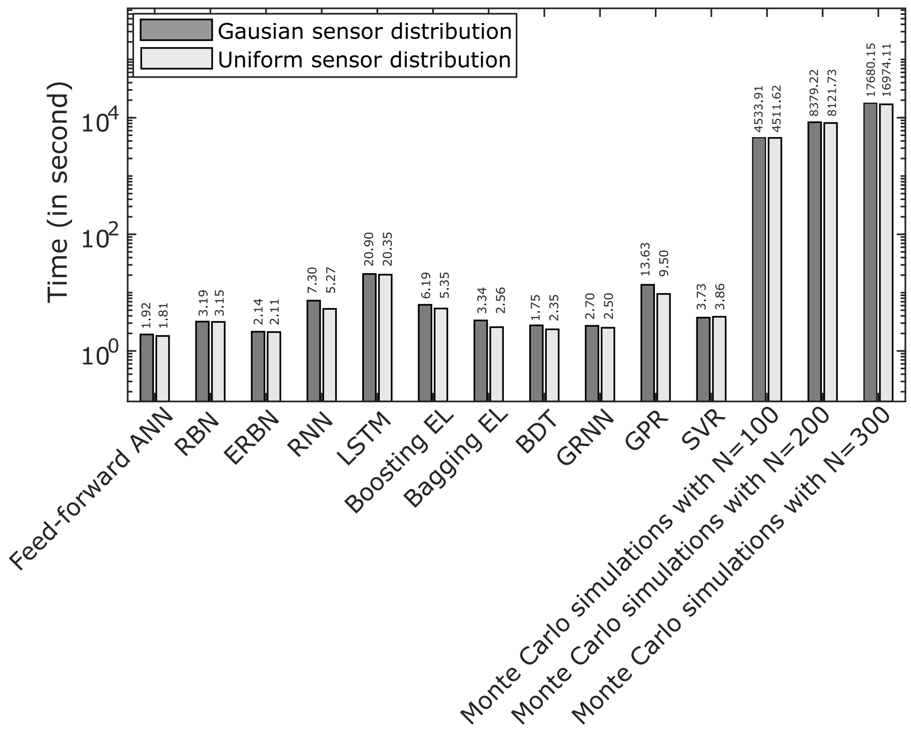

Fig. 14. Comparison of the computational time complexity of the feed-forward ANN with the benchmark algorithms with three different scenarios of Monte Carlo simulations. A log scale is used for the vertical axis.

Gaussian and uniform sensor distribution scenarios, we observed that sensing range, transmission range, and number of sensors are the most relevant features amongst all, which positively impact the predictand. In contrast, the area is the least important feature, and it is negatively related to the predictand. After training the models, we evaluated their performance over the unseen data and found that both models provide promising results.

Further, we have compared the performance of the feed-forward ANN with the benchmark algorithms. We found that the proposed feed-forward architecture outperforms the benchmark algorithms in terms of accuracy and computational time complexity.

This study is a step towards the accurate and time-efficient prediction of the number of k-barriers for intrusion detection using WSNs. Our methodology can be used to cut down the time and cost requirements during practical network setup.

CRediT authorship contribution statement

Abhilash Singh: Conceptualization, Methodology, Software, Data curation, Validation, Writing – original draft, Visualization, Investigation, Writing – review & editing. **J. Amutha:** Conceptualization, Methodology, Software, Data curation, Validation, Writing – original draft, Visualization, Writing – review & editing. **Jaiprakash Nagar:** Conceptualization, Methodology, Data curation, Visualization, Writing – original draft, Writing – review & editing. **Sandeep Sharma:** Methodology, Data curation, Visualization, Investigation, Writing – review & editing, Supervision, Project Administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

The datasets generated during and/or analysed during the current study can be downloaded from https://www.kaggle.com/datasets/abhilashdata/ffannid-intrusion-detection-in-wsnshere (Data) and the code can be downloaded from https://in.mathworks.com/matlabcentral/fileexchange/116670-a-deep-learning-approach-to-predict-the-number-of-k-barriers?s_tid=prof_contriblnkhere (Code) (accessed on 25 August 2022).

Acknowledgements

The authors would like to acknowledge IISER Bhopal, MITS Gwalior, IIT Kharagpur, and Gautam Buddha University Greater Noida for providing institutional support. They would like to thank to the editor and all the anonymous reviewers for providing helpful comments and suggestions.

References

Abbasi, J. S., Bashir, F., Qureshi, K. N., ul Islam, M. N., & Jeon, G. (2021). Deep learning-based feature extraction and optimizing pattern matching for intrusion detection using finite state machine. *Computers and Electrical Engineering*, *92*, Article 107094.

Amutha, J., Nagar, J., & Sharma, S. (2021). A distributed border surveillance (dbs) system for rectangular and circular region of interest with wireless sensor networks in shadowed environments. *Wireless Personal Communications*, *117*(3), 2135–2155.

Amutha, J., Sharma, S., & Nagar, J. (2020). WSN strategies based on sensors, deployment, sensing models, coverage and energy efficiency: Review, approaches and open issues. *Wireless Personal Communications*, *111*(2), 1089–1115.

Amutha, J., Sharma, S., & Sharma, S. K. (2021). Strategies based on various aspects of clustering in wireless sensor networks using classical, optimization and machine learning techniques: Review, taxonomy, research findings, challenges and future directions. *Computer Science Review*, *40*, Article 100376.

Arjun, D., Indukala, P., & Menon, K. U. (2019). PANCHENDRIYA: A multi-sensing framework through wireless sensor networks for advanced border surveillance and human intruder detection. In *2019 international conference on communication and electronics systems (ICCES)* (pp. 295–298). IEEE.

Aseeri, M., Ahmed, M., Shakib, M., Ghorbel, O., & Shaman, H. (2017). Detection of attacker and location in wireless sensor network as an application for border surveillance. *International Journal of Distributed Sensor Networks*, *13*(11), Article 1550147717740072.

Belle, V., & Papantonis, I. (2021). Principles and practice of explainable machine learning. *Frontiers in Big Data*, 39.

Benahmed, T., & Benahmed, K. (2019). Optimal barrier coverage for critical area surveillance using wireless sensor networks. *International Journal of Communication Systems*, *32*(10), Article e3955.

Duch, W., & Jankowski, N. (1999). Survey of neural transfer functions. *Neural Computing Surveys*, *2*(1), 163–212.

Elias, I., Rubio, J. d. J., Martinez, D. I., Vargas, T. M., Garcia, V., Mujica-Vargas, D., et al. (2020). Genetic algorithm with radial basis mapping network for the electricity consumption modeling. *Applied Sciences*, *10*(12), 4239.

Folino, F., Folino, G., Guarascio, M., Pisani, F. S., & Pontieri, L. (2021). On learning effective ensembles of deep neural networks for intrusion detection. *Information Fusion*, *72*, 48–69.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 1189–1232.

Gavel, S., Raghuvanshi, A. S., & Tiwari, S. (2021). Maximum correlation based mutual information scheme for intrusion detection in the data networks. *Expert Systems with Applications*, Article 116089.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993.

Huang, H., Gong, T., Zhang, R., Yang, L.-L., Zhang, J., & Xiao, F. (2018). Intrusion detection based on k-coverage in mobile sensor networks with empowered intruders. *IEEE Transactions on Vehicular Technology*, *67*(12), 12109–12123.

Kandris, D., Nakas, C., Vomvas, D., & Koulouras, G. (2020). Applications of wireless sensor networks: an up-to-date survey. *Applied System Innovation*, *3*(1), 14.

Karanja, S., & Badru, R. (2021). Development of a low cost wireless sensor network for surveillance along Kenya-Somalia border. Preprint.

Karthick, R., Prabaharan, A. M., & Selvaprasanth, P. (2019). Internet of things based high security border surveillance strategy. *Asian Journal of Applied Science and Technology (AJAST)*, *3*, 94–100.

Keung, G. Y., Li, B., & Zhang, Q. (2012). The intrusion detection in mobile sensor network. *IEEE/ACM Transactions on Networking*, *20*(4), 1152–1161.

Kotiyal, V., Singh, A., Sharma, S., Nagar, J., & Lee, C.-C. (2021). ECS-NL: An enhanced cuckoo search algorithm for node localisation in wireless sensor networks. *Sensors*, *21*(11), 3576.

Laranjeira, L. A., & Rodrigues, G. N. (2014). Border effect analysis for reliability assurance and continuous connectivity of wireless sensor networks in the presence of sensor failures. *IEEE Transactions on Wireless Communication*, *13*(8), 4232–4246.

Lee, S.-W., Mohammadi, M., Rashidi, S., Rahmani, A. M., Masdari, M., Hosseinzadeh, M., et al. (2021). Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review. *Journal of Network and Computer Applications*, Article 103111.

Lin, H., Dai, Q., Zheng, L., Hong, H., Deng, W., & Wu, F. (2020). Radial basis function artificial neural network able to accurately predict disinfection by-product levels in tap water: Taking haloacetic acids as a case study. *Chemosphere*, *248*, Article 125999.

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., et al. (2018). Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, *2*(10), 749–760.

Mansor, M., Mohd Jamaludin, S. Z., Mohd Kasihmuddin, M. S., Alzaeemi, S. A., Md Basir, M. F., Sathasivam, S., et al. (2020). Systematic Boolean satisfiability programming in radial basis function neural network. *Processes*, *8*(2), 214.

Mathew, J., Griffin, J., Alamaniotis, M., Kanarachos, S., & Fitzpatrick, M. E. (2018). Prediction of welding residual stresses using machine learning: comparison between neural networks and neuro-fuzzy systems. *Applied Soft Computing*, *70*, 131–146.

Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys & Tutorials*, *21*(1), 686–728.

Moldovan, A., Caţaron, A., & Andonie, R. (2020). Learning in feedforward neural networks accelerated by transfer entropy. *Entropy*, *22*(1), 102.

Mostafaei, H., Chowdhury, M. U., & Obaidat, M. S. (2018). Border surveillance with WSN systems in a distributed manner. *IEEE Systems Journal*, *12*(4), 3703–3712.

Nagar, J., Chaturvedi, S. K., & Soh, S. (2020). An analytical model to estimate the performance metrics of a finite multihop network deployed in a rectangular region. *Journal of Network and Computer Applications*, *149*, Article 102466.

Nagar, J., & Sharma, S. (2018). K-barrier coverage-based intrusion detection for wireless sensor networks. In *Cyber security* (pp. 373–385). Springer.

Nancy, P., Muthurajkumar, S., Ganapathy, S., Kumar, S. S., Selvi, M., & Arputharaj, K. (2020). Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks. *IET Communications*, *14*(5), 888–895.

Novickis, R., Justs, D. J., Ozols, K., & Greitāns, M. (2020). An approach of feed-forward neural network throughput-optimized implementation in FPGA. *Electronics*, *9*(12), 2193.

Pektaş, A., & Acarman, T. (2019). A deep learning method to detect network intrusion through flow-based features. *International Journal of Network Management*, *29*(3), Article e2050.

Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable machine learning for scientific insights and discoveries. *IEEE Access*, *8*, 42200–42216.

Safaldin, M., Otair, M., & 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), 1559–1576.

Saraereh, O. A., Ali, A., Al-Tarawneh, L., & Khan, I. (2021). A robust approach for barrier-reinforcing in wireless sensor networks. *Journal of Parallel and Distributed Computing*, *149*, 186–192.

Sharma, A., & Chauhan, S. (2020). Sensor fusion for distributed detection of mobile intruders in surveillance wireless sensor networks. *IEEE Sensors Journal*, *20*(24), 15224–15231.

Sharma, S., & Nagar, J. (2020). Intrusion detection in mobile sensor networks: A case study for different intrusion paths. *Wireless Personal Communications*, 1–21.

Si, P., Ma, J., Tao, F., Fu, Z., & Shu, L. (2020). Energy-efficient barrier coverage with probabilistic sensors in wireless sensor networks. *IEEE Sensors Journal*, *20*(10), 5624–5633.

Singh, A., Amutha, J., Nagar, J., Sharma, S., & Lee, C.-C. (2022a). AutoML-ID: automated machine learning model for intrusion detection using wireless sensor network. *Scientific Reports*, *12*(1), 1–14.

Singh, A., Amutha, J., Nagar, J., Sharma, S., & Lee, C.-C. (2022b). Lt-fs-id: Log-transformed feature learning and feature-scaling-based machine learning algorithms to predict the k-barriers for intrusion detection using wireless sensor network. *Sensors*, *22*(3), 1070.

Singh, A., Gaurav, K., Rai, A. K., & Beg, Z. (2021). Machine learning to estimate surface roughness from satellite images. *Remote Sensing*, *13*(19), 3794.

Singh, A., Kotiyal, V., Sharma, S., Nagar, J., & Lee, C.-C. (2020). A machine learning approach to predict the average localization error with applications to wireless sensor networks. *IEEE Access*, *8*, 208253–208263.

Singh, A., Nagar, J., Sharma, S., & Kotiyal, V. (2021). A Gaussian process regression approach to predict the k-barrier coverage probability for intrusion detection in wireless sensor networks. *Expert Systems with Applications*, *172*, Article 114603.

Singh, A., Sharma, S., & Singh, J. (2021). Nature-inspired algorithms for wireless sensor networks: A comprehensive survey. *Computer Science Review*, *39*, Article 100342.

Singh, A., Sharma, S., Singh, J., & Kumar, R. (2019). Mathematical modelling for reducing the sensing of redundant information in WSNs based on biologically inspired techniques. *Journal of Intelligent & Fuzzy Systems*, *37*(5), 6829–6839.

Sohi, S. M., Seifert, J.-P., & Ganji, F. (2021). RNNIDS: Enhancing network intrusion detection systems through deep learning. *Computers & Security*, *102*, Article 102151.

Sood, T., Prakash, S., Sharma, S., Singh, A., & Choubey, H. (2022). Intrusion detection system in wireless sensor network using conditional generative adversarial network. *Wireless Personal Communications*, 1–21.

Torres-Barrán, A., Alonso, A., & Dorronsoro, J. R. (2019). Regression tree ensembles for wind energy and solar radiation prediction. *Neurocomputing*, *326*, 151–160.

Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525–41550.

Wang, D., Xie, B., & Agrawal, D. P. (2008). Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE Transactions on Mobile Computing*, *7*(12), 1444–1458.

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, *5*, 21954–21961.

Zhang, D., Zhang, N., Ye, N., Fang, J., & Han, X. (2020). Hybrid learning algorithm of radial basis function networks for reliability analysis. *IEEE Transactions on Reliability*.