Computer Standards & Interfaces 89 (2024) 103808

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm

Burak Kolukisa [a,*], Bilge Kagan Dedeturk [b], Hilal Hacilar [a], Vehbi Cagri Gungor [a]

[a] Department of Computer Engineering, Abdullah Gül University, Kayseri, Turkey
[b] Department of Software Engineering, Erciyes University, Kayseri, Turkey

ARTICLE INFO

Keywords:
Network intrusion detection system
Anomaly detection
Machine learning
Artificial bee colony
Logistic regression
UNSW-NB15
NSL-KDD

ABSTRACT

In recent years, the widespread use of the Internet has created many issues, especially in the area of cybersecurity. It is critical to detect intrusions in network traffic, and researchers have developed network intrusion and anomaly detection systems to cope with high numbers of attacks and attack variations. In particular, machine learning and meta-heuristic methods have been widely used for network intrusion detection systems (NIDS). However, existing studies on these systems usually suffer from low performance results such as accuracy, F1-measure, false positive rate, and false negative rate, and generally do not use automatic parameter tuning techniques. To address these challenges, this study proposes a novel approach based on a logistic regression model trained using a parallel artificial bee colony (LR-ABC) algorithm with a hyper-parameter optimization technique. The performance of the proposed model is evaluated against state-of-the-art machine learning and deep learning models on two publicly available NIDS datasets. Comparative performance evaluations show that the proposed method achieved satisfactory results with accuracy of 88.25% on the UNSW-NB15 dataset and 90.11% on the NSL-KDD dataset, and F1-measures of 88.26% and 90.15%, respectively. These findings demonstrate the efficacy of the proposed LR-ABC model in enhancing the accuracy and reliability, while providing a scalable solution to adapt to the dynamic and evolving landscape of cybersecurity threats.

1. Introduction

In recent years, the number of people and applications utilizing the internet has expanded dramatically, owing largely to the development of smart technology. According to the Data Reportal, which collects data about internet usage worldwide, approximately one million new internet users have been added each day, and the total number of internet users increased by 13% in 2020 compared to previous years [1]. The increased usage of the internet has also brought several security problems. Cybercrime and threat activities have become a critical part of our daily lives, and the necessity of cybersecurity has emerged as a major concern. SonicWall reported that approximately 4.8 trillion intrusion attempts occurred in 2020, representing a 20% increase over the previous year [2]. These infiltration attempts are aimed at entering information systems and stealing or obscuring confidential-sensitive data. As a result of attacks and intrusions, credit card passwords are stolen, server systems are infiltrated, and information services are rendered useless. Numerous technologies are employed to close these security gaps, such as firewalls, data encryption, and user authentication. These security mechanisms protect against a wide variety of attacks. However, they are incapable of performing in-depth packet analysis. As a result, they are not sufficiently able to detect enough attacks or attack types. Therefore, as a result of these concerns, network intrusion detection systems (NIDS) have been created to compensate for the deficiencies of the security methods, which monitor the network continually for malicious attacks and warning users when intrusions or attacks occur. These systems are commonly used to guard against network attacks and warn users of abnormal conditions.

In general, NIDS are classified into two categories: signature and anomaly-based. Signature-based solutions rely on a database of malware signatures, but they are not preferable owing to the expanding diversity of attacks; an attacker may easily bypass the system with minimal alterations to the attack. Additionally, the signature-based system is slowing down as it continues to learn new signatures and restrictions. On the other hand, anomaly-based detection, in contrast, establishes a pattern without relying on signatures and identifies malware using the learned model. These models are capable of accomplishing more in-depth data analysis thanks to their algorithms based on artificial intelligence and machine learning (ML) approaches.

* Corresponding author.
E-mail addresses: burak.kolukisa@agu.edu.tr (B. Kolukisa), kagandedeturk@gmail.com (B.K. Dedeturk), hilal.hacilar@agu.edu.tr (H. Hacilar), cagri.gungor@agu.edu.tr (V.C. Gungor).

https://doi.org/10.1016/j.csi.2023.103808
Received 6 May 2022; Received in revised form 11 November 2023; Accepted 18 November 2023
Available online 21 November 2023

Specifically, NIDS uses a variety of ML approaches, including rule mining, classification, clustering, and deep learning based algorithms. These strategies are implemented to safeguard the network's security by detecting intrusions and attacks with a high rate of accuracy and F1-measure. While ML approaches offer benefits, they alone are insufficient and present difficulties; they require additional data preprocessing steps guided by human expertise to address issues, such as detecting outliers, a high cost of errors, a semantic gap between the results and their interpretation, diversity in the generated network traffic, a variety of attack types, and difficulties with data evaluation [3]. Recently, it was shown that a variety of critical issues can be handled, such as massive network traffic, diverse data distribution, and continuously changing environmental circumstances, by integrating the artificial bee colony (ABC) approach with ML methods [4]. To this end, the ABC algorithms provide the following advantages: (i) they need less previous knowledge about the data and human skills, which allows the classification process without using particular data preprocessing techniques [5]. (ii) The hybridization of the ABC approach with ML techniques is improving the model results [3]. (iii) Unlike many ML approaches, the ABC method is far less reliant on known labels within the dataset [6]. (iv) It is distributed by design and performs well in parallel and distributed computing environments [7].

This study proposes a novel anomaly-based NIDS approach using logistic regression (LR), known for its simplicity, fast classification in real-time applications [8], and efficiency. This approach avoids LR's tendency to convergence to poor local minima by training it using the ABC [9] algorithm, which is a nature-inspired swarm intelligence algorithm, that simulates the foraging behavior of honey bees.

To the best of our knowledge, this study builds the first anomaly-based NIDS approach that utilizes the parallel ABC as an LR learning algorithm. The ABC algorithm works well on multimodal and high-dimensional issues [10,11] and is capable of balanced exploration and exploitation ability, which makes it an attractive candidate for anomaly-based NIDS. Additionally, the computational time of the suggested solution is decreased via the use of parallel computing techniques, and the Sequential Model-Based Optimization Configuration (SMAC) technique is used to optimize ML algorithms' hyper-parameters. The proposed approach is evaluated using two publicly available network datasets: UNSW-NB15 and NSL-KDD. The performance of the proposed model is evaluated with state-of-the-art ML and deep learning (DL) models, such as decision tree (DT), linear discriminant analysis (LDA), logistic regression (LR), multi-layer perceptrons (MLP), random forest (RF), support vector machines (SVM), extreme gradient boosting (XGBoost), deep neural network (DNN), long short-term memory (LSTM), and gated reccurent unit (GRU). Comparative experiments on the UNSW-NB15 and NSL-KDD datasets show that the proposed model outperforms other methods in accuracy, false positive rate (FPR), and F1-measure for UNSW-NB15, as well as in accuracy, false negative rate (FNR), and F1-measure for NSL-KDD, while reducing the training time. The proposed model achieves an accuracy of 88.25% on the UNSW-NB15 dataset and 90.11% on the NSL-KDD dataset, and F1-measures of 88.26% and 90.15%, respectively. Additionally, thanks to GPU parallelization, the proposed model's training time was approximately 4.45 times faster than the CPU version of the LR-ABC approach, indicating a significant improvement in execution speed. Overall, the major contributions of this study can be summarized as follows:

- This study proposes an efficient approach based on a logistic regression model trained by a parallel artificial bee colony algorithm for network intrusion detection systems.
- To overcome the high computational time of the standard logistic regression (LR) - artificial bee colony (ABC) models, an efficient LR-ABC model has been developed based on CPU and GPU parallelization techniques to significantly reduce training time.
- The performance of the proposed approach outperforms the state-of-the-art machine learning and deep learning models in terms of accuracy, false positive rate, false negative rate, and F1-measure.
- Comparative performance evaluations are based on the publicly available UNSW-NB15 and NSL-KDD datasets, which are among the most comprehensive available datasets. The high performance of the proposed approach shows that the proposed model is reliable and robust to detect various attack types, and it provides a scalable solution for adapting to the dynamic and evolving landscape of cybersecurity threats.
- The SMAC parameter optimization method has been utilized to automatically optimize the hyper-parameters of the proposed logistic regression-artificial bee colony approach, the state-of-the-art machine learning and deep learning methods.

This study is organized as follows: Section 2 provides an overview of the current ML-based intrusion detection systems. Section 3 describes evaluation metrics, available datasets, preprocessing steps and the hyper-parameter optimization method. In Section 4, the proposed LR-ABC approach and the parallel computing method are explained. Section 5 outlines the experimental steps. Section 6 presents the performance results of the proposed LR-ABC and other classification methods. The last section concludes the paper and discusses future research directions.

2. Related work

In recent years, attackers have been upgrading themselves and the software that they use and inventing new malicious activities. Until now, different ML-based NIDS have been developed. Anomaly-based NIDS are favored for their ability to identify novel attacks types, unlike signature-based systems. Due to the automated nature of ML techniques, they are able to develop a variety of models without the strong involvement of human skills [12], which is sometimes a constraint and costly. For this purpose, many studies are aim to increase the performance of anomaly-based NIDS for different types of cybersecurity attacks.

Hajisalem et al. [13] suggest a hybrid method for anomaly-based NIDS that combines the ABC and Artificial Fish Swarm (AFS) algorithms. This hybrid method generates rules through the use of fuzzy C-means clustering (FCM) and correlation-based feature selection (CFS) techniques. They generate if-then rules using the CART technique to distinguish normal and anomalous records. Qureshi et al. [14] suggest a NIDS that utilizes a random neural network (RNN) trained with the ABC algorithm to discover the ideal weights for the neurons, followed by a comparison to the classic gradient descent-based RNN model. Mazini et al. [15] suggest a hybrid method that combines an ABC algorithm for feature selection to select the best subset of related features and an AdaBoost meta-algorithm for classification. Gu et al. [16] create a NIDS that uses SVM with the tabu-ABC for feature selection and parameter optimization at the same time. They adopted the tabu search algorithm to enhance the neighborhood search of ABC (TABC). TABC-SVM is utilized for reducing the feature size dimensions, and meanwhile, SVM parameters are optimized. Finally, the dataset is utilized to train the SVM classifier model using the appropriate feature subset and hyper-parameters. Rani et al. [17] use the ABC algorithm for the feature selection process and a random forest classifier for classification tasks. Additionally, they demonstrate in other research why feature selection procedures result in overfitting and are unable to improve classification accuracy on NIDS [18]. In our previous studies [19], the ABC algorithm was applied to LR on e-mail spam filtering tasks and then evaluated on three public datasets. The proposed approach is compared with other ML algorithms. The suggested methodology outperforms other approaches in terms of classification accuracy and false positive rate. The proposed model demonstrates a high degree of effectiveness on unbalanced and nonlinear spam datasets. However, the suggested method

has a shortcoming as it requires more training time compared to the other methods.

On the other hand, several studies on NIDS have focused on different preprocessing steps and used individual classifiers such as DT [20], LDA [21], LR [22], MLP [23], and SVM [24] on the UNSW-NB15 dataset. While these studies provide valuable insights into NIDS, no have explored LR-ABC classification for anomaly-detection on NIDS. In this study, the proposed LR-ABC approach is compared to the state-of-the-art ML and DL algorithms, which include DT, LDA, LR, MLP, RF, SVM, XGBoost, DNN, LSTM, and GRU classifiers, using the UNSW-NB15 and NSL-KDD datasets. Additionally, this study emphasizes that no cleaning process was applied to the datasets, and feature selection methods were not used. While some preprocessing methods could potentially increase accuracy, such enhancements are outside the scope of this study.

Overall, ML and metaheuristic methods have been widely used for NIDS. However, existing studies on NIDS usually suffer from low performance results such as accuracy, F1-measure, false positive rate, and false negative rate. Moreover, current studies generally do not use automatic parameter tuning techniques. To address these challenges, this paper proposes a novel approach based on a logistic regression model trained using a parallel artificial bee colony algorithm with a hyper-parameter optimization technique. To overcome the high computational time of the LR-ABC models, an efficient LR-ABC model has been developed based on CPU and GPU parallelized technique to significantly reduce training time. To the best of our knowledge, this study proposes the first anomaly-based NIDS approach that employing the parallel ABC as an LR learning algorithm.

3. Materials & methods

3.1. Evaluation metrics

Accuracy is a essential criterion for evaluating a model's overall performance. The major goal of the current research is to increase the accuracy of NIDS, but the accuracy criterion may not be adequate in NIDS, especially in anomaly detection. Therefore, in addition to the accuracy metric, F1-measure, FPR and FNR metrics, training time are also used to evaluate the classification performance. The FPR measures the rate of the normal traffic falsely detected as anomalies, while the FNR indicates the rate of actual anomalies mistakenly classified as normal. The F1-measure, the harmonic mean of recall and precision, reflects the model's sensitivity and robustness. These are important details to be examined in the NIDS. These performance metrics are given in Eqs. (1), (2), (3), and (4), respectively. These metrics help to assess the performance of the model in several aspects. The traditional confusion matrix is shown in Table 1.

$$Accuracy\,(ACC) = \frac{TP + TN}{TP + FN + FP + FN} \tag{1}$$

$$FPR = \frac{FP}{TN + FP} \tag{2}$$

$$FNR = \frac{FN}{TP + FN} \tag{3}$$

$$F1 - measure\,(F1) = \frac{2\,TP}{2\,TP + FP + FN} \tag{4}$$

3.2. Dataset

The UNSW-NB15 [25] and NSL-KDD [26] are benchmark network traffic datasets well-known in NIDS research. The goal of the datasets generation is to provide a robust and realistic dataset. The UNSW-NB15 dataset provides training and testing datasets separately. This dataset provides both multiclass and binary class labels. The training set contains a total of 175,341 samples, of which 56,000 are labeled "normal" and 119,341 are labeled "abnormal". Similarly, the testing set consists of 82,332 samples, of which 37,000 are labeled "normal", and 45,332 are labeled "abnormal" traffic samples. The dataset contains 45 features and abnormal classes containing nine attack types, including backdoors, analysis, DoS, exploits, fuzzers, generic, reconnaissance, shell code, and worms. The NSL-KDD is divided into two parts: KDDTrain+ and KDDTest+. The training set has 125,973 samples, with 67,343 labeled "normal" and 58,630 labeled "abnormal", including 22 attack types, which is categorized into four attack classes. The test set has 22,544 samples of which 9711 are labeled "normal", and 12,833 are labeled "abnormal" and including 37 attack types, also grouped into four attack classes. The distribution of four classes is: denial of service (DoS) attacks, root-to-local attacks (R2L), user-to-root attacks (U2R), and probing attacks (Probe). The NSL-KDD dataset contains 41 features. Table 2 shows the class distribution of the UNSW-NB15 and NSL-KDD datasets.

Table 1
Traditional confusion matrix.

	Predicted Abnormal	Predicted Normal
Actual Abnormal	TP	FN
Actual Normal	FP	TN

Table 2
Class distribution of UNSW-NB15 and NSL-KDD datasets.

Dataset	Class	Training Set	Test Set
UNSW-NB15	Normal	56.000	37.000
	Fuzzers	18.184	6.062
	Analysis	2.000	677
	Backdoors	1.746	583
	Dos	12.264	4.089
	Exploits	33.393	11.132
	Generic	40.000	18.871
	Reconnaissance	10.491	3.496
	Shellcode	1.131	378
	Worms	130	44
	Total	**175.341**	**82.332**
NSL-KDD	Normal	67.343	9.711
	Dos	45.927	7.458
	Probe	11.656	2.421
	U2R	52	200
	R2L	995	2.754
	Total	**125.973**	**22.544**

3.3. One hot encoding

Machine learning algorithms consider the magnitude of numerical values as the importance or significance of features. In other words, on the categorical values, it will consider the higher number more important or superior to a lower number. Therefore, on the UNSW-NB15 dataset, which has categorical features, one hot encoding technique is applied to transform categorical features to numeric values. For instance, the 'state' feature has nine categorical values: 'FIN', 'INT', 'CON', 'ECO', 'REQ', 'RST', 'PAR', 'URN', and 'no'. These were turned into binary vectors using the one-hot encoding method as follows: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], ..., [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]. The 'service', 'state', and 'proto' are the three categorical features in the UNSW-NB15 dataset and after encoding, the total number of features increases to 199. Similarly, the NSL-KDD dataset contains categorical features such as 'protocol type', 'service', and 'flag', which after binary encoding, expand the total feature count to 122.

3.4. Data normalization

Normalization eliminates the influence of various scales across features, thereby reducing the time required to train the model. There are several normalization approaches. To choose the most suitable one, the dataset is analyzed for sparsity, a measure indicating how prevalent

Table 3
Hyperparameter ranges for classification methods for UNSW-NB15 and NSL-KDD datasets.

Classifier	Parameter	Lowest	Highest
DT	Min Samples Split	2	100
	Min Samples Leaf	1	100
DL	Batch Size	8	64
	Learning Rate	10^{-6}	10^{-2}
	Neuron1	1	128
	Neuron2	1	128
	Neuron3	1	128
LDA	Shrinkage	0	1
LR	C	10^{-4}	10^{4}
LR-ABC	LB	−64	0
	UB	0	64
	Evaluation Number	10.000	160.000
	Limit	10	500
	P	10	100
	MR	0.02	0.5
	L2	0	0.1
MLP	Learning Rate	10^{-8}	10^{-1}
	Number of Hidden Units	2	40
	Batch Size	1	1024
	Number of Epochs	1	50
RF	Number of Trees	1	200
SVM	C	0.001	1
XGBoost	Eta	0.1	1
	Depth	1	40

zeros are. This metric indicates that max-abs normalization strategies should be used before classification. This max-abs normalization technique scales and transforms each feature independently, ensuring that each feature in the training set has a maximum absolute value of 1.0 and does not center or shift the values, hence preserving any sparsity. So, the max-abs normalization methods are applied to scale the feature values into the numeric range between 0 and 1.

3.5. Sequential model-based optimization configuration (SMAC)

In machine learning, there are numerous parameter optimization strategies that guarantee the model will achieve the best performance in the given space. For this reason, hyper-parameter selection is a critical procedure during training the model. The main advantage of hyper-parameter selection is that it is applicable to handling parameter tuning of many different models. The parameter tuning process has a strong impact on the performance or efficacy of a model, but this usually requires a large number of runs. This makes the tuning process time-consuming, which is the main disadvantage of hyper-parameter optimization [27]. The second disadvantage is that how to determine the parameter value is still challenging. SMAC optimization enables the search of a larger hyper-parameter space. For each parameter, the method accepts an interval (i.e., min and max values) and can consider any value within that interval. Another advantage is that SMAC optimization can be completed in a matter of days for the same search space and computational resources, whereas standard techniques can take up to a year [28]. According to the survey [27], the SMAC technique is currently the most powerful automatic parameter tuning method. Table 3 shows the hyper-parameter ranges for classification methods in this study.

4. Proposed LR-ABC method

4.1. Artificial bee colony (ABC) algorithm

The ABC algorithm proposed by Karaboga [9] is an optimization technique and simulates the foraging behavior of a honey bee population. There are three sorts of artificial bees in the ABC algorithm: employed, onlooker, and scout bees. A food source's position correlates to a viable solution. The nectar content of a food source indicates the solution's quality. The algorithm's objective is to locate the food source that contains the most nectar. The algorithm's steps are detailed in Algorithm 1.

Algorithm 1 ABC algorithm

1: Determine the number of food source (P), maximum evaluation number (MEN), limit and the number of parameters to be optimized (n)
2: Randomly create the food source locations
$for\ i \leftarrow to\ P:$
  $for\ j \leftarrow to\ n:$
    $w_j^{min} \leftarrow lowerbound$
    $w_j^{max} \leftarrow upperbound$
    $\phi_{ij} \leftarrow random(0,1)$
    $w_{ij} = w_j^{min} + \phi_{ij} \times (w_j^{max} - w_j^{min})$
3: Perform local searches around food source locations using employed bees
$for\ i \leftarrow to\ P:$
  $\phi \leftarrow random[-1,1]$
  $k \leftarrow randomInt[1,P]\ provided\ that\ i \neq k$
  $j \leftarrow randomInt[1,n]$
  $v_{ij} = x_{ij} + \phi \times (x_{ij} - x_{kj})$
4: Perform greedy selection
5: Calculate the fitness value for each food source; that is, evaluate the quality of each solution
$for\ i \leftarrow to\ P:$
  $f_i \leftarrow fitnessFunction(\vec{w})$
6: Calculate the probability value of each solution proportional to its quality
$for\ i \leftarrow to\ P:$
  $\rho_i \leftarrow 0.9 \times (\frac{f_i}{max(f)}) + 1$
7: Onlooker bees select food sources by considering probability values
$i \leftarrow 0$
$j \leftarrow 0$
$while\ t < P:$
  $if\ random(0,1) < p_i:$
    $\phi \leftarrow random[-1,1]$
    $k \leftarrow randomInt[1,P]\ provided\ that\ i \neq k$
    $j \leftarrow randomInt[1,n]$
    $v_{tj} = x_{ij} + \phi \times (x_{ij} - x_{kj})$
  $t \leftarrow t + 1$
  $i \leftarrow i + 1$
  $if\ i \geqslant P:$
    $i \leftarrow 0$
8: Perform greedy selection
9: Scout bee phase:
If there is an exhausted food source i then:
  $for\ j \leftarrow 0\ to\ N:$
    $\phi_{ij} \leftarrow random(0,1)$
    $w_{ij} = w_j^{min} + \phi_{ij} \times (w_j^{max} - w_j^{min})$

Employed bees are responsible for memorizing better regions around food source placements; employed bees also share this information about nectar quantities and food source location with onlooker bees waiting in the dance area. Each onlooker bee chooses a source to fly to, by observing the employed bees execute their dance. The ABC algorithm mimics the dance and profitable source selection using a stochastic selection strategy similar to that of a roulette wheel. The stochastic approach incorporates positive feedback, such that when a food source's nectar content is high, a greater number of onlooker bees is recruited to the source.

If the nectar content of a new solution is greater than that of the present solution, the employed bee saves the new solution in its memory and forgets about the old solution, during the greedy selection conducted in stages 4 and 8. Otherwise, it maintains the current answer in memory and increments the counter associated with it by one. The counters keep track of how many times a food source has been exploited and are used during the scout bee phase to determine when a food source has been exhausted. A solution is considered exhausted if its counter surpasses a predefined value, referred to as the limit. At most one employed bee can become a scout bee during each cycle of the basic algorithm. If more than one employed bee's food source is exhausted, the algorithm selects the source with the greatest counter value and converts this bee into a scout bee. Bees forsake exhausted sources and scout bees look for undiscovered sources to replace them.

Algorithm 2 Proposed LR-ABC classification method

1: Determine the input parameters: Input matrix $X_{M\times N}$, target $\vec{y}_M$, number of food sources P, position of the food sources $W_{P\times D}$, maximum evaluation number MEN, lower bound lb, upper bound ub, $limit$
Output:

$D \leftarrow N+1$
$W_{P\times D} \leftarrow lb + rand(P,D)\times(ub-lb)$
$W' \leftarrow W$
$\vec{fit} \leftarrow CalculateFitness(W)$
$\vec{\tau} \leftarrow zeros(P)$
$evaluation\_number \leftarrow 0$
while $evaluation\_number < MEN$ **do**
  Perform employed bee phase
  $s\vec{fit} \leftarrow CalculateFitness(W')$
  $\vec{ind} \leftarrow s\vec{fit} > \vec{fit}$
  $r\vec{ind} \leftarrow s\vec{fit} \leq \vec{fit}$
  $\vec{\tau}[\vec{ind}] \leftarrow 0$
  $W[\vec{ind}] \leftarrow W'[\vec{ind}]$
  $\vec{fit}[\vec{ind}] \leftarrow s\vec{fit}[\vec{ind}]$
  $\vec{\tau}[r\vec{ind}] \leftarrow \vec{\tau}[r\vec{ind}] + 1$
  Calculate probability values of all solutions
  Perform onlooker bee phase
  $s\vec{fit} \leftarrow CalculateFitness(W')$
  for $i \leftarrow 1 : P$ **do**
    $t \leftarrow tmp\vec{I}D[i]$
    if $s\vec{fit}[i] > \vec{fit}[t]$ **then**
      $\vec{\tau}[t] \leftarrow 0$
      $W[t,:] \leftarrow W'[i,:]$
      $\vec{fit}[t] \leftarrow s\vec{fit}[i]$
    else
      $\vec{\tau}[t] \leftarrow \vec{\tau}[t] + 1$
    end if
  end for
  Perform scout bee phase
  Memorize best source
end while
Return global best solution

Algorithm 3 Calculate fitness function

procedure $CalculateFitness(\phi)$
  $w \leftarrow \phi[:,1:]$
  $b \leftarrow \phi[:,0]$
  $p \leftarrow \sigma(X.dot(w^T)+b)$
  $f \leftarrow mean((\vec{y_M}-p)^2, axis=0)$
  $f \leftarrow 1/(f+1)$
  $evaluation\_number \leftarrow evaluation\_number + len(f)$
  $return \quad f$
end procedure

4.2. LR-ABC classification method

Due to the limitations of the gradient descent algorithm used in LR, such as the assumption of a continuous cost function, the ABC algorithm, a successful heuristic, is employed to train the LR model. In addition to making no assumptions about the function or parameter search space, the ABC algorithm can successfully search for both local and global solutions in the search space. When LR is trained using ABC, the LR model's weights correspond to the ABC algorithm's food source locations. As a result, the method initially generates a population of starting weights and bias values. The employed bee, onlooker bee, and scout bee phases then seek the optimal weight set ($\vec{w}_i$) and bias value that minimizes the mean squared error at the model's output. The algorithm's steps are provided in Algorithm 2.

After a training set is given $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}, y_i \in \{0,1\}$ and $\vec{x}_i \in Rn$, $1 \leqslant i \leqslant m$, LR-ABC classification model determines the class of the vector $\vec{x}_i$ by using Eq. (5):

$$p_i' = \begin{cases} 0, & p_i < 0.5 \\ 1, & p_i \geqslant 0.5 \end{cases} \tag{5}$$

where p_i is computed as seen in Eq. (6) and the function σ corresponds to sigmoid function, which is given in Eq. (7). The LR-ABC method's objective is to find the weights ($\vec{w}$) that minimize the cost function, which is provided by Eqs 8. As can be seen from Eq. (8), the cost function includes the mean square error (MSE) function with ridge regression (L2 regularization) used to avoid overfitting.

$$p_i = \sigma(w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in} + b) \tag{6}$$

$$\sigma(\theta) = \frac{1}{1+e^{-\theta}} \tag{7}$$

$$C(\vec{w}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - p_i)^2 + \frac{\lambda}{n}\sum_{j=1}^{n}(w_j)^2 \tag{8}$$

The LR-ABC algorithm first randomly generates a total of P food source positions using as described in step 2 of Algorithm 2. Each food source position corresponds to a weight vector. For each set of weight vectors, the LR-ABC model computes an output based on these weights and bias value. The fitness value of a solution is inversely proportional to the value returned by the cost function for that solution, as given in Eq. (9). Therefore, a solution with a higher cost value will have a lower fitness value. This fitness function and selection operator direct the algorithm to the better locations in the search space and Algorithm 3 demonstrates the calculation of the fitness value.

$$f_l = \frac{1}{1+C_l} \tag{9}$$

After evaluating the initial population, the algorithm begins iterating through the employed bee, onlooker bee, and scout bee phases until the termination requirements are met. In the employed bee phase, a local search around each solution in memory is performed, and a new solution is produced using the third step of Algorithm 1. In Algorithm 2, $\vec{\tau}$ is a vector that keeps track of the number of times each solution has failed to be improved, and in the scout bee phase, if there is a solution in this vector that is higher than the limit value, this solution is replaced with a new one.

4.3. Computation on GPU

The CPU version of the LR-ABC and state-of-the-art ML methods are implemented using the NumPy [29] library, which does not support GPU computations. NumPy is widely used in ML methods, and the Python community has built packages like Scikit-Learn on top of NumPy. With the rapid development of GPU technologies over the last few years, researchers have increasingly focused on parallel computing to accelerate algorithms. However, the CPU version of the LR-ABC method still has a limitation due to its high computational

time. Due to the large size of the training datasets (175,341 × 199 and 125,973 × 122 matrices for UNSW-NB15 and NSL-KDD, respectively), accelerating our model became imperative. Specifically, by harnessing GPU parallelization for vectorized loops, we significantly enhanced overall speed and efficiency. Additionally, we parallelized common array operations such as searching, comparing, addition, subtraction, and matrix multiplications on a GPU. We achieved this using CuPy [30], an open-source library that accelerates matrix operations using NVIDIA GPUs. CuPy is compatible with NumPy and enables full use of modern GPU capabilities through a NumPy-compatible interface. The improved parallel LR-ABC method was developed with the CuPy library. As a result, as shown in the performance results in Section 6, the training time has been dramatically reduced by approximately 4.5 times compared to CPU version of LR-ABC. Detailed implementation of the GPU version can be accessed in the references [31,32].

5. Experiments

In our experiment, eleven classification algorithms – DT, LDA, LR, MLP, RF, SVM, XGBoost, DNN, LSTM, GRU, and the proposed LR-ABC method – were evaluated using the publicly available UNSW-NB15 and NSL-KDD datasets. In the preprocessing steps for both datasets, the one-hot encoding technique was applied to transform categorical features into numeric values. Max-abs normalization methods were used to map the numeric feature values into the range 0 to 1. The classification methods were implemented using the SMAC optimization technique, and optimal hyper-parameters were determined for each classifier with the run count limit set to 20. Table 3 shows the hyper-parameter ranges for classification methods for UNSW-NB15 and NSL-KDD datasets. Individual results in terms of accuracy, FPR, FNR, and F1-measure were obtained for each classifier. Since both the UNSW-NB15 and NSL-KDD datasets contain separate training and test sets, each model was trained on the training set and evaluated on the test set.

The DNN, LSTM, and GRU architectures consist of an input layer, three hidden layers, and an output layer. The number of neurons in the hidden layers is determined through SMAC optimization. The hidden layers have a ReLU activation function, and the last layer has a sigmoid activation function. To prevent overfitting, batch normalization is implemented after each layer. The training process utilizes the Adam optimizer with the binary crossentropy loss function. Additionally, an early stopping mechanism stops model training if the validation loss does not improve after five epochs, at which point the best model weights are reinstated. The learning rate and mini-batch size are also determined through SMAC optimization, with the maximum number of epochs being set to 100.

All machine learning models were implemented using the Scikit-Learn library [33], while all deep learning models were implemented using Keras [34]. The proposed approach was developed in the Python programming language [35].

6. Performance results

In this study, we experiment the proposed LR-ABC method alongside seven different machine learning methods (DT, LDA, LR, MLP, RF, SVM, and XGBoost) and three different deep learning methods (DNN, LSTM, GRU) for anomaly detection in network traffic using two publicly available NIDS datasets. Each classifier is trained with hyper-parameter optimization using the SMAC technique.

Performance results of the proposed and other classification methods with optimum hyper parameters found by SMAC optimization are shown in Table 4. Hyper-parameter ranges for classification methods and optimum parameters found by SMAC optimization are shown in Table 5. For the UNSW-NB15 dataset, the proposed LR-ABC method achieved the highest accuracy (88.25%), F1-measure (87.86%) and the lowest FPR (0.1212) with the following optimum hyper-parameters: lower bound (LB) = −20, upper bound (UB) = 10, Evaluation Number

Table 4
Performance results of the proposed and other classification methods with optimum hyper-parameters found by SMAC optimization on UNSW-NB15 and NSL-KDD datasets.

Dataset	Classifier	Accuracy (%)	False Positive Rate	False Negative Rate	F1-measure (%)
UNSW-NB15	DT	86.81	0.2642	0.0237	86.5
	LDA	80.91	0.4218	**0.0023**	79.77
	LR	80.89	0.3892	0.0292	80.04
	LR-ABC	**88.25**	**0.1212**	0.1143	**88.26**
	MLP	83.37	0.3500	0.0162	82.72
	RF	86.79	0.2679	0.0211	86.47
	SVM	81.59	0.4050	0.0035	80.58
	XGBoost	86.93	0.2650	0.0210	86.62
	DNN	87.81	0.2492	0.0174	87.54
	LSTM	85.86	0.0161	0.2946	85.45
	GRU	84.64	0.3233	0.0150	84.10
NSL-KDD	DT	82.42	0.0310	0.2851	82.40
	LDA	78.70	0.0303	0.3511	78.51
	LR	75.67	0.0747	0.3707	75.49
	LR-ABC	**90.11**	0.0756	**0.1163**	**90.15**
	MLP	85.86	0.0778	0.1894	85.93
	RF	84.02	0.0314	0.2569	84.03
	SVM	76.53	0.0733	0.3567	76.40
	XGBoost	81.46	**0.0282**	0.3042	81.39
	DNN	81.75	0.0475	0.2845	81.74
	LSTM	81.75	0.0374	0.2922	81.71
	GRU	84.12	0.1398	0.1729	84.19

Table 5
Optimum parameters found by SMAC optimization on UNSW-NB15 and NSL-KDD Datasets.

Classifier	Parameter	UNSW-NB15	NSL-KDD
DT	M. Samples Split	99	19
	M. Samples Leaf	42	3
DNN	Batch Size	55	60
	Learning Rate	0.0052	0.0040
	Neuron1	37	9
	Neuron2	29	84
	Neuron3	127	50
LSTM	Batch Size	31	20
	Learning Rate	0.0028	0.0013
	Neuron1	115	73
	Neuron2	74	38
	Neuron3	118	94
GRU	Batch Size	44	61
	Learning Rate	0.0062	0.0036
	Neuron1	28	95
	Neuron2	122	94
	Neuron3	46	73
LDA	Shrinkage	5.26e−05	0.8579
LR	C	50 000	48 214.53
LR-ABC	LB	−20	−2
	UB	10	46
	Eva	77 885	48 296
	Limit	141	196
	P	15	10
	MR	0.0100	0.40421
	L2	2.836e−05	0.0628
MLP	Learning Rate	0.0714	0.7574
	# of Hidden Unit	22	13
	Batch Size	527	603
	# of Epochs	10	47
RF	# of Trees	173	4
SVM	C	0.9965	0.7331
XGBoost	Eta	0.2305	0.2148
	Depth	37	4

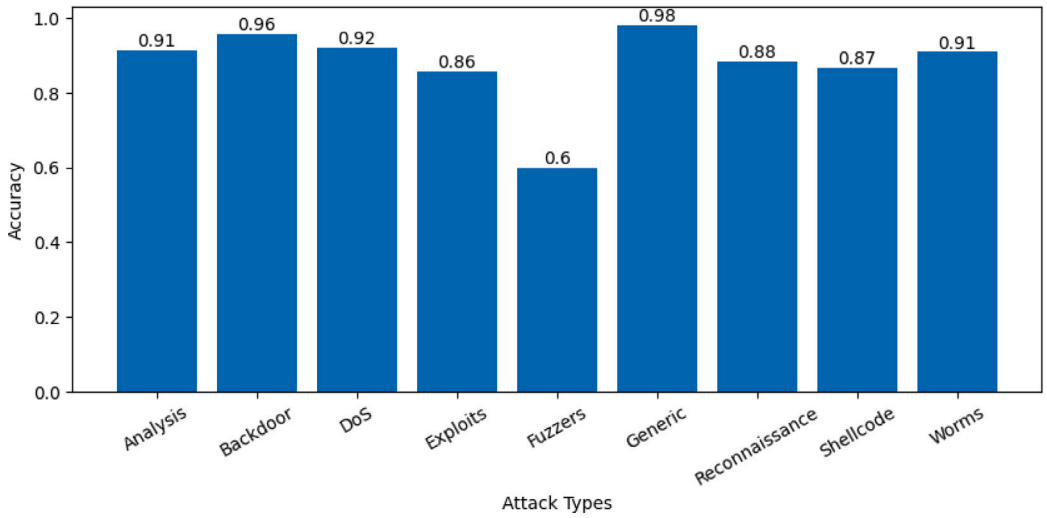

Fig. 1. The proposed LR-ABC method's accuracy for each different attack type on the UNSW-NB15 dataset.

Table 6
The training time of each classifier in seconds on UNSW-NB15 dataset.

Classifier	Best	Worst	Mean	Std.
DT	3.11	3.04	3.09	0.88
LDA	7.57	7.65	7.60	0.02
LR	6.69	6.75	6.73	0.02
LR-ABC (CPU)	541.61	542.51	541.91	0.25
LR-ABC (GPU)	121.38	122.41	121.56	0.28
MLP	436.60	790.39	534.09	109.93
RF	165.30	168.69	166.83	1.12
SVM	1437.91	1559.12	1505.68	53.15
XGBoost	120.21	120.96	120.47	0.25
DNN	129.91	391.74	246.37	82.38
LSTM	827.90	2322.50	1637.03	517.88
GRU	384.75	1130.84	716.75	309.28

= 77885, Limit = 141, population size (P) = 15, mutation rate (MR) = 0.0100, L2 Regularization (L2) = 2.8368, and with a threshold = 0.8. The class-based performance analysis for the UNSW-NB15 dataset is shown in Fig. 1. The accuracy for analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms was 91%, 96%, 92%, 92%, 86%, 60%, 98%, 88%, 87%, and 91%, respectively.

For the NSL-KDD dataset, the proposed method obtained the highest accuracy (90.11%), F1-measure (90.15%), and the lowest FNR (0.1163) with the following optimum hyper-parameters: LB = −2, UB = 46, Evaluation Number = 48296, Limit = 196, P = 10, MR = 0.40421, L2 = 0.06280, and with a threshold = 0.8. The class-based performance analysis for the NSL-KDD dataset is shown in Fig. 2. The accuracy for Dos, R2L, Probe, and U2R was 94%, 73%, 92%, and 69%, respectively.

The CPU and GPU versions of the LR-ABC algorithm are implemented using the Numpy and CuPy libraries, respectively, to improve training time. The training time for each classifier was recorded by running the models 10 times. As shown in Table 6, the average training times for DT, LDA, LR, CPU version of LR-ABC, GPU version of LR-ABC, MLP, RF, SVM, XGBoost, DNN, LSTM, and GRU are 3.09 s, 7.60 s, 6.73 s, 541.91 s, 121.56 s, 534.09 s, 166.83 s, 1505.68 s, 120.47 s, 246.37 s, 1637.03 s, and 716.75 s, respectively. Machine learning algorithms were run on a CPU, while deep learning methods were executed on a GPU. The training time of the proposed GPU version of LR-ABC is faster than that of MLP, RF, SVM, DNN, LSTM, GRU, and is comparable to the XGBoost classifier, with only about a one-second difference.

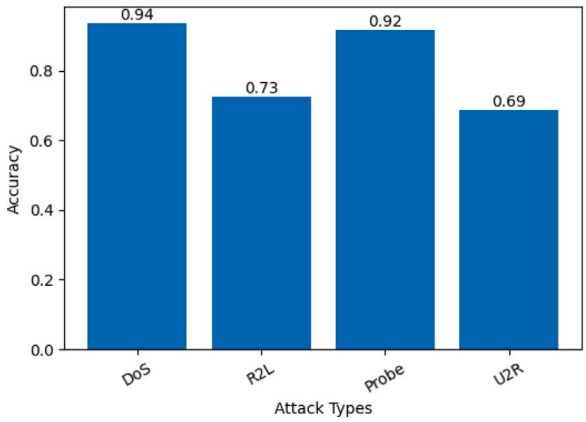

Fig. 2. The proposed LR-ABC method's accuracy for each different attack type on the NSL-KDD dataset.

7. Conclusion

Machine learning-based NIDS face several challenges, many of which arise from the nature of network traffic, such as high dimensionality, class imbalance, noise, and outliers. Addressing these challenges often requires advanced ML techniques capable of learning complex features from data. In this study, an efficient approach is proposed based on the LR model trained by a parallel ABC algorithm, which performs local and global searches in the solution space enabling the learning of highly nonlinear and high-dimensional data due to its training algorithm. The LR-ABC algorithm demonstrates high-performance results, however, it suffers from long training times. This represents a significant challenge particularly in the context of cybersecurity. Fast training allows for frequent model updates and iterations, ensuring that the system can rapidly adapt to the dynamic nature of network traffic and intrusion tactics. The ability to quickly retrain models is crucial for maintaining up-to-date defenses against new and evolving threats. To overcome the high computational time associated with LR-ABC models, CPU and GPU versions have been developed, significantly reducing training time. To the best of our knowledge, this is the first study propose an anomaly-based NIDS approach using the parallel ABC algorithm as the learning mechanism for an LR classification model.

Furthermore, the proposed approach outperforms state-of-the-art ML and DL models in terms of accuracy, FPR, and F1-measure on the UNSW-NB15 dataset, and in terms of accuracy, FNR, and F1-

measure, on the NSL-KDD dataset. Additionally, the training time of the proposed approach is faster than that of MLP, RF, and SVM, and is comparable to the XGBoost classifier, differing by only one second. The high performance of the proposed approach demonstrates that the proposed model is reliable and robust in detecting network intrusions.

Overall, this research provides a fresh perspective on NIDS, and demonstrates that proposed LR-ABC algorithm maintains high-quality results for anomaly-based NIDS. This approach not only simplifies the model's application but also opens avenues for future research where real-world, raw data can be used directly to secure networks against the ever-evolving array of cyber threats. Although this study focused on binary classification, the class-based performance analysis revealed that the model's classification performance for some attack types was low. Therefore, future studies focused on multi-class classification to improve the performance of these low-performing classes would be beneficial.

CRediT authorship contribution statement

Burak Kolukisa: Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Bilge Kagan Dedeturk:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Hilal Hacilar:** Writing – original draft, Analysis and/or interpretation. **Vehbi Cagri Gungor:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors certify that they have NO affilations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers bureaus; membership, employement, consultancies, stock ownership, or other equity interest; and expert tetimony 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

The datasets used in the study are publicly available and we have shared the codes of the proposed method.

References

[1] S. Kemp, Digital 2019: Global digital overview, 2019, URL https://datareportal.com/reports/digital-2019-global-digital-overview. (Accessed 20March 2022).

[2] 2021 Cyber threat report, 2021, URL https://www.sonicwall.com/medialibrary/en/white-paper/2021-cyber-threat-report.pdf. (Accessed 20March 2022).

[3] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: Proceedings of the 2010 IEEE Symposium on Security and Privacy, 2010, pp. 305–316, http://dx.doi.org/10.1109/SP.2010.25.

[4] A. Thakkar, R. Lohiya, Role of swarm and evolutionary algorithms for intrusion detection system: A survey, Swarm Evol. Comput. 53 (2020) 100631, http://dx.doi.org/10.1016/j.swevo.2019.100631.

[5] T. Bäck, D.B. Fogel, Z. Michalewicz, Evolutionary Computation 1: Basic Algorithms and Operators, CRC Press, 2018.

[6] V.R. Balasaraswathi, M. Sugumaran, Y. Hamid, Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms, J. Commun. Inf. Netw. 2 (4) (2017) 107–119, http://dx.doi.org/10.1007/s41650-017-0033-7.

[7] T.R. Peltier, Information security policies, procedures, and standards: Guidelines for effective information security management, CRC Press, 2016.

[8] Y. Han, M. Yang, H. Qi, X. He, S. Li, The improved logistic regression models for spam filtering, in: Proceedings of the 2009 International Conference on Asian Language Processing, 2009, pp. 314–317, http://dx.doi.org/10.1109/IALP.2009.74.

[9] D. Karaboga, An idea based on honey bee swarm for numerical optimization, (TR06) Erciyes University, Engineering Faculty, Computer Engineering Department, 2005, pp. 1–10.

[10] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132, http://dx.doi.org/10.1016/j.amc.2009.03.090.

[11] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142, http://dx.doi.org/10.1016/j.ins.2010.07.015.

[12] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, Appl. Sci. 9 (20) (2019) 4396, http://dx.doi.org/10.3390/app9204396.

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

[14] A.U.H. Qureshi, H. Larijani, N. Mtetwa, A. Javed, J. Ahmad, RNN-ABC: A new swarm optimization based technique for anomaly detection, Computers 8 (3) (2019) 59, http://dx.doi.org/10.3390/computers8030059.

[15] M. Mazini, B. Shirazi, I. Mahdavi, Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms, J. King Saud University - Computer and Information Sciences 31 (4) (2019) 541–553, http://dx.doi.org/10.1016/j.jksuci.2018.03.011.

[16] T. Gu, H. Chen, L. Chang, L. Li, Intrusion detection system based on improved ABC algorithm with tabu search, IEEJ Trans. Electr. Electron. Eng. 14 (11) (2019) 1652–1660, http://dx.doi.org/10.1002/tee.22987.

[17] M. Rani, Employing artificial bee colony algorithm for feature selection in intrusion detection system, in: Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development, INDIACom, 2021, pp. 496–500.

[18] M. Rani, Effective network intrusion detection by addressing class imbalance with deep neural networks, Multimedia Tools Appl. 81 (6) (2022) 8499–8518, http://dx.doi.org/10.1007/s11042-021-11747-6.

[19] B.K. Dedeturk, B. Akay, Spam filtering using a logistic regression model trained by an artificial bee colony algorithm, Appl. Soft Comput. 91 (2020) 106229, http://dx.doi.org/10.1016/j.asoc.2020.106229.

[20] S.M. Kasongo, Y. Sun, Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset, J. Big Data 7 (1) (2020) 1–20, http://dx.doi.org/10.1186/s40537-020-00379-6.

[21] S. Solani, N.K. Jadav, A novel approach to reduce false-negative alarm rate in network-based intrusion detection system using linear discriminant analysis, in: Proceedings of the Conference on Inventive Communication and Computational Technologies, 2021, pp. 911–921, http://dx.doi.org/10.1007/978-981-15-7345-3_77.

[22] S. Meftah, T. Rachidi, N. Assem, Network based intrusion detection using the UNSW-NB15 dataset, Int. J. Comput. Digital Syst. 8 (5) (2019) 478–487, http://dx.doi.org/10.12785/ijcds/080505.

[23] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Shallow neural network with kernel approximation for prediction problems in highly demanding data networks, Expert Syst. Appl. 124 (2019) 196–208, http://dx.doi.org/10.1016/j.eswa.2019.01.063.

[24] D. Jing, H.B. Chen, SVM based network intrusion detection for the UNSW-NB15 dataset, in: Proceedings of the 2019 IEEE 13th International Conference on ASIC, ASICON, 2019, pp. 1–4, http://dx.doi.org/10.1109/ASICON47005.2019.8983598.

[25] N. Moustafa, J. Slay, UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: Proceedings of the 2015 Military Communications and Information Systems Conference, MilCIS, 2015, pp. 1–6, http://dx.doi.org/10.1109/MilCIS.2015.7348942.

[26] M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani, A detailed analysis of the KDD cup 99 data set, in: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA, 2009, pp. 1–6, http://dx.doi.org/10.1109/CISDA.2009.5356528.

[27] C. Huang, Y. Li, X. Yao, A survey of automatic parameter tuning methods for metaheuristics, IEEE Trans. Evol. Comput. 24 (2) (2019) 201–216, http://dx.doi.org/10.1109/TEVC.2019.2921598.

[28] Y. Gormez, Z. Aydin, R. Karademir, V.C. Gungor, A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks, Int. J. Commun. Syst. 33 (11) (2020) e4401, http://dx.doi.org/10.1002/dac.4401.

[29] C.R. Harris, et al., Array programming with NumPy, Nature 585 (7825) (2020) 357–362, http://dx.doi.org/10.1038/s41586-020-2649-2.

[30] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, Crissman Loomis, CuPy: A numpy-compatible library for NVIDIA GPU calculations, in: Proceedings of the Workshop on Machine Learning Systems (LearningSys) At the 31st Conference on Neural Information Processing Systems (NIPS), 2017, URL http://learningsys.org/nips17/assets/papers/paper_16.pdf.

[31] PyPI : ABC-LR, 2022, URL https://pypi.org/project/abcLR/. (Accessed: 23 March 2022).

[32] GitHub : ABC-LR, 2022, URL https://github.com/kagandedeturk/ABC-LR. (Accessed: 23 March 2022).

[33] F. Pedregosa, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[34] F. Chollet, et al., Keras, 2015, https://keras.io.

[35] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

Burak Kolukisa received a B.S. in Computer Engineering from Erciyes University, Kayseri, Turkey, in 2016 and an M.S. degree in Electrical and Computer Engineering from Abdullah Gul University, Kayseri, Turkey, in 2020. Currently, he is working as a Research Assistant at the Department of Computer Engineering, Abdullah Gul University. His current research interests are Data Mining, Machine Learning, and Deep Learning. He is also a Ph.D. candidate in the Electrical and Computer Engineering program at Abdullah Gul University.

Bilge Kagan Dedeturk received his B.S. degree in Teaching Computer Systems from Gazi University, Ankara, Turkey, in 2010. He earned his M.S. and Ph.D. degrees in Electrical and Computer Engineering from Erciyes University, Kayseri, Turkey, in 2014 and 2020, respectively. He is currently an Assistant Professor in the Department of Software Engineering at Erciyes University. His research interests currently include machine learning, natural language processing, swarm intelligence, and meta-heuristics.

Hilal Hacilar received her Bachelor of Science degree in Computer Engineering from Erciyes University and graduated one term early in January 2012. She has started her doctoral education at Abdullah Gul University in 2017. She is a research assistant at the Department of Computer Engineering at Abdullah Gul University. Her research fields are in Anomaly Detection, Machine learning, Data mining, Deep Learning and Network Security.

V. Cagri Gungor received his B.S. and M.S. degrees in Electrical and Electronics Engineering from Middle East Technical University, 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 a Professor and Chair of Computer Engineering Department, Abdullah Gul University, Kayseri, Turkey. His current research interests are smart grid communications, machine-to-machine communications, next-generation wireless networks, wireless ad hoc and sensor networks, cognitive radio networks.