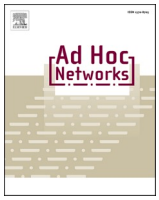
Ad Hoc Networks 144 (2023) 103139

Contents lists available at ScienceDirect

Ad Hoc Networks

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



Machine learning approaches for underwater sensor network parameter prediction

Osman Gokhan Uyan *, Ayhan Akbas, Vehbi Cagri Gungor

Abdullah Gul University, Department of Computer Engineering, Kayseri, Turkey

ARTICLE INFO

Keywords:
Data fragmentation
Encryption
Machine learning
Reliability
Security
Underwater acoustic sensor networks

ABSTRACT

Underwater Acoustic Sensor Networks (UASNs) have recently attracted scientists due to its wide range of real-world applications. However, there are design challenges in UASNs, such as limited network lifetime and low communication reliability provoked by the constrained battery supply of sensors and harsh channel conditions in the underwater environments. To meet communication reliability requirements, packet-duplication and multi-path routing algorithms have been recommended in the literature. Furthermore, underwater sensors may convey sensitive data, which must be masked to avoid eavesdropping attempts. To improve network security, cryptographic encryption is the most widely used method. Nevertheless, data encryption needs computations to cipher the data, which consumes extra energy, resulting in a cutback in the life span of the network. To address these challenges, an optimization model has been proposed to evaluate the impacts of multi-path routing, packet duplication, encryption, and data fragmentation on the lifetime of the UASNs. However, the solution time of the proposed optimization model is quite high, and sometimes it cannot come up with feasible solutions. To this end, in this study, different regression and neural network methods have been proposed to predict network parameters and energy consumptions of underwater nodes as supplementary methods to optimization models. Performance evaluations show that the proposed methods yield remarkably accurate predictions and can be used for energy consumption prediction in UASNs.

1. Introduction

The Earth surface is encircled by water with a roughly scale of 70%. The underwater realm has been majorly undiscovered until recently. Recent advances in Underwater Acoustic Sensor Networks (UASNs) enable opportunities to explore this field for academic studies and commercial applications. In UASNs, several nodes are placed in a three-dimensional space to trail the underwater environment. Underwater applications can be utilized in scientific (i.e., exploring the ecosystem), civil (i.e., overseeing fuel pipes) or military (i.e., detecting breaches or attacks) operations. The UASNs employ wireless communication, and sensors broadcast acoustic signals to send their sensed data to the base node located on the surface. Generally, there are several intermediary nodes serving as relays that convey the sensing data of other nodes to the base.

The sensors in UASNs run with the energy stored in their batteries, which have limited capacities. Hence, power requirements caused by data transmission, reception, and other computational operations in harsh underwater environments delimit network lifetime, and energy efficiency is an important design challenge for UASNs. Moreover, UASN operations expect a large amount of data transmission and reception, and communication reliability is another important design issue. Nevertheless, meeting communication reliability requirements has a negative effect on the energy expenditure of the sensors.

The security of the communications among underwater nodes is another essential topic. The nodes might generate private sensing data and the communications among them must be secured, since they remain in an underwater space and are vulnerable to several types of attacks. One of these attacks is eavesdropping, where an adversary listens to the broadcast data to snatch the information without being detected [1]. To prevent the classified data from being disclosed, the most acclaimed process is ciphering it before transmission, so that the data becomes futile if it cannot be deciphered. While encryption is a robust method for securing data during communication, the data must still be processed with additional computing, which requires additional energy consumption. This situation is a drawback for network lifetime

Abbreviations: UASN, Underwater Acoustic Sensor Network.

* Corresponding author.
E-mail addresses: gokhan.uyan@gmail.com (O.G. Uyan), ayhan.akbas@agu.edu.tr (A. Akbas), cagri.gungor@agu.edu.tr (V.C. Gungor).

https://doi.org/10.1016/j.adhoc.2023.103139
Received 22 November 2022; Received in revised form 9 February 2023; Accepted 5 March 2023
Available online 8 March 2023

because of the limited battery capacity of the sensor nodes. Another candidate method against eavesdropping attacks is data fragmentation, where the data to be transmitted is divided into pieces and each piece is sent over different paths to the sink node [2].

In our previous study, we proposed a multi-path routing method along with packet duplication to deal with the reliability of the network [3]. In addition, we developed a mixed-integer programming (MIP) framework to study the impacts of multi-path routing, packet duplication, encryption, and data fragmentation on the network lifetime. This MIP framework aims at maximizing the network lifetime by minimizing energy consumption of the highest energy depleting sensors. Moreover, AES and Twofish encryption methods have been adopted to suggest a balance against the trade-off between data protection gained via encryption and lifetime of the network.

Even though an MIP framework comes up with optimal solutions, running optimizations means high computational complexity ensuing from the nature of MIPs, and even in some cases optimizations cannot provide feasible solutions. With this motivation, in this paper, machine learning (ML) models have been proposed to predict network parameters and the energy consumptions of underwater nodes as supplementary methods to optimization models. This way, the computation time has been significantly reduced without running computationally intensive optimizations. Besides requiring heavy calculations, optimizations sometimes cannot produce feasible solutions or finish in a reasonable time. By using machine learning models, we can avoid the high computational burden of optimizations. Specifically, different regression and neural network-based models have been proposed to predict the energy consumptions of underwater nodes and thus, to identify if ML is practical to be used in our networking scenarios. The main contributions of this study can be summarized as follows:

- To the best of our knowledge, this paper is the first study concentrating on network parameter prediction for UASNs using Machine Learning (ML) approaches, including regression and neural network-based models.
- Comparative performance results show that the regression methods (i.e., Decision Trees, Gradient Boost, and Random Forest) and neural network-based methods (ANN and CNN) can make accurate predictions.
- It is shown that, even if optimizations cannot return feasible solutions, parameter prediction via ML can be used for parameter prediction, while designing underwater network.

The remainder of this paper is presented as follows: Section II gives a background on studies in the literature. Section III presents the network scenario. Section IV provides concise descriptions about the analyzed ML algorithms and their implementations. Section V provides the performance analysis of the implemented ML algorithms. Eventually, Section VI concludes the paper.

2. Related work

The sensors in an UASN use the energy stored in their batteries to perform their operations. When designing a sensor network architecture, it is important to analyze the energy spent on different sensor node processes, such as transmission, reception, and encryption. UASNs usually transmit their sensing data in a multi-hop manner, where neighboring nodes act as relays, carrying the data to the sink node. To examine the energy consumption of the nodes, researchers generally use optimization and analyze the generated data. On the other hand, instead of running bulky optimizations for different network designs, machine learning methods can be used to make predictions about different network parameters.

There are various studies in the literature some of which deal with parameter prediction using ML for terrestrial WSNs, while more recent studies are conducted for UASNs. If we consider characteristics of WSNs and UASNs, there exists apparent differences if operation such as broadcast medium, channel model, bandwidth, mobility etc. However, if parameter prediction with ML is discussed, in general there is a dataset fed as input to train a model, and even though the attributes of the data are different for WSNs and UASNs, the final aim is similar as ML models are used for predicting different network parameters. The following three studies are conducted for terrestrial WSNs.

Akbas et al. have used machine learning in classical wireless sensor networks (WSN) for parameter prediction [4]. Instead of using optimizations to find optimal operation configuration of a WSN, they used neural networks to reduce the computational cost. They focused on prediction network parameters such as lifetime, transmission power, and distance between the nodes. To train the neural network, they have used data generated with optimizations. The generated model makes predictions about parameters with acceptable errors.

Yilmaz et al. suggested a machine learning model based on deep neural network (DNN) to determine the lifetime of a WSN [5]. The proposed model was able to make good predictions about networks with high number of nodes even if it was trained with a dataset generated with small number of nodes.

Akbas et al. uses single-based and stack ensemble-based machine learning models to make parameter predictions in WSNs [6]. For single-based models, their test results show that Adaboost makes better estimations when compared to Elastic Net and SVR. Moreover, stack ensemble-based models make the best predictions for WSN parameters when compared to single-based models.

Some recent studies are conducted suggesting usage of ML for parameter prediction in UASNs. In [7], Chen et al. proposed using Logistic Regression algorithm to predict channel conditions according to the measured BER of the channel prior to data transmission. Their test results indicate that packet loss rates and energy consumption of the network can be decreased by the suggested method.

Kalaiarasu et al. suggested using Logistic Regression to make performance predict for an underwater sensor network based on different parameters like wind speed, tide, and modem features [8]. The results given in the paper present that their model can predict network performance with good accuracy.

Alamgir et al. proposed using Boosted Regression Tree to predict a suitable link adaption method for encoding and modulation [9]. Suggested model makes predictions according to data rate, SNR and BER of the broadcast paths. Their results show that the generated model classifies schemes with an accuracy of 99%.

Eldesouky et al. examined four machine learning algorithms for handover prediction depending on water flow speed and direction of the communication in underwater wireless sensor networks [10]. The given results show that the generated models make predictions with 95% accuracy.

In [11], Liu et al. suggested using deep neural networks to make channel state information predictions for underwater OFDMA system. Their results indicate that proposed model makes better predictions than the compared solutions in the study.

There are recent studies that offer using ML for predicting different parameters for underwater sensor networks. To the best of our knowledge, a study has not been conducted that uses machine learning and neural network algorithms to predict energy consumption values including reliability and security concerns in UASNs in the literature. In general, setting up network parameters based on optimization frameworks is a widely studied and efficient method. However, due to the complexity of the calculations, optimizations generally take too much time and sometimes they cannot produce any feasible results. To avoid running complex optimizations, we suggest using machine learning methods, which can be used for predicting parameters with low errors in a short amount of time. In this study, different ML methods have been analyzed to find adequate candidates to be used for energy consumption and network parameter prediction instead of running computationally expensive optimizations.

3. Network scenario

The network scheme presented in this study is formed by a sink node at the water surface and several sensor nodes placed randomly in three-dimensional underwater area. Throughout the network, every sensor senses the environment and produces data, which it transmits to the sink. The sink node does not produce or transmit any data packets and it does not have any constraints on energy. The data generated by sensors is transmitted to the sink in a multi-hop manner. During transmissions, we assume that the sensors adopt a TDMA based approach, so that every node transmits data during the time slot allotted to it, thus there is no interference between the communications.

As mentioned in the first section, encryption is a widely used technique to keep transmitted data secure. Additionally, data security can be improved with data fragmentation in addition to encryption. In contrast to sending an entire data over a single path, fragmenting the data, and sending each piece over multiple paths to the sink is another remedy against eavesdropping attacks. After receiving all the pieces, the sink node will first defragment and then decrypt the data to access the information. As a node transmits each piece of data through diverse paths, an attacker is compelled to gather every piece before capturing the original data. Moreover, to maintain a reliability threshold, a node might send duplicate packets, and it is easier for an attacker to capture any of these duplicate packets. Here, data fragmentation appears to be a decent method against the security risk of packet duplication. An important point is that data fragmentation not only causes the attackers deplete high energy, but also places an additional energy load on the sensor nodes. When the data is sliced and each piece is conveyed over different lines, all the data cannot be sent by employing optimal lines. Some part of the data needs to be sent over suboptimal paths, which will cause some increase in the energy consumption of the network.

The MIP framework proposed in our previous study [3] is given in Fig. 1. In the model, the network aims at satisfying the network success rate (NSR), which is the reliability threshold defining the smallest necessity of a transmitted packet to arrive at the sink favorably. In the sharp underwater channel conditions, if the potential of transmission success over a single way is below this threshold, then the data packets must be copied, and transmitted over the same way several times, or sent through diverse ways at the same time.

The choice of duplication number and which paths to use is made according to a number named packet success rate (PSR). For a given packet length of l and bit error rate (BER), PSR is computed as:

$$PSR = (1 - BER)^{l} \tag{17}$$

If a source sends data over a single path, then PSR for the selected path is equivalent to the NSR. But if packet duplication will be employed, then NSR for the packet going out from a source is computed based on PSRs of all the arcs, where each copy is sent over:

$$NSR = 1 - \prod_{i \in D} (1 - PSR_i) \tag{18}$$

where D is the group of copied packets and PSR_i is the PSR of i^{th} arc that a copy is transmitted through. For simplicity, BER is assumed to be the same for all sensors so that PSR is also same for all arcs. This way, sending all copy packets over same arc or over different arcs does not affect computation of NSR. In this case the NSR is computed as:

$$NSR = 1 - (1 - PSR)^{n} \tag{19}$$

And from Eq. (19), the smallest number of copies for achieving NSR is computed as:

$$\textit{minimize } \theta \tag{1}$$

$$\textit{subject to:}$$

$$\theta \geq t_p \sum_{k=1}^{n} P_k \left(\sum_{\ell \in \delta^+(j)} P_{r,\ell} y_{\ell k} + \sum_{\ell \in \delta^-(j)} P_{t,\ell} x_{\ell k} \right) + \sum_{\ell \in \delta^+(j)} E_k x_{\ell k} \ , \forall j \in N \backslash \{0\}, \tag{2}$$

$$M \sum_{\ell \in \delta^+(j)} y_{\ell k} \geq \sum_{\ell \in \delta^-(j)} x_{\ell k} \ , \forall j \in N \backslash \{0\}, \forall k \in N \backslash \{0\}, j \neq k, \tag{3}$$

$$M \, u_{ik} \geq x_{ik} \ , \forall i \in A, \forall k \in N \backslash \{0\}, \tag{4}$$

$$\sum_{\ell \in \delta^-(j)} x_{\ell k} \leq M r_a, \forall j \in A, \forall k \in N \backslash \{0\}, \tag{5}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \leq \sum_{\ell \in \delta^-(j)} x_{\ell k} \ , \forall j \in N \backslash \{0\}, \forall k \in N \backslash \{0\}, j \neq k, \tag{6}$$

$$\sum_{\ell \in \delta^-(j)} x_{\ell k} \geq 1, \forall k \in N \backslash \{0\}, \tag{7}$$

$$x_{ik} = M y_{ik}, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{8}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \geq r_a, \forall k \in N \backslash \{0\}, \tag{9}$$

$$\sum_{k=1}^{n} P_k \left(\sum_{\ell \in \delta^+(j)} y_{\ell k} + \sum_{\ell \in \delta^-(j)} x_{\ell k} \right) \leq t_r, \forall j \in N \backslash \{0\}, \tag{10}$$

$$v_{t_i k} - v_{h_i k} + n u_{ik} \leq n - 1, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{11}$$

$$x_{ik} \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{12}$$

$$y_{ik} \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{13}$$

$$u_{ik} \in \{0,1\}, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{14}$$

$$\theta \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{15}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \leq L_{node}, \forall k \in N \backslash \{0\}. \tag{16}$$

Fig. 1. MIP model.

$$M = \frac{\log(1 - NSR)}{\log(1 - PSR)} \tag{20}$$

In this setup, a relay node can decide to copy the data transmitted by the source node to achieve the NSR, before conveying it to the sink. The choice of using diverse ways or single one is essential for distributing energy consumption between the nodes and the optimization model is formulated to allow this balancing. In every transmission round, the model strives to minimize the highest energy depleted by the sensors. By minimizing the energy consumption of the sensors at each round, network lifetime maximization is achieved. A transmission round is defined as the necessary time for every packet produced by every node to arrive at the sink.

In the framework, θ (1) is the objective function that minimizes the energy consumption. Eq. (2) defines the objective function and ensures that θ is at least equal to the energy depletion of a node except sink, including the energy spent for broadcasting, receiving, and encrypting data at each round. Statement (3) ensures that a relay cannot transmit data coming from another node if it does not receive the data. Constraints (4) and (14) assure a traffic on a path to be occupied. Eq. (5) adjusts packet duplication such that it compensates failures in the next path. Eq. (6) tolerates packet replication and approves that sum of outgoing data of a relay is larger than or equal to its total incoming data. Constraint (7) affirms that a node sends one or more packets, which allows packet duplication. Eq. (8) is given to manage packet losses, and Eq. (9) ensures that NSR is obtained. Constraint (10) defines total packets that can be transmitted or received in a single round (t_r) by a node. Statement (11) avoids cycles during transmissions. Eqs. (12), (13), and (15) are given to provide that the decision variables are positive, and (14) provides that u_{ik} are binary. Lastly, constraint (16) is added for packet fragmentation.

In the optimizations, Node Numbers represent the amount of sensor nodes deployed in 3D underwater space forming the UASN. NSR is used as an abbreviation for Network Success Rate, which represents the reliability level of the network. An NSR of 0.8 implies that 80% of the packages needs to be delivered successfully to the sink node. L_{node} is used for defining the number of fragments of the packets. Total of all transmitted slices of a node's data is restricted by L_{node}, so that only one piece of data of a node can be sent to the next node. $L_{node} = 0.25$ indicates that a relay should receive only 25% of the data of the source node, which means the sender must slice its data into four pieces. Likewise, $L_{node} = 1$ means that there is no fragmentation, and a packet is submitted by the source node without slicing. BER is the Bit Error Rate of the channel, which affects the success of transmission in wireless channel. If BER is high, the nodes must transmit data using more transmission power, which requires more energy and might change the lifetime of the nodes. Packet Size describes the size of data packets in kilobytes, which are used by all nodes throughout the network. The size of the packet might affect the successful transmission and it is an important parameter in network design. For maintaining security against eavesdropping attacks, using encryption was suggested in our previous study. Four types of encryption combinations are used in the study. These are, none of the nodes use encryption, all the nodes use Twofish algorithm, all the nodes use AES algorithm or the nodes closer to sink use Twofish and nodes far from the sink use AES algorithm as mentioned in [3].

In the network, transmission power selected for the nodes is quite important as it affects the energy consumption considerably. In the sharp underwater environment, transmission powers are eminently related to the scope of transmission. In [12], Felemban et al. uses necessary intensity (I_t) in 1 meter gap to calculate the necessary transmission power:

$$P_t = I_t \times 2\pi \times 1m \times h \tag{21}$$

And from (21), reference intensity is computed as:

$$I_t = 10^{\frac{SL(d,f)}{10}} \times I_0 \tag{22}$$

To find the Source Level (*SL*), Signal to Noise Ratio (*SNR*), attenuation (*A*) and noise (*N*) must be known.

$$SL(d,f) = A(d,f) + N(f) + SNR \tag{23}$$

SNR, A, and *N* are computed according to the formulas:

$$A(d,f) = \kappa \log(d) + \alpha(f) \times d \times 10^{-3} \tag{24}$$

$$N(f) = N_t(f) + N_s(f) + N_{th}(f) + N_w(f) \tag{25}$$

$$SNR = 10^{\frac{SNR(d,f)}{10}} \tag{26}$$

(26) gives SNR in decibel (dB). Non-dB *SNR* is calculated using energy per bit to noise power spectral density ratio (E_b/N_0), data rate (*R*) and noise bandwidth (B_N):

$$SNR(d,f) = \frac{E_b}{N_0} \times \frac{R}{B_N} \tag{27}$$

Where and E_b/N_0 is calculated by:

$$BER = \frac{3}{8} erfc\left(\sqrt{\frac{4}{10} \times \frac{E_b}{N_0}}\right) \tag{28}$$

For reception operations, the power for the receiving node is taken as 10% of the transmission power of the source node. The origin of Eqs. (21) to (28) are the measurements taken during field tests in the underwater environment and classical signal processing theories.

4. Machine learning algorithms

In this study, the main objective is to employ various machine learning algorithms to make predictions about the energy consumption values instead of running bulky optimizations. The main reason behind this idea is that optimizations can provide us with accurate numbers about energy consumption of the UASN, however they take a long time, and in some instances, they cannot find practical solutions or even finish in an acceptable time frame. Instead of optimizations, if we can build adequate regression models using the data we have collected, we can make nearly accurate predictions about deployed network. This way, after producing a regression model, we can run simple mathematical functions to predict data instead of running bulky optimizations, with a sacrifice of accuracy in the range of 2 to 5 percent. Note that to generate a model, a ML algorithm needs to be fed with a dataset to train the model and test its performance. Input data can be collected either via running field tests by placing sensor nodes in a testbed and recording the output of different network configurations or running network simulations on a computer. Thus, if there is a dataset in hand, running optimizations is not mandatory and ML methods can be used as a standalone option. Albeit, since we do not have a testbed setup, we have used network simulations in this study.

Regression is a statistical method used in many disciplines, that tries to discover the the relationship between a dependent variable and several independent variables. A formula, as we name it a regression model, is generated by the regression method which tries to express the relationship between the variables best. The general form of a regression function is:

$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_nX_n + u \tag{29}$$

where, Y is the dependent variable that the function tries to predict, X is the independent variable used for making predictions, a is the intercept, b is the slope and u is the remainder.

Regression receives a group of variables as inputs, that are related to Y, and seeks a mathematical relationship between them. The mathematical relationship is normally in the form of a straight line or hyperplane that fits all the individual data points approximately.

To generate a model, an ML method must be supplied with pre-

generated data, so that it can try to find the relations between the variables. To gather the data, we started by running simulations with different parameters. We have run the simulations according to the MIP model that we have built in [3]. We have prepared our parameters using MATLAB [13] and run our optimization algorithm in CPLEX Solver [14] with different combinations of parameters to generate a dataset to train and test the regression methods. We have run the simulations with the following combinations of different parameters:

- Node Numbers: {10, 15, 20, 25, 30, 35, 40}
- NSR: {0.5, 0.6, 0.7, 0.8, 0.9}
- L_{node}: {1, 0.5, 0.33, 0.25, 0.2}
- BER: {1e-5 1e-4 1e-3 1e-2 1e-1}
- Packet Size: {256 512 1024 2048 4096 8192 10000}
- Encryption Type: {No encryption, Twofish, AES, Mixed}

The number of combinations of the variables is 24500. To be adaptive, the data used in the ML methods are generated by running optimizations with different random topologies. One of the most important factors about energy consumption is distance between the nodes as transmission energies are calculated according to the distances between the nodes to complete successfully. By using random topologies, we generate data that is adaptive to different underwater conditions. We have run the simulations for each combination 100 times with different random topologies of the nodes and we took their average energy consumption as the final value. At the end of the simulations, we managed to collect 5880 rows of data while the rest of the simulations did not come up with a feasible solution.

Table 1 presents runtimes for the optimizations that were held with combinations of the parameters above. From the table, it can be seen that increasing node number increases the runtime exponentially, since the solver makes calculations for every candidate node on a path in multi-hop fashion and high number of nodes increases the number of candidate nodes for a source node to send its packet. Total runtime of optimizations lasts more than 2 months. Moreover, we see that only 24% of the simulations produce feasible results. This situation supports the idea of using regression methods for predicting data, both to avoid the computational burden of the optimizations and to be able to make predictions where optimizations fail. Fig. 2 shows the flow for preparing the data to train and test machine learning methods.

After collecting the data, we have used Python 3.9 [15] with Scikit-learn [16] to produce regression models and Keras [17] running on Tensorflow [18] to model neural networks for the prediction of energy consumption values. Before starting to build models, we used Pandas package [19] to process and Pandas-profiling package [20] to analyze our data. We employed Jupyter Notebook [21] as the development platform. We have used Seaborn [22] package to plot our results. The results of data analysis generated using pandas-profiling package are shown in Table 2.

Fig. 3 illustrates the Spearman's correlations [23] between the variables. The Spearman's rank correlation coefficient (ρ) gives a description of monotonic correlation between given variables. To compute ρ for two variables, the covariance of the rank of these variables is divided by the multiplication of their standard deviations. As it can be seen from Fig. 3, Energy is highly correlated with Encryption Type and it is strongly correlated with Bit Error Rate (BER) and Packet Size.

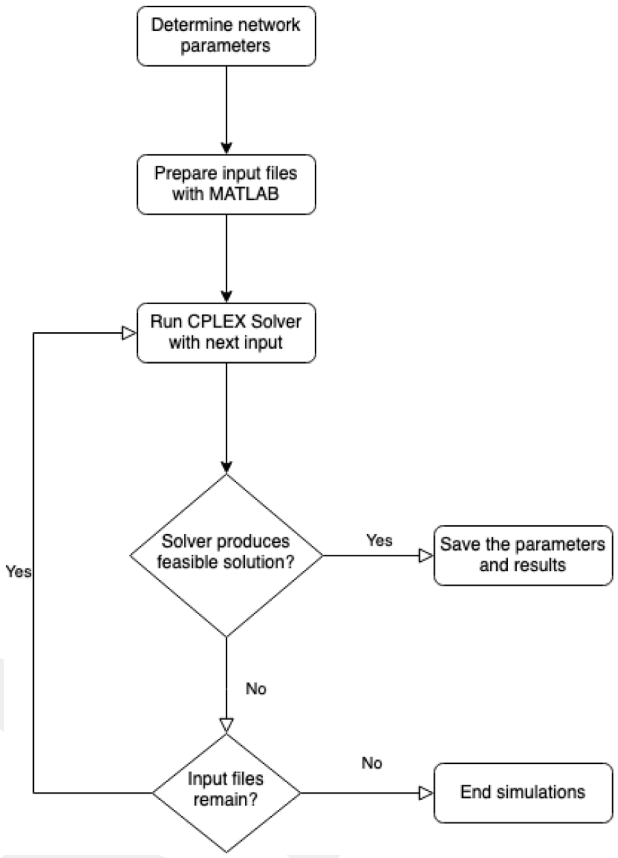


Fig. 2. Data preparation flowchart.

Table 2
Dataset statistics.

Dataset statistics	
Number of variables	7
Number of observations	5880
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	
Duplicate rows (%)	0.0%
Total size in memory	321.7 KiB
Average record size in memory	56.0 B

Fig. 4 shows a pair-plot of the variables demonstrating the relations between them. Pair-plot visualization is widely used for data analysis to discover the best set of features that define a relationship between two variables. The plots are given in matrix format where the row name indicates x axis and column name indicates y axis. The subplots on the main-diagonal depict the distributions for each variable. Note that the distributions shown in Fig. 4 does not indicate the correlations between the dependent variable and the independent variables. Instead, it shows

Table 1
Runtimes for optimizations.

	Number of nodes							
	10	15	20	25	30	35	40	Total
Average time for 1 parameter configuration (sec)	0.11	0.49	1.12	2.11	6.02	14.5	38.2	62.5
Average time for running 100 times for random topologies (min)	0.18	0.82	1.86	3.52	10.1	24.1	63.7	104.2
Average time for running 100 times for random topologies (hrs)	0.003	0.013	0.03	0.06	0.17	0.41	1.06	1.74
Total time for running 100 times for random topologies (hrs)	2.49	11.4	26.0	49.3	140.4	336.5	892.1	1458.1
Total time for running 100 times for random topologies (days)	0.11	0.48	1.08	2.05	5.85	14.02	37.18	60.76

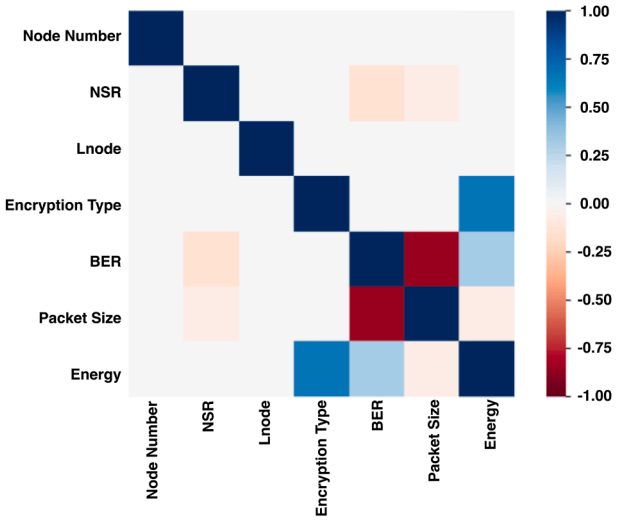


Figure 3. Correlations between variables

Fig. 3. Correlations between variables.

how these variables are distributed and how their values change according to each other. The subplots under the diagonal show the data distribution for each pair of variables. For example, from the Energy – NSR subplot (row 7, column 2), it can be seen how energy increases according to increasing NSR, except for a few outliers. Similarly, if other subplots are checked, it can be seen that Energy is mainly affected by NSR, Encryption Type, BER and Packet Size. The subplots over the diagonal show the distributions of the variables using kernel density estimation (KDE) [24]. KDE is one of the most popular methods for density estimation and it is calculated using the following function:

$$\widehat{p_n}(x) = \frac{1}{nh}\sum_{i=1}^{k} K\left(\frac{X_i}{h}\right) \tag{30}$$

where; *K(x)* is called the kernel function that is generally a symmetric function and *h* is the smoothing bandwidth to control the amount of smoothing. Fundamentally, KDE smooths out every point X_i into small density knobs and sums these knobs to find the final density estimate. In the subplots in Fig. 4, smooth densities for each variable pair are depicted. For example, Energy – Encryption Type subplot (row 4, column 7) in the Fig. shows the density of the energy values according to encryption types. The regression methods examined in this study are Linear Regression, Support Vector Regression, Gradient Boosting, k-Nearest Neighbors (kNN), Ridge Regression, Decision Tree, Random Forest, and XGBoost Regression. And the examined neural network methods are Artificial Neural Network (ANN) and Convolutional Neural Network (CNN). The reason behind selecting these algorithms is that we wanted to analyze both basic regression methods such as LR and KNN; ensembled methods such as Random Forest and Gradient Boosting; and neural network methods that are commonly used for regression.

Basic algorithms can be trained quite quickly, and they are efficient for making predictions in most of the cases while ensembled algorithms are more complex as they are built using several basic algorithms and they are very good at making predictions in most of the cases. Nevertheless, ensembled algorithms take longer times for training. Neural networks mimic the process of human brain and in general they can provide more accurate predictions than classical regression methods.

When building models, the data gathered via optimizations were used in two ways. First, the data was used as raw and then it was normalized using min-max scaler and used in the normalized form. Min-max scaler transforms features by scaling each feature to a given range. The formula of min max scaling is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{31}$$

Normalization is an important facility in machine learning, as it transforms raw data to avoid problems with datasets by generating new values and maintaining a general distribution in the data. Furthermore, it enhances the efficiency and veracity of machine learning models. Thus, we used normalized data to improve the performance of the models as it will be presented in the following parts of the report. We have used 80% of the data for training and 20% for testing.

To evaluate the performance of the models, we used R^2 score, mean absolute error, mean squared error, mean squared log error, root mean squared error, mean absolute percentage error, median absolute error, max error, and explained variance score metrics. The ML methods and the evaluation metrics that we have used in this study are described in the following part.

1) Proposed Machine Learning Methods

a Linear Regression

In statistics and mathematics, linear regression is utilized for defining the relationship between a dependent variable and one or more independent variables. If there is a single independent variable, the operation is named simple linear regression, and if there are multiple independent variables, the operation is called multiple linear regression [25]. Linear regression is the first type of regression method that was studied broadly by many scientists, and it is used in many practical applications.

b Support Vector Machine (SVM)

Support vector machines are supervised learning models used for data analysis for classification or regression tasks. SVM was developed and presented by Vapnik et al [26]. It tries to fit a line or hyperplane that divides classes of variables. Then it makes decisions about a new point according to the side of the hyperplane it resides. Its goal is to find a decision boundary at some distance from the hyperplane such that the data points nearby the hyperplane are inside that boundary.

c Gradient Boosting

In machine learning, boosting means combining some basic models into a single complex model. Generally, decision trees are selected as basic models in gradient boosting [27]. The term gradient is used because the algorithm utilizes gradient descent to minimize the loss and errors.

At every step, the algorithm finds the error between the new prediction and the pregiven target. Then it generates a simple model to map features to the residual. This residual is fed back to the current model as input and this way the algorithm moves the model closer to the correct prediction value.

d K-Nearest Neighbors (KNN)

KNN was first presented by Evelyn Fix and Joseph Hodges in 1951 in a project report [28]. The basic assumption is that similar objects or similar values exist in the vicinity.

It is easy to develop and comprehend, but it can become significantly slow as the size of the experimented data increases. KNN operates by calculating the gaps between a prediction and the test data, then chooses k samples (neighbors) which are closest to the prediction, and lastly it takes the average of the selected samples as the new prediction.

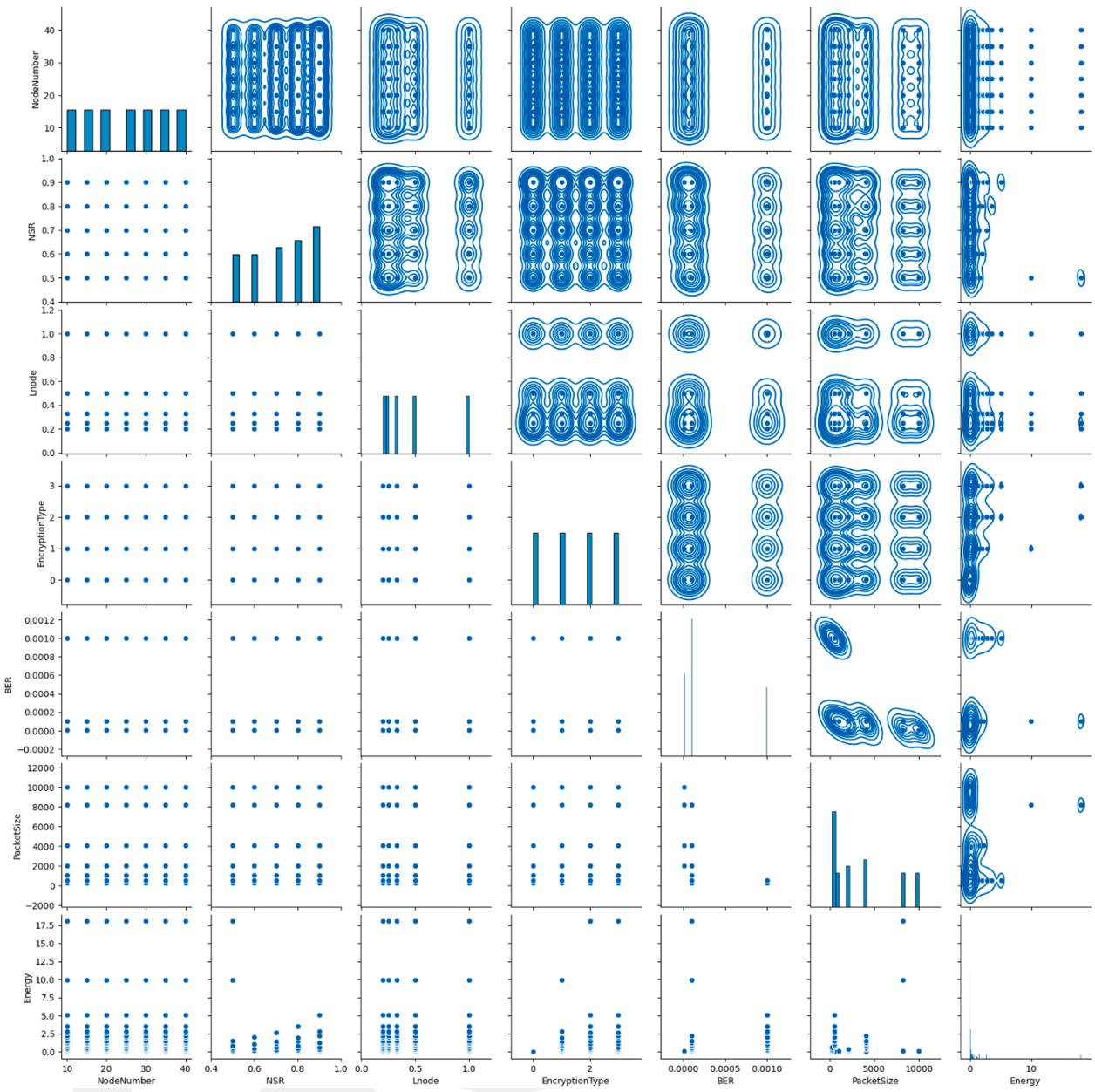


Fig. 4. Pair-plot of the variables.

e Ridge Regression

Ridge regression is generally used when independent variables are vastly correlated [29]. The method was proposed in 1970 by Hoerl et al. [30].

If there is multicollinearity, least squares are unbiased, but their variances are high so the model may lead to predictions distant from the correct value. The method adds a level of bias to the regression to decrease the standard errors.

f Decision Tree

Decision trees can be built from inspections about an attribute given at the branches of the tree and ending at the predictions about the target value given at the leaves [31].

g Random Forest

Random Forest is a combined learning method applied by building multiple decision trees at training. When evaluating a regression model, the average, or the mean of the predictions from individual decision trees is calculated [32]. The advantage of Random Forests is their resistance to overfitting where decision trees may face.

h XGBoost Regression

XGBoost is an abbreviation for Extreme Gradient Boosting, and it is developed as an efficient version of the gradient boosting algorithm [33]. With gradient boosting, the basic models are decision trees, where every tree allocates an input to a leaf that has a continuous value. As in gradient boosting, the training runs iteratively, adding new trees at each step to make new predictions according to the prediction errors of

preceding trees. Then the new trees are merged with prior trees to complete the final model.

i Artificial Neural Networks (ANN)

An artificial neural network is a computational model that simulates the nerve cells in human brain [34]. Each neuron in the layers holds a weight and at each iteration these weights are adjusted according to the error calculated by comparing actual value and the predicted value. The calculated errors are fed back to refine the weights and this process is called backpropagation. Thus, the calculated errors are used to adjust the weight of the neurons.

j Convolutional Neural Networks (CNN)

Convolutional neural networks are a customized type of ANNs, which use a mathematical operation called convolution for general matrix multiplication in at least one of its layers [35]. In a CNN, there are hidden layers which are responsible for convolutions. These hidden layers include one that computes the dot product of the convolution kernel with the layer's input matrix. As the kernel moves over the input matrix, the convolution process produces a feature map, which advances as the input of the next layer.

2) Evaluation Metrics

a R^2 Score

R^2 score is a statistical metric that illustrates the fraction of the variance for a dependent variable that is defined by independent variables in a regression function. R^2 depicts how variance of a variable is defined by variance of another variable. R^2 is formulated as:

$$R^2 = 1 - SS_{res}/SS_{total} \tag{32}$$

where SS_{res} is the sum of squares of the residual errors and SS_{total} is the sum of squares of all errors.

b Mean Absolute Error (MAE)

The mean absolute error of a regression model is the mean of the absolute error values encountered at separate predictions about all data values in the test set. A prediction error is the disparity between the correct value and the predicted value for a data point.

c Mean Squared Error (MSE)

Mean squared error indicates how distant a set of predicted values is from a regression line. It takes squares of the errors to avoid negative signs and improve their weights.

d Mean Squared Log Error (MSLE)

It is a variation of MSE, which considers the proportional difference between the actual and predicted values which are log-transformed.

e Root Mean Squared Error (RMSE)

It is the standard deviation of the errors of predictions generated by a regression model. It tries to measure how these errors are scattered around the best fit line.

f Mean Absolute Percentage Error (MAPE)

It considers accuracy as a percentage, and it is computed as the average absolute percent error minus actual values divided by actual values. It gives better interpretations if there are no outliers in the data.

g Median Absolute Error (MAE)

The median absolute error is a measure that is robust to outliers. The error is computed by finding the median of all absolute errors between the target and the prediction values.

h Max Error (ME)

As its name implies, it is simply the largest residual error among the actual and predicted values. It can be useful to understand the accuracy of the regression model when combined with other metrics.

i Explained Variance Score (EVS)

Explained variance is used to measure the proportion of the variability of the predictions of a regression model. The formula is 1 – variance (y – y') / variance (y), where y is the real data and y' is the predicted data.

5. Performance results

Before discussing the evaluation scores, training and testing time of the algorithms are presented in Table 5 for models generated with raw data and normalized data with %80 of the dataset for training and 20% of the dataset for testing. As it can be seen from the table, training times are quite short, sometimes even negligible when compared to optimization runtimes given in Table 1. This supports the proposal of using machine learning methods for network parameter prediction instead of running heavy optimizations.

In the convention, scores and error metrics are analyzed together to decide the success of a model and most acclaimed metrics are R2 score, mean absolute error and mean squared error. In most of the cases, if errors are low then scores become high or vice versa. Nevertheless, instead of analyzing metrics for each model separately, comparing the errors and scores of models helps better interpret the success of the models.

The first method implemented was Linear Regression (LR). When the performance of this method is evaluated with different performance metrics, it is shown that the predictions of LR are not successful. Both executions with raw and normalized data return similar results. R^2 score generated by LR is 0.076 both with row and normalized data, which means the generated model cannot define the function well. Oppositely, the error metrics are high indicating that the predictions of the method are erroneous.

The next examined method was Support Vector Regression. If the results of this method are examined, the effect of normalizing the data can be understood better. The algorithm gives inadequate results when run with raw data, but it is more successful when run with normalized data. Nevertheless, the scores and errors of the method show that it is still not satisfactory to be used with the dataset in hand.

Another implemented method was Gradient Boosting. It is a very efficient method, and it is widely used among data scientists. The results generated by this algorithm prove that it is efficient with both raw data and normalized data. It generates a good model with quite good scores for evaluation metrics. R^2 score is 0.987 for both raw and normalized data, meaning that the method defines the function well. Still, error

values calculated for normalized data are lower than the error values of raw data.

The next investigated method was KNN regression. The scores and errors generated by the model show that it can make good predictions. Though, it is not the best method to use for the generated dataset. The calculated evaluation metrics depict that normalizing the data before training generates a better model. KNN produces a moderate fit of the predictions against the actual values.

Another method was Ridge Regression. The scores and errors generated by the model indicate that it cannot make very good predictions. However, the method again makes better predictions with normalized data. Yet, Ridge regression is not suitable for the current dataset.

The next method implemented was Decision Tree. The performance metrics calculated according to the predictions of the model presents that it produces quite high scores against small errors. From the results, it is obvious that the method also works well without normalizing the data. Decision Tree seems to be a candidate method for the current dataset.

The next method was Random Forest. The calculated evaluation metrics for the model depict that it produces quite good scores with very minor errors. The metrics also show that the method still works well without normalizing the data, although running with normalized data improves the model. Random Forest algorithm is suitable to be used with the dataset in hand.

Table 3 demonstrates a comparison of scores and errors for all the applied methods using raw data. Regarding R^2 scores, the highest values are provided by Decision Tree and XGBoost. Moreover, Gradient Boosting and Random Forest methods provide quite good scores even with raw data. A negative R^2 score means that the chosen model does not follow the trend of the data and scores close to 0 indicate that the model cannot interpret the data well. Mean Absolute Error is the mean of sum of all errors of the predictions against real values. A MAE close to 0 means that the predictions of the model are less erroneous. When we check MAE values, again Decision Tree and XGBoost provide very small numbers. Also, Gradient Boosting and Random Forest yield acceptably small errors which are adequate.

Table 4 presents a comparison of scores and errors for all the applied methods using normalized data. When compared to the results generated using raw data, the metrics show that all the methods perform better with normalized data. The unsuccessful algorithms for the current dataset are Linear Regression and Ridge Regression. The highest scores and lowest errors are still provided by Decision tree and XGBoost methods, and other successful algorithms are again Gradient Boosting and Random Forest while SVM and KNN are moderate. When the metrics of neural network models are observed, CNN generates very high scores and very low errors as Decision Tree and XGBoost. On the other hand, ANN generates a successful model that can be used with the dataset in hand.

6. Conclusion

In this paper, machine learning (ML) models have been proposed to predict network parameters and energy consumption of underwater nodes as supplementary methods to optimization models. To this end, we have implemented different regression models and neural network-based models using Scikit-learn and Keras tools and analyzed the performance of these models using score and error metrics. To collect the data to be used by the models, we have run our optimization model with different combinations of our parameters.

The reason that we suggest using machine learning models is that once we build the model, it can be run very fast on any computer. We can change the parameters to calculate new predictions using our models without running computationally intensive optimizations. Besides requiring heavy calculations, optimizations sometimes cannot produce feasible solutions or finish in a reasonable time. By using machine learning models, we can avoid the high computational burden of optimizations. Another point to express is that we can always make a prediction with different parameters even if the optimization returns unfeasible solutions with given parameters. Performance results of the examined algorithms show that:

- Linear Regression, KNN, and Ridge Regression are not successful with the given dataset as they produce R^2 scores around 0.075 showing that the model cannot define the relations between the variables well.
- Gradient Boosting, XGBoost, Decision Tree and Random Forest are the best regression methods in terms of different performance metrics. They are quite successful both with raw and normalized data, producing R^2 scores close to 1.
- ANN and CNN cannot define the model if they are run with raw data. On the other hand, they produce successful models if they are run with normalized data. With normalized data, R^2 scores for ANN and CNN are 0.96 and 0.99 respectively.
- Normalization improves the prediction performance for all the examined algorithms and reduces errors.

An important point to highlight is that the success of the neural networks increases if the data is normalized before building the model. Normalization is important for our case since our data has a wide range of values and mapping these values to a range helps the algorithms process the data better. Moreover, even if optimizations cannot come up with feasible results or cannot finish in an acceptable time frame, once a regression model is built, predictions can be still made using the previously generated data which proves that machine learning is an ancillary technique for optimizations.

As a future work, we are planning to design a Reinforcement Learning (RL) based method for predicting design parameters for underwater sensor networks. In RL, positive behaviors are rewarded, and negative behaviors are punished meaning that the model is forced to search for a long-term optimum reward to find the best solution. One of the areas that RL is applied is resource management. RL can allocate limited resources for different jobs to find optimum solutions, which is suitable for UASNs that have limited batteries. In our scenario, for example, RL can be used to predict the optimal encryption method or optimal packet size for a node. Nevertheless, applying RL depends on

Table 3
Scores and errors of analyzed methods using raw data.

Method/Metrics	Lin. Reg.	SVM	Grad. Boost.	KNN	Ridge Reg.	Dec. Tree	Rand. Forest	XGBoost	ANN	CNN
R^2 score	0.076	-0.054	0.987	0.332	0.0237	0.999	0.981	0.999	-0.004	-0.660
Mean Abs. Err.	0.815	0.575	0.111	0.495	0.870	5.434e-06	0.118	0.001	0.906	1.630
Mean Sq. Err.	3.189	3.639	0.0450	2.304	3.369	3.081e-10	0.0672	2.049e-06	3.468	5.730
Mean Sq. Log Err.	0.275	0.241	0.0114	0.166	0.314	2.661e-10	0.009	1.386e-06	0.305	0.766
Root Mean Sq. Err.	1.786	1.908	0.212	1.518	1.836	1.755e-05	0.259	0.002	1.862	2.393
Mean Abs. Perc. Err.	264.84	150.53	52.49	2.219	157.58	0.1360	41.822	6.394	321.437	166.301
Median Abs. Error	0.493	0.101	0.052	0.0165	0.564	5.551e-17	0.031	0.0003	0.688	1.072
Max Error	16.335	17.959	1.273	17.995	16.379	0.0002	1.097	0.006	17.327	14.078
Expl. Var. Score	0.081	0.0015	0.987	0.333	0.027	0.999	0.981	0.999	0.0	-0.309

Table 4
Scores and errors of analyzed methods using normalized data.

Method/Metrics	Lin. Reg.	SVM	Grad. Boost.	KNN	Ridge Reg.	Dec. Tree	Rand. Forest	XGBoost	ANN	CNN
R^2 score	0.076	0.752	0.987	0.729	0.076	0.999	0.981	0.999	0.966	0.997
Mean Abs. Err.	0.045	0.028	0.006	0.020	0.0451	3.1e-07	0.007	0.0001	0.009	0.003
Mean Sq. Err.	0.009	0.003	0.0001	0.003	0.009	9.6e-13	0.0002	7.9e-08	0.0004	3.1e-05
Mean Sq. Log Err.	0.006	0.002	0.0001	0.002	0.006	9.5e-13	0.0002	7.5e-08	0.0003	2.3e-05
Root Mean Sq. Err.	0.0989	0.051	0.0118	0.054	0.0989	9.8e-07	0.0143	0.0003	0.0190	0.006
Mean Abs. Perc. Err.	445.23	229.79	86.25	1.764	445.12	0.220	47.532	16.580	61.873	9.557
Median Abs. Error	0.027	0.009	0.003	0.001	0.0273	7.8e-18	0.0017	9.9e-05	0.003	0.003
Max Error	0.904	0.442	0.0705	0.665	0.905	1.1e-05	0.061	0.001	0.095	0.039
Expl. Var. Score	0.081	0.773	0.987	0.729	0.0812	0.999	0.981	0.999	0.966	0.997

Table 5
Runtimes for ML algorithms.

Runtimes (secs)/Algorithms	Raw Data		Normalized Data	
	Train	Test	Train	Test
Lin. Reg.	0.0421	0.0005	0.0011	0.0003
SVM	0.3438	0.0869	0.0853	0.0132
Grad. Boost.	0.1572	0.0028	0.1398	0.0023
KNN	0.0117	0.0059	0.0013	0.0041
Ridge Reg.	0.0059	0.0007	0.0007	0.0003
Dec. Tree	0.0103	0.0007	0.0042	0.0004
Rand. Forest	0.1726	0.0059	0.1704	0.0057
XGBoost	0.1380	0.0021	0.1016	0.0017
ANN	6.1751	0.0344	6.3844	0.0372
CNN	8.0554	0.0389	8.2937	0.0378

exploring the parameters and resources in a time period and requires a different dataset that contains information about the state of the network parameters at each round.

Declaration of Competing Interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Data availability

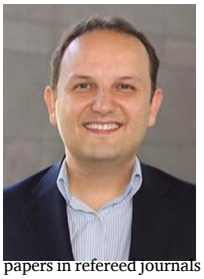
Data will be made available on request.

References

[1] H. Zlatokrilov, H. Levy, Session privacy enhancement by traffic dispersion, in: Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), 2006, pp. 1–12.
[2] D. Incebacak, K. Bicakci, B. Tavli, Evaluating energy cost of route diversity for security in wireless sensor networks, Comput. Standarts Interfaces (39) (2015) 44–57.
[3] O.G. Uyan, A. Akbas, V.C. Gungor, A reliable and secure multi-path routing strategy for underwater acoustic sensor networks, Comput. Networks 212 (2022).
[4] A. Akbas, H.U. Yildiz, A.M. Ozbayoglu, B. Tavli, Neural network based instant parameter prediction for wireless sensor network optimization models, Wireless Networks 25 (6) (2019) 3405–3418.
[5] M. Yilmaz, A.M. Ozbayoglu, B. Tavli, Efficient computation of wireless sensor network lifetime through deep neural networks, Wireless Networks 27 (3) (2021) 2055–2065.
[6] A. Akbas, S. Buyrukoglu, Stacking ensemble learning-based wireless sensor network deployment parameter estimation, Arabian J. Sci. Eng. (2022) 1–10.
[7] Y. Chen, W. Yu, X. Sun, L. Wan, Y. Tao, X. Xu, Environment-aware communication channel quality prediction for underwater acoustic transmissions: a machine learning method, Appl. Acoustics 181 (2021).
[8] V. Kalaiarasu, H. Vishnu, A. Mahmood, M. Chitre, Predicting underwater acoustic network variability using machine learning techniques, OCEANS 2017-Anchorage (2017) 1–7.
[9] M. Alamgir, M. Sultana, K. Chang, Link adaptation on an underwater communications network using machine learning algorithms: boosted regression tree approach, IEEE Access 8 (2020) 73957–73971.
[10] E. Eldesouky, M. Bekhit, A. Fathalla, A. Salah, A. Ali, A robust UWSN handover prediction system using ensemble learning, Sensors 21 (5777) (2021).
[11] L. Liu, L. Cai, L. Ma, G. Qiao, Channel state information prediction for adaptive underwater acoustic downlink OFDMA system: deep neural networks based approach, IEEE Trans. Veh. Technol. 70 (9) (2021) 9063–9076.
[12] M. Felemban, E. Felemban, Energy-delay tradeoffs for underwateracoustic sensor networks, in: 2013 First International Black Sea Conference on Communications and Networking (BlackSeaCom), 2013, pp. 45–49.
[13] MATLAB, 9.12.0.1884302 (R2022a), The MathWorks Inc., Massachusetts, 2022.
[14] IBM Cplex Optimizer, IBM, 2022.
[15] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.
[16] F. Pedregosa, Scikit-learn: machine learning in python, JMLR 12 (2011) 2825–2830.
[17] F. Chollet, "Keras: deep learning for humans.," 2022. [Online]. Available: https://github.com/keras-team/keras. [Accessed 2022].
[18] M. Abadi, TensorFlow: a system for large-scale machine learning, in: Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation, 2016, pp. 265–283.
[19] W. McKinney, Data structures for statistical computing in python, in: Proc. of the 9th Python in Science Conf, 2010, pp. 56–61.
[20] S. Brugman, "Pandas-profiling.," 2022. [Online]. Available: https://github.com/ydataai/pandas-profiling. [Accessed 2022].
[21] T. Kluyver, Jupyter notebooks – a publishing format for reproducible computational workflows. Positioning and Power in Academic Publishing: Players, Agents and Agendas, 2016, pp. 87–90.
[22] M.L. Waskom, Seaborn: statistical data visualization, J. Open Source Software (6) (2016) 60.
[23] J. Zar, Spearman rank correlation, Encycloped. Biostatistics 7 (2005).
[24] S. Węglarczyk, Kernel density estimation and its application, ITM Web Conf. 23 (2018) 37.
[25] D.A. Freedman, Statistical Models: Theory and Practice, Cambridge University Press., 2009.
[26] C. Corinna, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[27] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Statistics 29 (5) (2001) 1189–1232.
[28] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Statis. 46 (3) (1992) 175–185.
[29] D.E. Hilt, D.W. Seegrist, Ridge, a computer program for calculating ridge regression estimates, Depart. Agricult., Forest Serv., Northeastern Forest Experiment Station (1977).
[30] A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, Technometrics 12 (1) (1970) 55–67.
[31] X. Wu, V. Kumar, J.R. Quinlan, Top 10 algorithms in data mining, Knowl. Inf. Syst. 57 (3) (2008) 238–247.
[32] T.K. Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition 1, 1995, pp. 278–282.
[33] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[34] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[35] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.0845,* 2015.

Osman Gokhan Uyan received his B.S degree in Computer Science from Bilkent University, Ankara, Turkey, in 2005. He received his M.Sc. and Ph.D. degrees in Electrical and Computer Engineering at Abdullah Gul University. His main research interests are in wireless and mobile communications, wireless and ad-hoc sensor networks, fuzzy logic and intelligent optimization algorithms, image processing.

Asst. Prof. Dr. Ayhan Akbas has received B.Sc. and M.Sc degrees from Middle East Technical University, Electrical and Electronics Department in 1991 and 1995 respectively. He earned his Ph.D. in Computer Engineering from TOBB University of Economics and Technology. He worked for multinational companies as SIEMENS, Sun Microsystems, and NEC in technical and managerial positions for over 20 years in the IT business. Currently, he is continuing his career as an academician in Computer Engineering Department at Abdullah Gul University. His areas of interest are Wireless Sensor Networks, Wireless Communication, Computer Networks, IoT, RFID, Mathematical Modelling and Optimizations.



Prof.Dr. V. Cagri Gungor received his B.S. and M.S. degrees in Electrical and Electronics Engineering from METU, Ankara, Turkey, in 2001 and 2003, respectively. He received his Ph.D. degree in electrical and computer engineering from the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA, USA, in 2007. Currently, he is an Associate Professor and Chair of Computer Engineering Department, Abdullah Gul University (AGU), Kayseri, Turkey. His current research interests are in smart grid communications, machine-tomachine communications, next-generation wireless networks, wireless ad hoc and sensor networks, cognitive radio networks. Dr. Gungor has authored several papers in refereed journals and international conference proceedings, and has been serving as an editor, reviewer and program committee member to numerous journals and conferences in these areas. He is also the recipient of TUBITAK Young Scientist Award in 2017, Science Academy Young Scientist Award (BAGEP) in 2017, Turkish Academy of Sciences Distinguished Young Scientist Award (TUBA-GEBIP) in 2014, the IEEE Trans. on Industrial Informatics Best Paper Award in 2012, the European Union FP7 Marie Curie IRG Award in 2009, Turk Telekom Research Grant Awards in 2010 and 2012, and the San-Tez Project Awards supported by Alcatel-Lucent, and the Turkish Ministry of Science, Industry and Technology in 2010.