The Journal of Supercomputing (2025)81:856
https://doi.org/10.1007/s11227-025-07292-w

ABIDS-VEM: leveraging an equilibrium optimizer and data ramification in association with ensemble learning for anomaly-based intrusion detection system

Priyanka Verma[1] · Donna O'Shea[2,5] · Thomas Newe[3] · Nakul Mehta[4] · Nitesh Bharot[4] · John G. Breslin[4]

Accepted: 5 April 2025

Abstract

The convergence of the Internet of Things (IoT) and Industrial Internet of Things (IIoT) within the Industry 4.0 paradigm leverages software-defined networking, multi-cloud architectures, and edge/fog computing to enhance industrial processes. However, this digital transformation introduces significant cybersecurity and privacy vulnerabilities within the complex, data-intensive IoT/IIoT ecosystems. To mitigate these risks, this research proposes a novel Anomaly-based Intrusion Detection System using Voting-based Ensemble Model (ABIDS-VEM) in Industry 4.0 environments. The VEM architecture synergistically combines multiple machine learning algorithms and gradient boosting frameworks, including CatBoost (CB), XGBoost (XGB), LightGBM (LGBM), Logistic Regression (LR), and Random Forest (RF), to enhance the precision and computational efficiency of intrusion detection systems (IDS) in IoT/IIoT contexts. The proposed framework incorporates a data ramification process, in which the data is divided into multiple parts, feature selection process which is optimized through the Equilibrium Optimizer (EO) algorithm, and outlier detection utilizing the Isolation Forest (IF) method. Comprehensive empirical evaluations were conducted using three benchmark datasets: XIIoTID, NSL-KDD, and UNSW-NB15, to validate the efficacy of the proposed system. The model achieves high accuracy across datasets: 98.1476% for XIIoT-ID, an impressive accuracy of 98.9671% for NSL-KDD, and 94.1327% for UNSW-NB15 dataset. These experimental results demonstrate the potential of this approach to significantly enhance the resilience of critical industrial systems and data against evolving cyber threats, thereby supporting the continued evolution of Industry 4.0 technologies and bolstering the security posture of IoT/IIoT ecosystems. This research contributes to the ongoing efforts to secure the rapidly expanding digital industrial landscape, offering a robust solution for detecting and mitigating sophisticated cyberattacks in the increasingly interconnected and data-driven industrial environments of the future.

Keywords Industry 4.0 · Equilibrium optimizer · IIoT · Anomaly-based IDS · Ensemble learning

Published online: 12 May 2025

1 Introduction

The onset of Industry 4.0 has marked the commencement of a new era in advanced industrial processes, driven by the extensive adoption of the Internet of Things (IoT) and the Industrial Internet of Things (IIoT) [1, 2]. This technological revolution has heralded the amalgamation of state-of-the-art networking and computing paradigms, software-defined networking, multi-cloud platforms, edge computing, fog computing, and artificial intelligence, into traditional industrial settings. While these innovations have unlocked unprecedented opportunities for increased efficiency and productivity, they have also given rise to significant concerns regarding network security against various cyber threats and user privacy in complex and data-rich IoT ecosystems.

As the volume of data and the complexity of IIoT ecosystems continue to expand, there is a growing need for robust and effective security models [3–5] and intrusion detection systems (IDSs) [6, 7]. An IDS plays a critical role in ensuring the safety of computer networks and cyber systems by identifying and mitigating cyber threats. IDSs can be broadly classified into two types depending on their objectives: Signature-based IDS and Anomaly-based IDS.

A signature-based IDS operates by monitoring network traffic or system activity and identifying anomalies by comparing them to specific signatures or patterns that are previously kept in its memory. These signatures or patterns are derived from known attack profiles [8–10]. However, they require substantial storage capacity to maintain a comprehensive database of attack signatures.

On the other hand, anomaly-based IDS employs a distinct methodology, monitoring system, or network behavior, pinpointing deviations from established patterns of normal behavior. Any behavior that significantly deviates from this predefined normal behavior is marked as a potential threat or cyberattack [11]. This method offers the advantage of being capable of detecting new, previously unidentified attacks based on behavior rather than specific signatures. However, it is also more prone to false positives, as any unusual but normal activity can trigger alerts.

Despite the deployment of various machine learning (ML) algorithms for IDS aimed at improving detection accuracy [12], existing IDS methods continue to face challenges in achieving satisfactory results [13]. This limitation can be attributed to the predominant focus on individual classifiers, which often prove ineffective in terms of both accuracy and F1-score. Notable ML algorithms, including decision trees (DT) [14], support vector machines (SVM) [13], genetic algorithms (GA) [15], particle swarm optimization (PSO) [16], spider wasp optimization [17], binary quadratic interpolation optimization [18], statistical approaches (SA) [19], and various swarm intelligence (SI) techniques [20, 21], have been explored for constructing IDSs.

One of the fundamental limitations in the concept of utilizing ML for IDSs was the recognition that a single classifier might not possess the requisite power to effectively construct a robust IDS [22]. As a result, relying on one classifier to function adequately across all settings and scenarios has thus far proved unfeasible.

This challenge has led to the emergence of the need for ensemble classification techniques.

Ensemble learning techniques [23, 24] have been introduced to address various challenges in both ML and IDS [25]. It involves the concept of combining multiple classification models while making decisions, specifically in supervised ML tasks. The fundamental idea underlying ensemble learning is that the prediction errors of one classifier can be offset in combination with other classifiers while they are working together. This leads to the final prediction of an ensemble being more accurate than that of an individual classifier. It has been statistically demonstrated that combining multiple classifiers significantly reduces the probability of incorrect predictions in ensemble-based models [26].

Therefore, we propose anomaly-based intrusion detection system (ABIDS) that utilizes a voting-based ensemble model (VEM) in conjunction with an equilibrium optimizer (EO) [27] for a feature selection, and data ramification process to address imbalanced data, along with isolation forest (IF) for the detection of outliers. The VEM leverages various ML and boosting techniques [28], including logistic regression (LR) [29], random forest (RF) [30], LightGBM (LGBM) [31], CatBoost (CB) [32], and the eXtreme gradient boosting machine (XGB) [33]. Furthermore, we have assessed the performance of the IDS models on three distinct datasets, namely, XIIoTID [34], NSL-KDD [35], and UNSW-NB15 [36], evaluated the performance, and demonstrated the advantages of using our proposed ABIDS-VEM method. Finally, we evaluated the proposed framework using F1-score, accuracy, precision, recall, and Matthew Correlation Coefficient (MCC) metrics. Thus, the contributions of this paper are:

- The research introduces a novel feature selection approach using equilibrium optimization (EO). This advanced technique intelligently extracts the most relevant features from complex datasets, significantly enhancing the intrusion detection system's efficiency. By reducing data dimensionality while preserving critical insights, this method addresses a fundamental challenge in machine learning for cybersecurity: balancing computational efficiency with detection accuracy.
- The paper presents a sophisticated Voting Ensemble Model (VEM), integrating diverse machine learning and boosting algorithms including Logistic Regression, Random Forest, LightGBM, CatBoost, and XGBoost. This ensemble approach leverages the strengths of multiple algorithms to create a robust, high-performance intrusion detection system. The VEM's ability to combine these varied techniques represents a significant advancement in adaptive cybersecurity measures for Industry 4.0 environments.
- The research demonstrates exceptional versatility through rigorous testing across multiple datasets (XIIoTID, NSL-KDD, UNSW-NB15) that represent a wide array of network security scenarios. This comprehensive evaluation not only validates the ABIDS-VEM framework's effectiveness but also showcases its adaptability to diverse intrusion landscapes. Such versatility is crucial for developing intrusion detection systems capable of addressing the evolving and varied nature of cyber threats in modern networked environments.

The rest of the paper is organized as follows: Section 2 discusses the related work, followed by Section 3, which establishes pre-requisite knowledge. Section 4 describes the proposed framework, and an evaluation of its results is provided in Section 5. Lastly, Section 6 ends with conclusions and the scope for future work.

2 Related work

The growing demand for efficient network analytics solutions has encouraged numerous researchers to make significant contributions in their designs of IDS [37–40]. In response to an evolving landscape of security threats, these researchers have also introduced hybrid models that leverage ML techniques to enhance the detection of security attacks. Many initial IDS models were rule-based, involving a continuous scanning process to identify potential attacks and assess them against a predefined set of attack patterns. These rule-based systems served as the foundation for subsequent developments in intrusion detection technology. With this view, this section delves into the frameworks and methodologies designed to analyze network traffic and formulate IDS systems.

2.1 ML-based IDS systems

Astha et al. (2017) [41] proposed a hybrid approach named SVM-Classification and Regression Tree (CART). This approach combines the capabilities of an SVM with a regression tree algorithm. The researchers compared the performance of SVM-CART with the k-nearest neighbors (KNN) algorithm using the KDD dataset. While the hybrid SVM-CART algorithm shows potential for improving intrusion detection systems, its limitations regarding dataset dependency, scalability, comparative analysis, and false positive rates, show the need for further improvement.

Guo et al. [42] introduced a two-level hybrid intrusion detection approach, combining misuse & anomaly detection to achieve a high detection rate and a low False positive rate (FPR). The first stage employs a low-complexity anomaly detection method, while the second stage uses the KNN algorithm. These components work in tandem to minimize false positives and false negatives. Experimental results on the KDD'99 dataset and Kyoto University Benchmark dataset validate the effectiveness of this hybrid approach in detecting network anomalies with a low FPR. Concerns about the computational complexity of the hybrid approach, particularly the scalability of the k-NN algorithm in stage 2, and points out limitations in the analysis of false positive/negative rates across various attack scenarios.

Contemporary Smart Power Networks (SPNs) rely on cyber-physical systems (CPSs) for connectivity and control, making them vulnerable to cyberattacks. Khan et. al. [43] presented a privacy-conserving intrusion detection system (PC-IDS) designed to protect SPNs from attacks like data poisoning. The framework includes data preprocessing for privacy and an intrusion detection component using a particle swarm optimization-based probabilistic neural network. Tested on Power System and UNSW-NB15 datasets, PC-IDS demonstrated superior performance with

detection rates of 96.03% and 95.91%, and false positive rates of 0.18% and 0.14%, respectively, outperforming existing methods in efficiency and accuracy.

Sara et al. (2019) [44] demonstrated an intrusion detection approach which incorporates feature selection & clustering algorithms. This approach employs both filter and wrapper methods for feature selection and utilizes a decision tree as the base classifier. Their findings demonstrated exceptional performance, achieving an accuracy of 95.03% and a notably low FPR of 1.65%. However, the study lacks comparison with recent advancements like deep learning-based approaches in intrusion detection, potentially overlooking improvements in accuracy and efficiency.

Gite et al. [45] highlighted the importance of IDS in maintaining the integrity of wireless sensor networks (WSN). It introduces a system that effectively detects common network attacks (black hole, wormhole, gray hole, and DDoS) using a Base Station ML algorithm (BS). Their method employed pattern analysis to identify malicious nodes and achieve high accuracy in simulations, which can improve energy efficiency and packet delivery in WSNs. As the proposed scheme's use of C4.5 and CART classifiers may be computationally demanding for wireless sensor networks (WSNs), which typically operate under strict resource constraints. This mismatch between the algorithm's requirements and the limited processing power, memory, and energy of many sensor nodes could restrict the practical deployment of the system in resource-constrained WSN environments.

These ML models aim to effectively produce good-performing IDS; however, they face significant challenges in real-world applications. The hybrid approaches, while promising, struggle with dataset dependency, scalability issues, and high computational demands. Furthermore, the lack of comprehensive comparisons with recent advancements, such as deep learning-based methods, and insufficient analysis of false positive/negative rates across diverse attack scenarios limit their proven effectiveness. Striking a balance between retaining valuable information and reducing computational burden is essential for effective model training and improved accuracy, particularly in scenarios with extensive or intricate data sets.

2.2 Ensemble learning-based IDS systems

There have been multiple ensemble models employed for enhancing the security front [46]. A dual ensemble model, joining bagging & gradient boosting decision tree (GBDT), offering a competitive solution was proposed by Louk et al. [47]. The Bagging-GBM combination stands out, achieving superior results among various schemes and comparable techniques in the field.

Wang et al. [48] introduced the Ensemble Feature Selection-based Deep Neural Network (EFS-DNN) for enhancing attack detection in high-traffic networks. It leverages a Light Gradient Boosting Machine (LightGBM) for robust feature selection & employs a deep neural network with batch normalization & embedding techniques for classification. Extensive experiments on public datasets highlight the EFS-DNN's superiority over baseline methods in intrusion detection. Although the use of LightGBM as the base selector enhances robustness, the overall robustness of the EFS-DNN model in various real-world scenarios remains to be fully validated.

The paper does not extensively discuss how the model performs under different types of cyberattacks or varying network conditions, which could affect its reliability.

A tree-based Stacking Ensemble Technique (SET) for IDS was proposed by Rashid et al. [49] and tested on NSL-KDD and UNSW-NB15 datasets. Results indicate that their SET model excels in distinguishing normal and anomaly network traffic, demonstrating its potential for enhancing cybersecurity in IoT and large-scale networks. However, the single classifiers may not perform well for large-scale data. Also complexity of intrusion analysis is increasing exponentially due to data size.

El et. al. [50] presented an innovative intrusion detection system combining SHAP-based feature selection, the PV-DM shallow learning algorithm, and XGBOOST for classification. Tested on NSL-KDD and UNSW-NB15 datasets, the model achieves impressive results with 82.86% accuracy on UNSW-NB15. However, limitations include potential overfitting to the specific datasets used, the need for validation on more diverse and real-time data, and the challenge of adapting to rapidly evolving attack patterns.

The proliferation of wireless devices and increased network traffic create a heightened risk of network intrusion and security threats. Traditional packet-based IDS struggle with high-speed and encrypted traffic. To address this, a cortex-inspired ensemble-based network IDS (CI-EnsID) was proposed in [51]. It combines ensemble classification with cortex-like information processing, utilizing both unsupervised and supervised learning techniques. Testing on KDDCup99 and CICIDS2017 datasets demonstrates the CI-EnsID's superior performance compared to contemporary classification techniques like NB, Neural Networks (NN), LR, DT, and RFs. While the CI-EnsID system is designed to minimize overhead by relying on network flow information, there is still a risk of resource consumption, especially if the system is not optimized for specific network conditions. This could lead to performance degradation in resource-constrained environments.

[52] presented an anomaly-based intrusion detection system utilizing a one-dimensional convolutional neural network (CNN) trained on the NSL-KDD dataset. The proposed model outperforms existing methods in accuracy, precision, and F1-score, particularly in detecting minority attack classes. However, limitations include potential overfitting, as indicated by performance discrepancies between training and testing datasets, and reliance on a single benchmark dataset, which may affect generalizability to other environments.

[53] addressed the growing complexity of cyberattacks by proposing an innovative intrusion detection system that combines feature selection and adaptive voting. Applied to the widely used NSL-KDD dataset, it achieves 86.5% accuracy. The results underscore the potential of this approach in advancing cybersecurity research and practice. However, it is worth noting that it only focuses solely on the NSL-KDD dataset, which may limit its broader applicability; the 86.5% accuracy, while high, still leaves room for missed intrusions; and the model's performance in real-time environments remains unexplored.

Uddin et. al [54] proposed two semi-supervised learning strategies: using synthetic attack samples for supervised learning and employing One Class

Classification (OCC) trained solely on benign traffic. Experiments across 10 benchmark datasets reveal that the OCC model, particularly using the usfAD technique, outperforms conventional supervised methods and other OCC approaches in detecting unknown attacks. However, limitations include the need for further exploration of hierarchical multi-classification for specific attack types, potential overfitting to the datasets used, and the challenge of generating representative synthetic attack samples across feature spaces.

3 Preliminaries

This section establishes the prerequisites for the ABIDS-VEM such as EO, its parameters, and ensemble learning.

3.1 Equilibrium optimizer (EO)

The equilibrium optimizer (EO) [27] is a physics-inspired optimization algorithm designed for addressing continuous optimization challenges. Some optimization features of the EO algorithm include its capability to incorporate randomness into the solution space, which enables more exploration and exploitation. In the EO algorithm, particles and their concentrations bear a resemblance to the particles and positions found in a PSO, serving as representative search agents. These search agents dynamically adjust their concentrations based on the best solutions encountered thus far, known as equilibrium candidates. This iterative process ultimately leads them to attain an equilibrium state, representing optimal result [27].

The EO approach employs a mass balance [55] equation to depict the concentration of a non-reactive component within a control volume, taking into account various mechanisms that contribute to its sources and sinks. The mass balance equation forms the physical basis for conserving mass within the control volume, encompassing mass inflow, outflow, and generation. This equation is commonly represented as,

$$V\frac{dQ}{dt} = V_f Q_{\text{eq}} - V_f Q + M \tag{1}$$

Where Q denotes the concentration within the control volume V, and the term $V\frac{dQ}{dt}$ signifies the rate of change of mass within this control volume. Additionally, V_f represents the volumetric flow entering and exiting the control volume, while Q_{eq} is indicative of the equilibrium state concentration, characterized by the absence of any generation within the control volume. The variable M designates the rate of mass generation under the control volume. Upon reaching a state where $V\frac{dQ}{dt}$ equals zero, a steady equilibrium condition is attained. By rearranging Eq. (1), it becomes feasible to compute $\frac{dQ}{dt}$ as a function of $\frac{V_f}{V}$, which corresponds to the inverse of residence time or the turnover rate, here referred to as δ. Furthermore, Eq. (1) can also be rearranged to determine the value of Q, the concentration within the control volume, as a function of time (t):

$$\frac{dQ}{\delta Q_{\text{eq}} - \delta Q + \frac{M}{V}} = dt \tag{2}$$

On integrating with respect to time we get,

$$\int_{Q_0}^{Q} \frac{dQ}{\delta Q_{\text{eq}} - \delta Q + \frac{M}{V}} = \int_{t_0}^{t} dt \tag{3}$$

which gives,

$$Q = Q_{\text{eq}} + (Q_0 - Q_{\text{eq}}E + \frac{M}{\delta V}(1 - E)) \tag{4}$$

On calculating E from Eq. (4), we find,

$$E = e^{-\delta(t-t_0)} \tag{5}$$

Here, t_0 and Q_0 represent the initial starting time and concentration, respectively, and they are dependent on the integration interval. Equation (4) serves a dual purpose: it can be employed to either estimate the concentration within the control volume when the turnover rate is known, or it can be utilized to calculate the average turnover rate. This calculation can be carried out through a linear regression analysis when the generation rate and relevant conditions are known.

The initial concentrations Q are generated by uniformly initializing the number of particles and dimensions in search space using random values:

$$Q_p^{\text{initial}} = Q_{\min} + \text{random}_i(Q_{\max} - Q_{\min}) \tag{6}$$

p = 1, 2 3,....n

The vector Q_i^{initial} represents the initial concentration of p^{th} particle, while $Q_{\min}$ and $Q_{\max}$ denote the minimum and maximum values within which the dimensions of random_i vary, with random_i falling within [0, 1]. Here, n corresponds to the population size, signifying the number of particles. The particles undergo evaluation based on their fitness function, following which they are sorted to identify potential equilibrium candidates.

A precise definition of E plays a pivotal role in assisting EO to maintain a well-considered equilibrium between exploration and exploitation. Given that the turnover rate could exhibit temporal variations within an actual control volume, it is considered that δ is a random vector $\in$ [0,1].

$$\vec{E} = e^{-\vec{\delta}(t-t_0)} \tag{7}$$

where time t is confined under iterations "itr" and thus decreases with number of iterations:

$$t = (1 - \frac{\text{itr}}{\text{max\_itr}})^{(c\frac{itr}{\text{max\_itr}})} \tag{8}$$

where c is a constant value used to manage exploitation ability.

M is the generation rate and it could be expressed using many models [56]. For example, a versatile model that characterizes M as a first-order exponential decay process is defined as:

$$\vec{M} = \vec{M_0} e^{-\vec{\delta}(t-t_0)} = \vec{M_0}\vec{E} \tag{9}$$

$$\vec{M_0} = \vec{CP}(\vec{Q_{eq}} - \vec{\delta}\vec{Q}) \tag{10}$$

where M_0 is initial value & δ is alias of decay constant.

$$\vec{CP} = 0.5 rn_1 \quad if\ rn_2 \geq GP \tag{11}$$

$$\vec{CP} = 0 \quad if\ rn_2 < GP \tag{12}$$

where rn_1 and rn_2 represent random numbers within [0,1] and the CP vector is constructed by repeating the same value obtained from eqs. 11 and 12. In these equations, CP serves as the control parameter for the generation rate, encompassing the contribution of the generation term for the updating process. The probability of this contribution, which dictates how many particles utilize the generation term for updating their states, is determined by another parameter known as generation probability (GP), as defined by eqs. 10, 11, and 12. A well-balanced trade-off between exploration and exploitation is obtained when GP is set to 0.5. Consequently, the update rule for EO is formulated as follows:

$$\vec{Q} = \vec{Q_{\text{eq}}} + (\vec{Q} - \vec{Q_{\text{eq}}}).\vec{E} + \frac{\vec{M}}{\vec{\delta} V}(1 - \vec{E}) \tag{13}$$

The initial component in Eq. (13) signifies the equilibrium concentration, while the subsequent two terms account for changes in concentration.

3.2 Ensemble learning

Ensemble learning, a technique in ML, amalgamates the forecasts of numerous individual models, known as base learners, to enhance overall prediction accuracy. The rationale behind ensemble learning lies in aggregating the predictions of diverse models to offset the shortcomings of individual models and bolster their combined performance. This can be achieved through techniques like bagging, boosting, and stacking. Bagging methods involve creating diverse subsets of training data and independently training multiple models, while boosting techniques assign different weights to data points, emphasizing the ones that were previously misclassified. Stacking combines the outcomes of various models into a final model. Every general framework of ensemble learning uses some aggregation function, say A_g, to combine B number of base classifiers such that: $b_i \in B$ for final prediction. For a dataset size d, number of features 'f', and real numbers R, dataset = $\{(X_i, y_i)\}, 1 \leq i \leq d, X_i \in R^f$, the prediction based on this is given using,

$$y_i = \psi(X_i) = A_g(b_1, b_2, b_3, ...b_B) \tag{14}$$

Consider a series system of ensemble classifiers having 5 base classifiers, each with a loss value of, say, 0.35. The combined loss of the classifiers would be given as:

$$\text{Loss} = (0.35)^5 = .00525 < < < < 0.35$$

Hence, ensemble techniques considerably reduce the error of the system. Moreover, further categorization of ensemble methods is described in Table 1.

4 Proposed framework

The ABIDS-VEM algorithm 1 presents a comprehensive approach to intrusion detection, incorporating several sophisticated phases as shown in Fig. 1. The process begins with a thorough data preprocessing phase, where unnecessary columns are dropped, missing values are imputed, and categorical variables are encoded. This phase also includes an innovative step of using isolation forest for outlier detection, which is particularly effective for high-dimensional datasets typical in network intrusion scenarios. Following this, the algorithm employs the equilibrium optimizer (EO) for feature selection, a critical step in reducing dimensionality and focusing on the most relevant attributes. The EO, inspired by control volume mass balance models, uses multiple agents to identify an optimal subset of features that best represent the entire dataset. After feature selection, the algorithm introduces a unique data ramification phase. This phase is crucial for addressing the common problem of class imbalance in intrusion detection datasets. The core of the classification process lies in the voting-based ensemble model (VEM). Here, multiple base classifiers are trained on different subsets of the ramified data, leveraging the strengths of diverse models. The final prediction is determined through a majority voting mechanism, where an instance is classified as an intrusion if more than two models in the ensemble predict it as such. This

Table 1 Ensemble methods

Category	Order	Heterogeneity
Bagging	Parallel	Homogeneous
RF	Parallel	Homogeneous
Boosting	Series	Homogeneous
AdaBoost	Series	Homogeneous
GDBoost	Series	Homogeneous
XGBoost	Series	Homogeneous
Stacking	Parallel	Heterogeneous
Hybrid	Any	Any

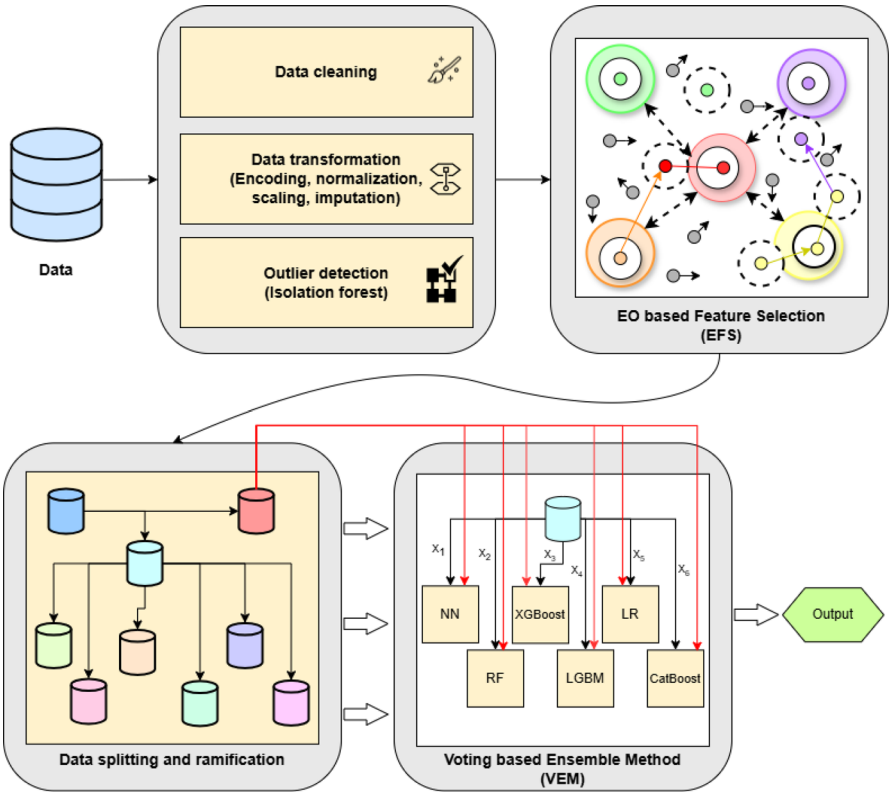

Fig. 1 Proposed ABIDS-VEM architecture

ensemble approach helps to reduce individual model biases and improve overall prediction accuracy.

4.1 Data preprocessing phase

This phase opts for data cleaning, data transformation, and outlier detection. Data cleaning is the process of identifying and correcting errors or inconsistencies in a dataset to ensure its accuracy and reliability. It involves tasks such as handling missing values, removing duplicates, and correcting typos. Data cleaning plays a critical role in data quality and is a significant step in the data preprocessing pipeline for various applications. After cleaning the data, data transformation is employed to label and scale the data (indicated by steps 4–5 in Algorithm 1).

The data transformation process often includes tasks such as normalization (scaling data to a common range), encoding categorical variables into numerical formats, creating new derived features, and aggregating or summarizing data. Data transformation aims to improve the quality of the dataset, make it more compatible with the chosen analytical techniques, and enhance the interpretability of the results. Lastly, this step involves outlier detection using Isolation Forest (IF). The IF

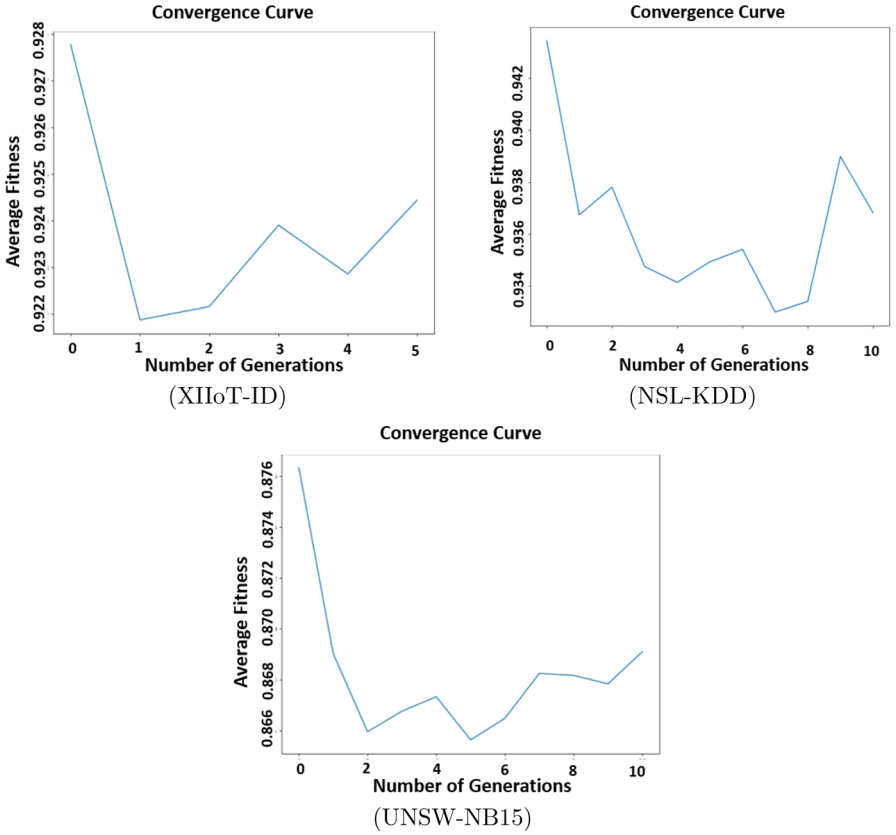

Fig. 2 Convergence curve of EO on different datasets

algorithm was initially designed for outlier detection. It works by randomly partitioning data samples based on certain attributes or features. The core idea is that rare observations, which are likely outliers, can be isolated more quickly with random splits, as they tend to end up alone in one branch. This concept is extended to split hyperplanes and guided splits in advanced models. This updated version includes heuristics for managing missing data and categorical variables.

Input: Dataset df
Output: Final classification results y_pred_final

Data preprocessing phase
Drop the unnecessary columns from df
Apply imputation technique
  df = df.replace(['?','-',nan,....],np.mean)
Create train and label data
  X = df.iloc[:,:-1] /* all columns except the last one */
  y = df.iloc[:,-1] /*only the last column (the training output)*/
Apply encoding schema to train data and labels
  X = $convert\_categorical\_values(X)$
  y = $LabelEncoder().fit\_transform(y)$
  df = pd.concat([X,y],axis = 1)
Use Isolation Forest for the outlier detection
  if = IsolationForest(n_estimators=100,
  random_state=42, bootstrap=false, contamination=auto,
    max_samples=auto)
  if.fit(X)
  scores = if.decision_fxn(X)
  outliers_indices = np.where(if.predict(X)==-1)[0]
  data_clean = df.drop(outlier_indices,axis = 1)
Feature selection using EO
fs = EO(num_agents=30, max_iter=10,train_data=
  data_clean.iloc[:,:-1],train_label = data_clean.iloc[:,
  -1],save_conv_graph=true)
results = fs.run()
li = [results.solution.best_agent == 1]
Data ramification phase
X = data_clean.iloc[:,li]
y = data_clean.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y,
  test_size=0.3, stratify = y)
Ramify(X_train,y_train)
VEM
Initialize the ensemble models
  $model_\eta = VEM(hyper\_parameters)_\eta$
Train the models
  $model_\eta.fit(X\_train_\eta, y\_train_\eta)$
  $y\_pred_\eta = model_\eta.predict(X\_test)$
  $\eta \leftarrow \eta + 1$
  until $\eta \leq N$
y_pred_final = []
  for i in X_test:
    if(count(y_pred[i]==1) > 2):
      y_pred_final.append(1)
    else:
      y_pred_final.append(0)
return y_pred_final

Algorithm 1 Proposed framework

It can also approximate pairwise distances and densities by examining the depth at which two observations separate and fitting trees beyond the balanced-tree height limit. Additionally, it provides options for both randomized and deterministic splits, making it a versatile tool for various data analysis tasks. IF is created by computing a score based on a collection of trees:

$$\text{if\_score}(X, \text{itr}) = 2^{-\frac{A(f(X))}{\overline{\text{itr}}}} \tag{15}$$

where *itr* is iterations, A(f(X)) is the average number of successful iterations for X, and $\overline{itr}$ is the average iterations for unsuccessful iterations.

4.2 EO-based feature selection phase

After passing through the data preprocessing phase, data is passed through the EO-based feature selection [57]. EO can reduce overfitting, improve accuracy, and decrease training time making it a promising choice for various ML tasks. The feature selection based on EO employs a continuous search space within a binary search algorithm. Particles representing solutions similar to the feature subset are placed within this search space, and their positions are determined by their concentration. The particle population, initial concentration, and fixed parameters are initialized. Concentration is represented as a binary vector, where 0 describes the deselection of a feature, and 1 indicates the selection of a feature. Feature selection is treated as a multi-objective optimization problem due to two distinct criteria for evaluating a considered feature subset: classification accuracy and the number of selected features. To address both objectives, a fitness score is utilized, which is assessed as follows:

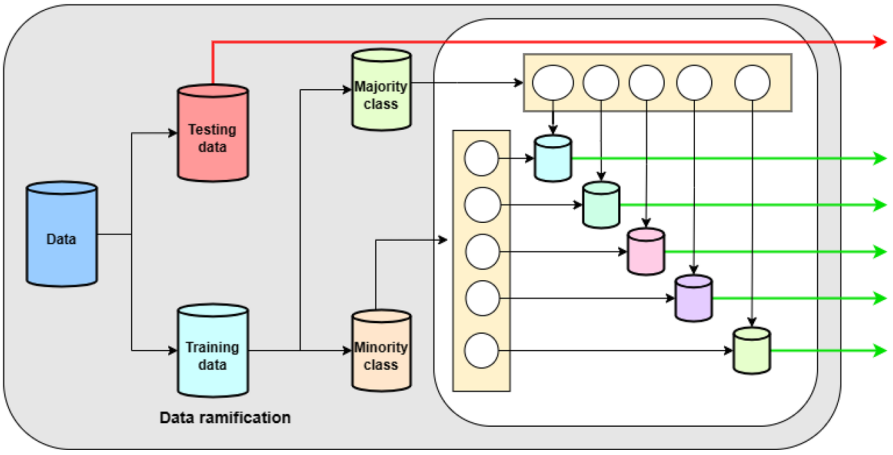

Fig. 3 Data ramification phase (combining the minority and majority dataset to produce more balanced data subsets)

$$\text{Fitness}(x) = \frac{\text{accuracy}(x)}{1 + \delta(\text{num\_particle}(x))} \tag{16}$$

The score is computed for the initial population, and the particles in global equilibrium are selected. The concentration is then updated to shift all particles toward the global optimum using Eq. (13). The updating rule combines the current equilibrium state (first term), exploration (second term), and exploitation (third term). Equations (12) and (11) exclusively manage exploitation through the CP, while exploration is introduced through mutation. Figure 2 describes the convergence of EO-based feature selection on the chosen datasets. These convergence curves help to monitor the performance of feature selection.

4.3 Data ramification phase

After feature selection, the data undergoes the data ramification phase, as outlined in Fig. 3. Initially, the dataset is bifurcated into train and test sets, reserving test data to evaluate. The train data is further split based on majority and minority labels, with each label divided into 5 subsets. The process of stratified sampling is employed to ensure uniformity among each subset. These subsets are then combined to create 5 individual training sub-datasets, which are used to train the ensemble model's 5 base classifiers. This phase significantly reduces the dataset's load on individual classifiers, leading to shorter training times and lower memory usage for all base classifiers. It also addresses the class imbalance issue by potentially combining multiple minority subsets with a majority subset, achieving a balanced sub-dataset without the need for sampling techniques or specialized algorithms for imbalanced datasets.

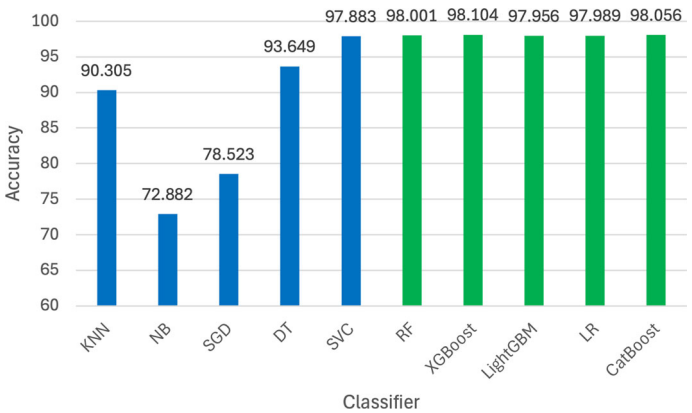

Fig. 4 Accuracy comparison of various ML classifiers for ensemble model selection

4.4 VEM

Following the data splitting and ramification phase, the subsequent phase involves the utilization of the voting-based ensemble model (VEM), which incorporates 5 base classifiers (denoted as model_η), namely, RF, LR, LGBM, XGB, and CB. The selection process of these classifiers was based on the accuracy score achieved on the X-IIoTID dataset. Initially, various well-known ML-based classifiers were employed on the selected dataset to analyze the performance. Considering the accuracy of all employed classifiers, an ensemble approach was devised, taking the top 5 classifiers with the highest accuracy as ensemble base classifiers as shown in figure 4. These classifiers are then individually trained in parallel on their respective training datasets derived from the preceding phase (X_train_η). After the training phase is completed, these model_η are combined to create an ensemble model. For the final classification, the predictions from each model_η are collected, and the ultimate output is determined using the voting scheme outlined in steps 17–18 of Algorithm 1. The combined power of these classifiers jointly reduces error and enhances precision and FAR values.

4.4.1 Random forest

RF is an ensemble machine learning algorithm that leverages a large ensemble of decision trees [58]. In the architecture of random forests, decision trees serve as the foundational predictors. While implementing RF, multiple decision trees are generated, each of which selects multiple samples from the original dataset and begins splitting based on minimum leaf node mean square error (MSE) in the case of regression tasks. This process continues until no more features are available, and the predictions from the multiple decision trees are combined to produce the final result, often by averaging the results (Eq. (17)) in the case of regression tasks.

$$\text{RF} = \frac{1}{T}\sum_{t=1}^{T} dt_t(X) \tag{17}$$

where RF denotes the combined RF model, $dt_t(X)$ is a single decision tree model and T is the total number of trees employed.

4.4.2 XGBoost

It is a scalable end-to-end tree-boosting technique that sequentially trains numerous weak learners [33]. Each subsequent learner works to rectify the errors made by the previous one, ultimately leading to the creation of an effective model primarily utilized for classification tasks. XGB not only addresses overfitting concerns associated with the GBDT algorithm but also improves real-world performance through its utilization of sparsity-aware metrics and multi-threading capabilities. Since the process of boosting is iterative, the goal of present iteration itr in terms of prediction of previous iteration $\hat{y}_a^{(\text{itr}-1)}$ could be expressed using the most recent tree dt_t:

$$\text{obj\_}fxn^{\text{itr}} = \sum_{a} \zeta(y_a, \hat{y}_a^{(itr-1)} + \sum_{b} \xi(dt_t)) \tag{18}$$

XGB uses a weight value, w, to minimize the objective function. The best leaf weight w in a present tree structure is given as:

$$w = -\frac{\sum_{a \in A} d1}{\sum_a \int_A d2 + r} = -\frac{P_i}{Q_i + r} \tag{19}$$

where d1 and d2 are the first and second derivatives of the loss function, for instance, a, respectively. At the end, with the best w, an objective function for the best tree structure is defined as:

$$\text{obj\_}fxn^{itr} = -\frac{1}{2}\sum_i \frac{P_i^2}{Q_i + r} + \gamma.T \tag{20}$$

4.4.3 LightGBM

LGBM stands out as an efficient implementation of gradient-boosting trees. It harnesses histogram and leaf-wise algorithms for enhancing both computational speed and prediction accuracy. The histogram technique is employed to amalgamate features that are incompatible with each other. The core idea is to discretize continuous features into 'n' integers, creating an 'n'-width histogram. Training data is then scanned based on these discretized histogram values for constructing a decision tree. This histogram-based method reduces time complexity. Additionally, LGBM identifies the leaf with the largest splitting gain and divides it using a leaf-wise strategy [31]. Although leaf-wise can lead to overfitting and deeper decision trees, LGBM mitigates this by imposing a maximum depth constraint, ensuring high efficiency and preventing overfitting.

In contrast to the traditional approach of using information gain to split nodes in a decision tree algorithm, LGBM employs a gradient-based one-side sampling (GOSS) method to compute variance gain for identifying the optimal split point. This involves sorting the absolute gradient values of training examples in descending order and selecting top U × 100% of data samples with the highest gradient values, denoted as U. Then from the remaining samples U^s, a subset V of size r * $|U^s|$ is chosen randomly. Finally, the instances are partitioned using the estimated variance σ on set U ∪ V. The gain of the split feature f of a node at point p is defined as:

$$V_f(p) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} \alpha_i + \frac{1-c}{d}\sum_{x_i \in B_l} \alpha_i\right)^2}{n_l^j(p)} + \frac{\left(\sum_{x_i \in A_r} \alpha_i + \frac{1-c}{d}\sum_{x_i \in B_r} \alpha_i\right)^2}{n_r^j(p)}\right) \tag{21}$$

where $A_l = \{x_i \in A : x_{ij} \leq p\}, A_r = \{x_i \in A : x_{ij} > c\}, B_l = \{x_i \in B : x_{ij} \leq p\}, B_r = \{x_i \in B : x_{ij} > p\}$, α denotes negative gradient of loss function, $\frac{1-c}{d}$ is utilized to normalize the sum of gradients.

4.4.4 Logistic regression

LR [59] is a widely used statistical model for binary classification. It is a type of regression analysis where the dependent variable is binary (0 or 1), representing two possible outcomes. LR models the relationship between the binary dependent variable and one or more independent variables, often using the logistic function, also known as the sigmoid function represented as $P_\theta(x)$, Eq. (22), to transform the linear combination of the predictor variables into a probability value between 0 and 1. This probability can be interpreted as the likelihood of the binary outcome occurring.

$$P_\theta(x) = \frac{1}{1 + e^{-x}} \tag{22}$$

The cost function will be,

$$J(\theta, y) = \frac{1}{2I} \sum_{i=1}^{I} (h_\theta(x)^i - y^i)^2 \tag{23}$$

where $h_\theta(x)$ is the output obtained.

Due to the non-convex function, the cost function of LR becomes,

$$J(C_\theta(x)^i, y) = -\log(h_\theta(x)) \qquad if\ y = 1 \tag{24}$$

$$J(C_\theta(x)^i, y) = -\log(1 - h_\theta(x)) \qquad if\ y = 0 \tag{25}$$

therefore, another representation of the cost function J(h_θ,y) is given as:

$$J(C_\theta(x), y) = -\frac{1}{2I} \sum_{i=1}^{I} (y^{(i)} \log(h_\theta(x)^{(i)}) + (1 - y)^{(i)} \log(1 - C_\theta(x)^{(i)})) \tag{26}$$

4.4.5 CatBoost

CB is constructed on the foundation of symmetric decision trees [32]. This algorithm is acknowledged as a classification method that not only produces outstanding results but also demonstrates a tenfold enhancement in prediction speed compared to other techniques that do not utilize symmetric decision trees. Distinguishing itself from previous gradient-boosted-decision trees (GBDT) algorithms, CB effectively handles gradient bias and prediction shift, leading to improved prediction accuracy and generalization capabilities, especially with large datasets. CB comprises two vital algorithms: ordered boosting, which calculates leaf values during tree structure selection to prevent overfitting and a unique technique for handling categorical features during the training process. Instead of using one-hot encoding for categorical variables, CB utilizes the concept of "ordered target

Table 2 Model parameters

Dataset	Features selected
XGBoost	eta=0.3, gamma=0, max_depth=6, subsample=1, min_child_weight=1, max_bin=256
RF	n_estimators=300, criterion=gini, min_samples_split=2, min_samples_leaf=1, max_features=sqrt
LightGBM	learning_rate=0.01, num_iterations=1000, num_leaves=30, max_depth=15, lambda_l1=0.5, lambda_l2=0.5, feature_fraction=0.75, max_bin=1000
LR	penalty=l2, dual=False, tol=0.0001, C=1, intercept_scaling=1, max_iter=100
CatBoost	iterations=1000, depth=6, learning_rate=0.3, l2_leaf_reg=5, border_count=50, boosting_type=Plain

statistics," a value derived from the ground truth output values that correspond to specific values of an absolute attribute in the dataset.

5 Results evaluation

In this section, we empirically validate the efficacy of the proposed framework for anomaly-based intrusion detection systems. To showcase the versatility and applicability of the proposed model, we conducted experiments on three datasets. The outcomes are juxtaposed with several baseline models including neural network (NN), k-nearest neighbor (KNN), Naive Bayes (NB), SCD, decision tree (DT), support vector machine (SVC), and Perceptron. This section presents, the experimental setup and performance metrics selected for analysis of results are elucidated, followed by an overview of the dataset used and the results analysis.

5.1 Experimental setup

The ABIDS-VEM[1] system was developed in Python utilizing TensorFlow 2.13, with the deep learning model crafted using Keras 2.13 and was tuned using random hit and trial experiments. The implementation and evaluation of ABIDS-VEM were carried out in Python 3.10.11 on an Asus Vivobook gaming laptop equipped with an NVIDIA 1650 GPU, an i5 processor, and 8 GB of RAM, running a 64-bit operating system. A detailed description of the model parameters is provided in Table. 2.

5.2 Performance metrics

In the evaluation of intrusion detection methods, various commonly employed metrics are employed to gauge their effectiveness. These metrics offer insights into how well the method can detect and categorize intrusions. Some of the essential metrics used in this work for assessing intrusion detection methods include accuracy, recall, precision, F1-score, and MCC. The efficacy of ABIDS-VEM is

[1] https://l1nk.dev/ABIDS-VEM

measured using metrics such as accuracy and F1-score while in cases with imbalanced data, it can be measured through the precision, recall, and MCC values. Accuracy assesses the overall correctness of the intrusion detection method in distinguishing between attack and normal samples. Precision reflects the ratio of correctly classified attack requests to the total samples classified as attacks. Recall (or Sensitivity or True Positive Rate) indicates the number of correctly classified attacks relative to the actual attack samples. The F1-score calculates the harmonic mean of precision and recall. MCC serves as a measure of the quality of binary and multiclass classifications, accounting for true and false positives and negatives, and providing a balanced assessment even for imbalanced class sizes. The MCC is essentially a correlation coefficient ranging from -1 to +1. For better depiction, a multiplier of 100 is used on the MCC score, making its range from -100 to 100.

$$\text{Accuracy} = \frac{s+t}{s+t+u+v} \tag{27}$$

$$\text{Precision} = \frac{s}{s+u} \tag{28}$$

$$\text{Recall (TPR)} = \frac{s}{s+v} \tag{29}$$

$$F1\_\text{score} = 2 * \frac{s}{2*s+v+u} \tag{30}$$

$$\text{MCC} = \frac{((s*t)-(u*v))}{\sqrt{(s+u)*(s+m)*(t+u)*(t+v)}} \tag{31}$$

Where s is true positive, t is true negative, u is false positive and v is false negative.

5.3 Dataset description

To obtain comprehensive results, the proposed framework is evaluated on three distinct datasets: XIIoTID, NSL-KDD, and UNSW-NB15. The descriptions of these datasets can be found in Tables 3, 4, and 5, respectively. These datasets are readily accessible and have been widely utilized for benchmarking purposes within the field of IDS. XIIoTID is specifically designed for IoT intrusion detection, reflecting the challenges of securing interconnected industrial devices. NSL-KDD, though older, remains a benchmark dataset for assessing network intrusion detection systems, enabling comparisons with established methods and providing insights into broader network security. UNSW-NB15 captures contemporary network behaviors, incorporating realistic attack patterns and normal traffic, making it particularly suitable for evaluating advanced Industry 4.0 systems. Together, these datasets comprehensively address the diverse security challenges faced in modern industrial

Table 3 XIIoTID dataset description

Category	Training instances	Testing instances
Reconnaissance	89286	38304
Weaponization	46954	20306
Exploitation	785	348
Lateral movement	21960	9636
Command & control	1973	890
Exfiltration	15551	6583
Tampering	3598	1524
Crypto Ransomware	327	131
RDoS	99194	42067
Normal	294955	126462
Total	574583	305501

Table 4 NSL-KDD dataset description

Category	Training instances	Testing instances
Normal	67343	9711
DOS	11656	7458
Probe	45927	2421
U2R	52	200
R2L	995	2754
Total	125973	22544

Table 5 UNSW-NB15 dataset description

Category	Label	Training instances	Testing instances
Analysis	0	2000	677
Backdoor	1	1746	583
DoS	2	12264	4089
Exploits	3	33393	11132
Fuzzer	4	18184	6062
Generic	5	40000	18871
Normal	6	56000	37000
Reconnaissance	7	10491	3496
Shellcode	8	1133	378
Worms	9	130	44
Total	10	175341	82332

environments, enhancing the empirical validation's applicability to practical scenarios.

For the UNSW-NB15 dataset, specific columns such as 'dur,' 'proto,' 'service,' and 'state' are initially removed. Subsequently, all the datasets undergo preprocessing and feature selection using the EO method, with details of the selected features provided in Table 6.

Table 6 EO selected features

Dataset	Features selected
XIIoTID	resp_bytes, missed_bytes, orig_pkts, resp_ip_bytes, total_bytes, paket_rate, byte_rate, orig_packts_ratio, resp_pkts_ratio, orig_bytes_ratio, resp_bytes_ratio, Avg_nice_time, STD_system_time, Avg_iowait_time, std_iowait_time, std_rtps, avg_kbmemused, Avg_num_Proc.s, std_num_proc.s, Avg_num_cswch.s, std_num_cswch.s, Conn_state, is_syn_only, Is_SYN_ACK, is_SYN_with_RST, OSSEC_alert, OSSEC_alert_level, Login_attmp, Succ_login, file_act
NSL-KDD	protocol_type, flag, dst_bytes, wrong_fragment, urgent, num_compromised, root_shell, num_root, num_file_creations, is_host_login, count, srv_serror_rate, srv_rerror_rate, diff_srv_rate, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate
UNSW-NB15	spkts, dpkts, sbytes, rate, dttl, sload, sinpkt, dinpkt, sjit, djit, swin, stcpb, dtcpb, dwin, ackdat, smean, trans_depth, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_srv_dst, is_sm_ips_ports

Table 7 Performance evaluation on XIIoTID dataset

Method	Recall	F1-score	Accuracy
NN	90.6796	89.8441	85.4992
KNN	93.1238	92.5003	90.3056
NB	66.1871	75.8097	72.8825
SGD Classifier	90.1103	84.3437	78.5234
DT	94.9705	95.0495	93.6489
SVC	95.9677	97.8036	97.8832
Perceptron	63.2281	74.2072	71.7823
Proposed	**96.4549**	**98.0141**	**98.1476**

Table 8 Time comparison of the proposed approach on XIIoTID dataset

Method	Training time
NN	00249.1990
KNN	00235.2274
NB	00000.4830
SGD Classifier	00028.4167
DT	00003.6127
SVC	13448.1150
Perceptron	00000.5521
Proposed	00059.1193

5.4 Performance comparison using XIIoTID dataset

Table 7 presents a comparative evaluation of different ML methods based on various performance metrics. When considering recall ABIDS-VEM outperformed the others with a value of 96.4549%. Whereas other methods range from 63.2281%

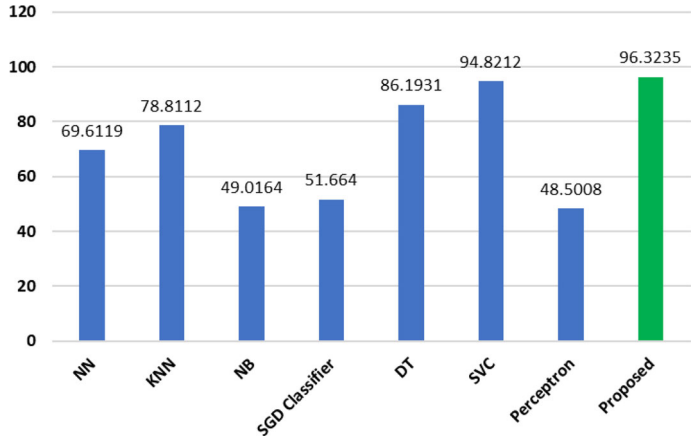

Fig. 5 Comparison of MCC on XIIoTID dataset

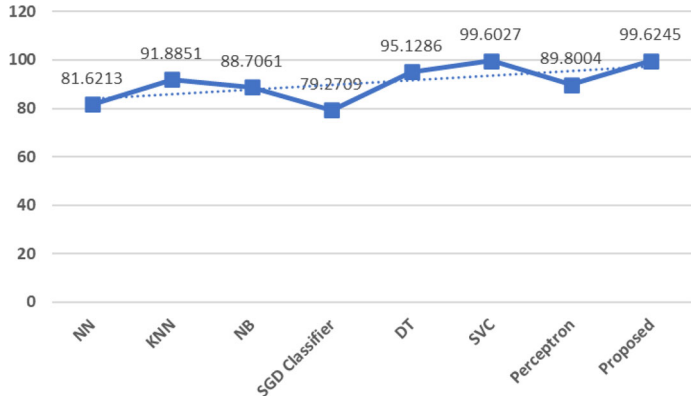

Fig. 6 Precision comparison on XIIoTID dataset

Table 9 Performance evaluation on NSL-KDD dataset

Method	Recall	F1-score	Accuracy
NN	91.6796	89.8441	85.4992
KNN	85.9722	91.9582	92.4636
NB	02.3336	05.5036	50.3371
SGD Classifier	42.9064	37.9031	29.5378
DT	86.5298	92.4015	92.8673
SVC	00.0973	00.1945	49.9290
Perceptron	43.7118	37.4365	26.7743
Proposed	**98.6939**	**98.8499**	**98.9671**

to 95.9677% which is less in comparison to the proposed approach. In terms of the F1-score the proposed framework excelled with a score of 98.0141%. The SVC method was closely followed with a score of 97.8036%. Accuracy, indicating the overall correctness of the classification, was highest for the proposed framework at 98.1476%, closely followed by SVC with 97.8832%. The proposed method demonstrates a training time of 59.1193 s as described in Table 8, offering a perspective on its computational efficiency compared to other techniques considering both accuracy and training time. These results suggest that the proposed framework demonstrates competitive performance across these metrics and could be a promising choice for the given task.

Figure 5 provides a comprehensive comparison of various ML methods based on the MCC metric, which is commonly used to assess the quality of binary and multiclass classifications. Among the methods, the proposed framework achieves the highest MCC score of 96.3235%, highlighting its excellence in binary classification tasks. Whereas other methods ranges from 48.5008% to 94.8212%, which shows that the proposed method out performed over them.

Figure 6 offers a comparative analysis based on their precision metric indicating that the proposed method stands out with the highest precision score of 99.6245%. These results establish the significance of the precision of the proposed approach in positive classifications, making it a robust choice for tasks that require accurate positive predictions.

5.5 Performance comparison using NSL-KDD dataset

Table 9 presents a comparative analysis of techniques on the NSL-KDD dataset. ABIDS-VEM demonstrates exceptional performance across all metrics. It method excels in recall with a high score of 98.6939 %. The proposed framework also achieves the highest F1-score at 98.8499, underlining its balanced precision and recall for attack and normal requests. Moreover, the proposed framework attains a remarkable accuracy of 98.9671 %, indicating a high degree of correctness in classifying attack and normal requests. Among other methods, the KNN also delivers a commendable performance, showcasing relatively short training time, high recall, excellent F1-score, and impressive accuracy.

Table 10 Time comparison of proposed approach on NSL-KDD dataset

Method	Training time
NN	0249.1990
KNN	0010.8241
NB	0000.5259
SGD Classifier	0000.8898
DT	0001.5004
SVC	4097.928
Perceptron	0000.7055
Proposed	0013.1976

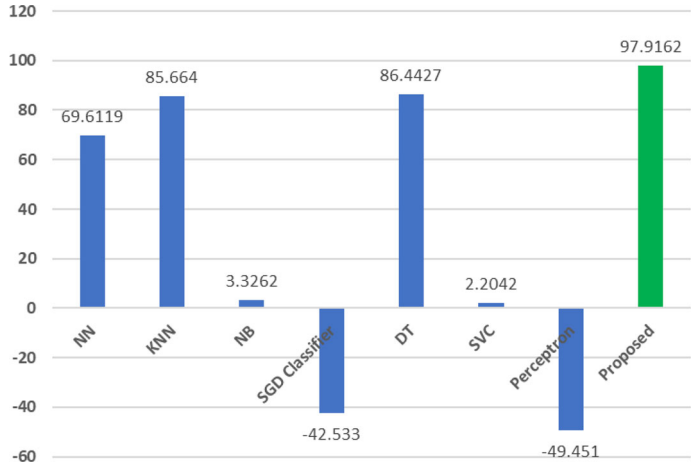

Fig. 7 MCC comparison on NSL-KDD dataset

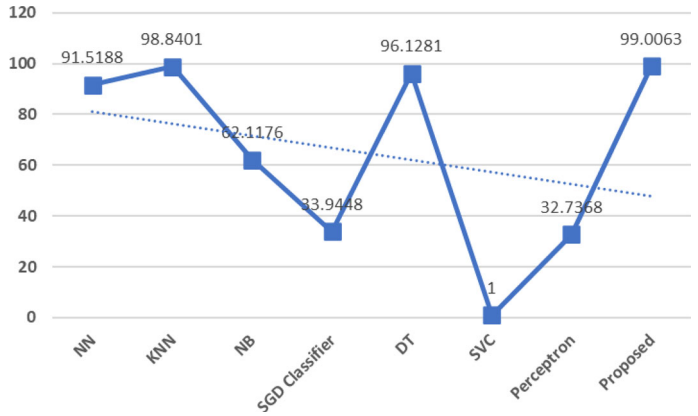

Fig. 8 Precision comparison on NSL-KDD dataset

On the other hand, methods like NN, DT, and NB exhibit varying degrees of performance in different metrics. Meanwhile, the SGD Classifier, SVC, and Perceptron methods demonstrate relatively lower performance across these metrics.

The proposed framework has a moderate training time of 13.1976s as indicated in Table 10, showcasing a reasonable training duration. These results highlight the robust performance of the proposed framework across various evaluation criteria, making it a promising choice for applications demanding strong performance in intrusion detection.

Figure 7 provides an MCC comparison, where higher scores indicate better performance. ABIDS-VEM stands out with a remarkable MCC score of 97.9162 %, showcasing its outstanding quality in classification tasks. It exhibits a high level of agreement between predicted and actual classifications. The DT method also achieves a notable MCC score of 86.4427 %, showing its effectiveness in

Table 11 Performance evaluation using UNSW-NB15 dataset

Method	Recall	F1-score	Accuracy
NN	90.9093	89.8441	85.4992
KNN	93.1238	92.5003	90.3056
NB	66.1871	75.8096	72.8825
SGD Classifier	90.1103	84.3438	78.5233
DT	94.9705	95.0495	93.6489
SVC	92.1943	91.2986	88.012
Perceptron	63.228	74.2072	71.7823
Proposed	**96.6264**	**96.2352**	**94.1327**

Table 12 Time comparison for UNSW-NB15 dataset

Method	Training time
NN	0249.1999
KNN	0235.2274
NB	0048.3004
SGD Classifier	0028.4167
DT	0003.6127
SVC	6335.2206
Perceptron	0000.5520
Proposed	0023.0519

classification tasks. On the contrary, the KNN method demonstrates a respectable MCC score of 85.664, implying its strong performance in classification. However, several methods, such as NN, NB, and SVC, exhibit relatively lower MCC scores, suggesting less effective performance in classification. Notably, the SGD Classifier and perceptron methods display negative MCC scores, signifying a significant discord between predicted and actual classifications. In contrast, the

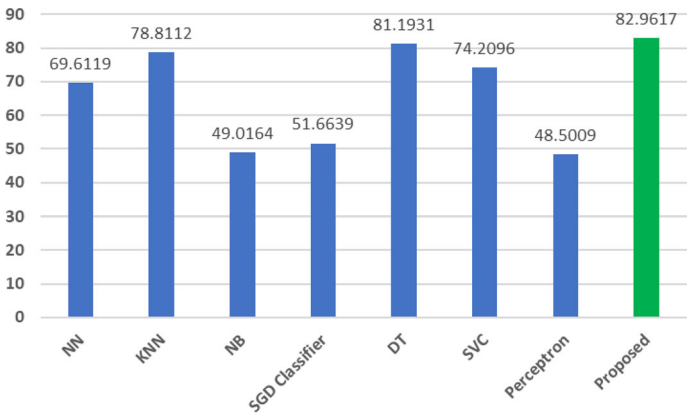

Fig. 9 MCC comparison for UNSW-NB15 dataset

proposed framework excels in terms of MCC, making it a promising choice for applications requiring precise and reliable classifications.

Figure 8 showcases a comparison of precision scores across various methods, with the proposed framework achieving an outstanding precision score of 99.0063 %. Close behind, DT method records a precision score of 96.1281 %, while the KNN method impresses with a score of 98.8401 %, and NN posts a commendable score of 91.5188 %. On the other hand, methods like SVC and NB register relatively lower precision scores. Particularly, the SGD and perceptron methods exhibit the lowest precision scores, highlighting their propensity for a higher frequency of false positives in their classifications. Contrarily, the proposed framework distinguishes itself by its superior precision, rendering it an exemplary option for scenarios demanding highly accurate identification of positive samples.

5.6 Performance comparison using UNSW-NB15 dataset

Table 11 presents an evaluation of ML methods on the UNSW-NB15 dataset, with Table 12 describing the training time. It indicates that the proposed framework has a training time of 23.0519 s, a recall of 96.6264 %, an F1-score of 96.2352 %, and an accuracy of 94.1327 %. Whereas other methods achieved training time ranging in between 0.5520 to 6335.22 s and recall ranges from 63.228 % to 94.9705 %. Similarly, for other metrics such as F1-Score and accuracy, results lie between 74.2072 % and 95.0495 %, respectively, for other methods.

Figure 9 illustrates that the proposed method outshines all others with an MCC score of 82.9617 %, surpassing the performance of alternative techniques, which achieve MCC scores ranging from 48.500 % to 81.193 %. Consequently, in terms of efficacy in managing the UNSW-NB15 dataset for intrusion detection tasks, the proposed approach stands as the superior option.

Figure 10 displays precision values and demonstrates that the proposed framework achieves the highest precision of 95.8471 %, indicating a strong ability

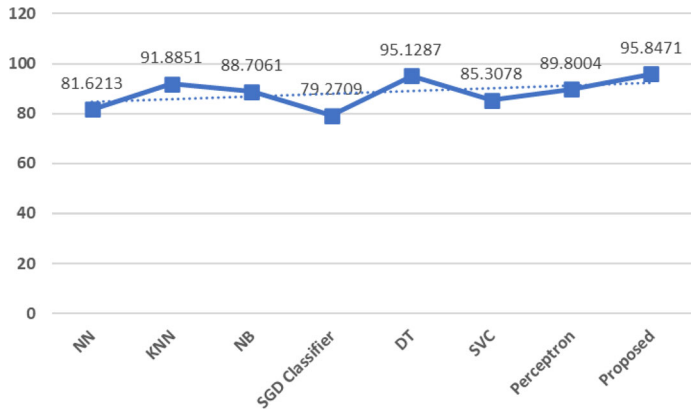

Fig. 10 Precision comparison using UNSW-NB15 dataset

to make accurate positive predictions. DT follows closely with a precision value of 95.1287 %, showcasing robust classification performance. KNN achieves a precision value of 91.8851 %, signifying its accuracy in positive predictions. Other methods, including NN, SVC, and NB, exhibit precision values of 81.6213, 85.3078, and 88.7061, respectively, showing their ability to make accurate positive predictions. SGD Classifier and perceptron show precision values of 79.2709 and 89.8004, respectively, providing insights into their performance in handling the UNSW-NB15 dataset for intrusion detection applications.

Figure 10 reveals that the DT method demonstrates its effectiveness with a precision of 95.1287 %, indicating a high level of accurate classifications. The KNN method, with a precision of 91.8851 %, also proves to be reliable in predicting positive outcomes accurately. Other methodologies like NN, SVC, and NB register precision scores of 81.6213 %, 85.3078 %, and 88.7061 %, respectively, underscoring their proficiency in making correct positive predictions. The SGD Classifier and perceptron, with precision scores of 79.2709 % and 89.8004 %, respectively, offer insights into their relative effectiveness when applied to the UNSW-NB15 dataset in intrusion detection scenarios, illustrating a spectrum of accuracy in identifying true positives across different techniques. However, the proposed framework outperforms all and leads at 95.8471 %, indicating its superior capability in accurately identifying true positive instances.

5.7 State-of-art comparison

Table 13 offers a comparative analysis between the ABIDS-VEM and various state-of-the-art models. The proposed model exhibits significant improvements

Table 13 State-of-art comparison with proposed approach

Dataset	Model	Accuracy	MCC	Recall	F1-score
XIIoT-ID	[54]	93.5	–	87.9	93.3
	[60]	99.8	–	99.7	99.6
	[61]	91.07	–	–	–
	Proposed	98.1	96.3	96.4	98.0
NSL-KDD	[52]	93.2	–	93.2	93.1
	[53]	86.5	–	79.2	87.0
	[54]	95.9	–	95.0	95.9
	[60]	99.5	–	99.4	99.4
	[62]	91.5	–	–	-
	Proposed	99.0	98.0	99.0	99.0
UNSWNB-15	[54]	82.2	–	74.3	81.8
	[50]	77.4	–	71.3	75.5
	[62]	99.2	–	–	–
	[63]	95.3	–	98.9	97.4
	Proposed	94.1	83.0	96.6	96.2

Table 14 Performance evaluation for various feature selection techniques on XIIoTID dataset

Method	Recall	F1-score	Accuracy
EO + VEM (Proposed)	**96.4549**	98.0141	**98.1476**
MI + VEM	96.6156	97.5642	97.6566
Pearson coefficient + VEM	96.5895	97.9736	98.0621
RFE + VEM	95.1918	96.4179	96.5536
RF + VEM	96.1638	97.2687	97.3753
ANOVA + VEM	96.3836	**98.0268**	98.0982

across all performance metrics, indicating a substantial enhancement in intrusion detection capabilities. However, the studies cited in [60, 62, 63] employ DL models, which perform better than the proposed approach. Nonetheless, the emphasis of this paper lies in presenting a proposed method characterized by low computational cost, suitable for deployment in resource-constrained environments. The training and deployment of ML-based ensemble models is faster and more cost-effective compared to DL, which frequently necessitates the use of GPUs. ML ensembles offer greater ease of implementation in low-resource settings compared to intricate deep-learning models. These consistent improvements across diverse datasets suggest that the proposed approach is not merely incrementally better but potentially represents a transformative advancement in intrusion detection capabilities.

5.8 Ablation study

Table 14 presents the performance evaluation of various feature selection techniques on the XIIoTID dataset. The proposed method, EO + VEM, outperforms other techniques in terms of recall (96.4549) and accuracy (98.1476), while achieving a competitive F1-score of 98.0141. ANOVA + VEM attains the highest F1-score (98.0268) but slightly lower recall (96.3836) and accuracy (98.0982) compared to the proposed approach. Other techniques, such as MI + VEM, Pearson

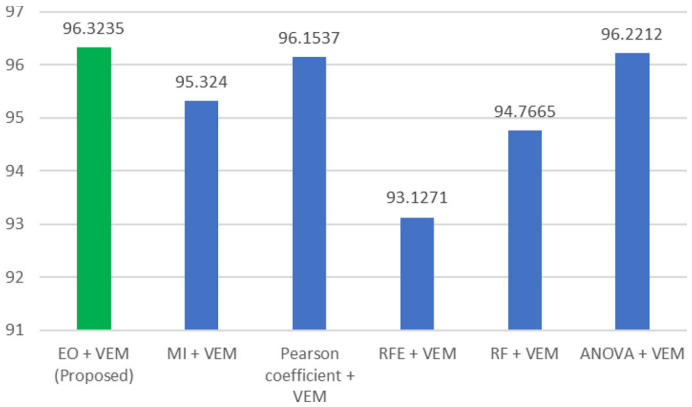

Fig. 11 MCC comparison for various feature selection techniques on XIIoTID dataset

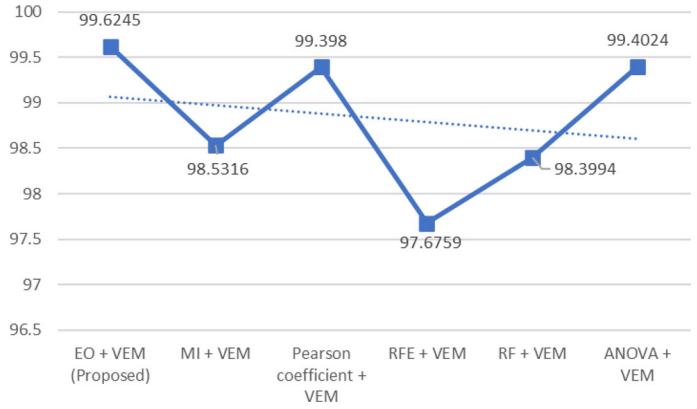

Fig. 12 Precision comparison for various feature selection techniques on XIIoTID dataset

coefficient + VEM, and RF + VEM, exhibit strong performance, though they fall short of EO + VEM in overall accuracy. RFE + VEM shows the lowest recall (95.1918) and accuracy (96.5536), indicating it may be less effective.

Figure 11 presents a comparison of MCC indicating that the proposed EO + VEM method achieves the highest MCC score (96.3235), indicating superior model performance. ANOVA + VEM (96.2212) and Pearson coefficient + VEM (96.1537) also perform well but fall slightly short of the proposed approach. MI + VEM (95.324) and RF + VEM (94.7665) demonstrate moderate MCC values, while RFE + VEM records the lowest MCC (93.1271), suggesting weaker predictive capability. These results highlight the effectiveness of EO + VEM in optimizing feature selection for improved model performance.

Figure 12 presents a comparison of precision scores. ANOVA + VEM (99.4024) and Pearson coefficient + VEM (99.398) demonstrate strong precision values, though slightly lower than the proposed approach (99.6245). MI + VEM (98.5316) and RF + VEM (98.3994) follow with moderate performance, while RFE + VEM records the lowest precision (97.6759). These results highlight the effectiveness of EO + VEM in achieving superior precision compared to alternative feature selection methods.

5.9 Computational complexity analysis

The computational complexity of an EO is described by a function that correlates the algorithm's execution time with the input problem's size. In this context, the widely used Big-O notation is employed. The complexity is influenced by several factors, including the no. of particles, the no. of dimensions, the no. of iterations "itr", and the cost of function evaluation.

$$O(EO) = O(pd) + O(\text{init}) + O(itr(fe)) + O(\text{itr}(ms)) + O(\text{itr}(cu)) \tag{32}$$

where, pd=problem definition, init=initialization, fe=function evaluations, ms=memory saving, cu=concentration update, itr=iterations, um=update mechanism.

Neglecting less influential terms such as $O(pd)$, and combining the terms $O(\text{itr}(ms)) + O(\text{itr}(cu))$ to $O(um)$, Eq. (32) could be given as,

$$O(EO) = O(\text{init}) + O(\text{itr}(fe)) + O(um) \tag{33}$$

Consider P as the number of particles, D as dimensionality, Eq. (33) could be given as,

$$O(EO) = O(P * D) + O(P * fe) + O(P * D) \text{ per iteration} \tag{34}$$

$$O(EO) = O(2 * P * D) + O(P * fe) \text{ per iteration} \tag{35}$$

$$O(EO) = O(P * (D + fe)) \text{ per iteration} \tag{36}$$

For I iteration Eq. (36) could be given as,

$$O(EO) = O(\text{itr} * P * (D + fe)) \text{ per iteration} \tag{37}$$

The overall computational complexity of the proposed framework could be analyzed as:

$$O(\text{Proposed\_framework}) = O(EO) + O(\text{model}_\eta) + \tag{38}$$

$O(Voting)$

This can be simplified to,

$$O(\text{Proposed\_framework}) = O(\text{itr} * P * (D + fe)) + O(\text{model}_\eta) + O(N * V) \tag{39}$$

where N is the number of classifiers and V is the time for each classifier.

Therefore, the overall computational complexity is of the polynomial order.

6 Conclusion and future work

The proposed Anomaly-Based Intrusion Detection System using the Voting-based Ensemble Model (ABIDS-VEM) exhibits exceptional efficacy across diverse datasets, positioning it as a formidable solution for fortifying network infrastructures against emerging cyber threats. Rigorous evaluation on the XIIoTID, NSL-KDD, and UNSW-NB15 benchmarks demonstrates its superiority over traditional machine learning paradigms, including neural networks, k-nearest neighbors, Naive Bayes, stochastic gradient descent, decision trees, support vector classifiers, and perceptrons. The framework's remarkable performance is evidenced by its high recall rates, peaking at 98.6939%, which underscores its proficiency in minimizing false negatives and effectively identifying intrusions. Moreover, the model achieves F1-scores exceeding 98.8499%, indicative of an optimal balance between precision and recall. With accuracy metrics surpassing 98.9671%, ABIDS-VEM showcases unparalleled prowess in discriminating between benign and malicious network activities. These compelling results underscore the model's potential as a critical component in next-generation cybersecurity architectures, capable of adapting to

and mitigating evolving threat landscapes. Future research directions include enhancing the framework's resilience against adversarial AI-based attacks and exploring its applicability in real-time, high-throughput network environments. In parallel, we recognize that our initial experiments were conducted on an Asus Vivobook gaming laptop equipped with an NVIDIA 1650 GPU and 8GB RAM-a setup that, while sufficient for research, prompts questions about the model's scalability in large-scale industrial networks. Moving forward, we plan to investigate the memory and processing time requirements in production environments, incorporate additional metrics for a more comprehensive evaluation, and compare our approach with traditional balancing techniques.

Author contributions PV: Concept, Design, Implementation, Writing-original draft. DOS: Design, Review. TN: Concept, Review. NM: Concept, Implementation. NB: Concept, Design, Review. JB: Concept, Design, Review.

Funding This work is supported by SFI under Grant Number SFI NCF Project (Cyber Shock) with Grant No. 22/NCF/DR/11165 and SFI 12/RC/2289-P2 (Insight) and. To facilitate Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version resulting from this submission.

Data availability Not Applicable

Declarations

Conflict of interest There is no Conflict of interest among the authors of the manuscript.

References

1. Denning DE (1987) An intrusion-detection model. IEEE Trans Softw Eng 2:222–232
2. Sommer R, Paxson V, (2010) Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy. IEEE 2010:305–316
3. Khan IA, Keshk M, Pi D, Khan N, Hussain Y, Soliman H (2022) Enhancing IIoT networks protection: a robust security model for attack detection in internet industrial control systems. Ad Hoc Netw 134:102930
4. Khan IA, Moustafa N, Pi D, Sallam KM, Zomaya AY, Li B (2021) A new explainable deep learning framework for cyber threat discovery in industrial IoT networks. IEEE Internet Things J 9(13):11604–11613
5. Khan IA, Pi D, Abbas MZ, Zia U, Hussain Y, Soliman H (2022) Federated-SRUs: a federated-simple-recurrent-units-based IDS for accurate detection of cyber attacks against IoT-augmented industrial control systems. IEEE Internet of Things J 10(10):8467–8476
6. Wang Z, Fok KW, Thing VL (2022) Machine learning for encrypted malicious traffic detection: approaches, datasets and comparative study. Computers Secur 113:102542

7. Zhang Y, Liu Q (2022) On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. Future Gener Computer Syst 133:213–227
8. Yang Y, Zheng K, Wu B, Yang Y, Wang X (2020) Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. IEEE Access 8:42169–42184
9. Zhong Y, Chen W, Wang Z, Chen Y, Wang K, Li Y et al (2020) HELAD: a novel network anomaly detection model based on heterogeneous ensemble learning. Computer Netw 169:107049
10. Alazzam H, Sharieh A, Sabri KE (2020) A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. Expert Syst Appl 148:113249
11. Sharma B, Sharma L, Lal C (2019) Anomaly detection techniques using deep learning in IoT: a survey. In: 2019 international conference on computational intelligence and knowledge economy (ICCIKE). IEEE 2019:146–149
12. Mishra P, Varadharajan V, Tupakula U, Pilli ES (2018) A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun Surv Tutor 21(1):686–728
13. Zhou Y, Cheng G, Jiang S, Dai M (2020) Building an efficient intrusion detection system based on feature selection and ensemble classifier. Computer Netw 174:107247
14. Nancy P, Muthurajkumar S, Ganapathy S, Santhosh Kumar S, Selvi M, Arputharaj K (2020) Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks. IET Commun 14(5):888–895
15. Sai Satyanarayana Reddy S, Chatterjee P, Mamatha C (2019) Intrusion detection in wireless network using fuzzy logic implemented with genetic algorithm. In: Computing and Network Sustainability: Proceedings of IRSCNS 2018. Springer; pp 425–432
16. Kan X, Fan Y, Fang Z, Cao L, Xiong NN, Yang D et al (2021) A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network. Inform Sci 568:147–162
17. Saber S, Salem S (2023) High-performance technique for estimating the unknown parameters of photovoltaic cells and modules based on improved spider wasp optimizer. Sustain Mach Intell J 5:1–2
18. Salem S (2023) An improved binary quadratic interpolation optimization for 0–1 knapsack problems. Sustain Mach Intell J 4:1–1
19. Amma NB, Selvakumar S, Velusamy RL (2020) A statistical approach for detection of denial of service attacks in computer networks. IEEE Trans Netw Serv Manag 17(4):2511–2522
20. Nasir MH, Khan SA, Khan MM, Fatima M (2022) Swarm intelligence inspired intrusion detection systems-a systematic literature review. Computer Netw 205:108708
21. Verma P, Tapaswi S, Godfrey WW (2021) A request aware module using CS-IDR to reduce VM level collateral damages caused by DDoS attack in cloud environment. Clust Comput 24(3):1917–1933
22. Aljanabi M, Ismail MA, Mezhuyev V (2020) Improved TLBO-JAYA algorithm for subset feature selection and parameter optimisation in intrusion detection system. Complexity 2020:1–18
23. Sagi O, Rokach L (2018) Ensemble learning: a survey. Wiley Interdiscip Rev: Data Min Knowl Discov 8(4):e1249
24. Bharot N, Verma P, Sharma S, Suraparaju V (2018) Distributed denial-of-service attack detection and mitigation using feature selection and intensive care request processing unit. Arabian J Sci Eng 43:959–967
25. Tama BA, Lim S (2021) Ensemble learning for intrusion detection systems: a systematic mapping study and cross-benchmark evaluation. Computer Sci Rev 39:100357
26. Mohammed A, Kora R (2023) A comprehensive review on ensemble deep learning: opportunities and challenges. J King Saud Univ-Computer Inform Sci 35(2):757–74
27. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl-Based Syst 191:105190
28. Yang J, Sheng Y, Wang J (2020) A GBDT-paralleled quadratic ensemble learning for intrusion detection system. IEEE Access 8:175467–175482
29. Zou X, Hu Y, Tian Z, Shen K (2019) Logistic regression model optimization and case analysis. In: 2019 IEEE 7th international conference on computer science and network technology (ICCSNT). IEEE 2019:135–139
30. Biau G, Scornet E (2016) A random forest guided tour. Test 25:197–227
31. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. (2017) Lightgbm: a highly efficient gradient boosting decision tree. Adv Neural Inform Process Syst. 30

32. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A (2018) Boost: unbiased boosting with categorical features. Adv Neural Inform Process Syst. 31
33. Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd acm SIGKDD international conference on knowledge discovery and data mining, pp 785–794
34. Verma P, Breslin JG, O'Shea D (2022) FLDID: federated learning enabled deep intrusion detection in smart manufacturing industries. Sensors 22(22):8974
35. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In (2009) A detailed analysis of the KDD CUP 99 data set. IEEE Symposium on Computational Intelligence for Security and Defense Applications 2009:1–6
36. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military communications and information systems conference (MilCIS). IEEE 2015:1–6
37. Saba T, Rehman A, Sadad T, Kolivand H, Bahaj SA (2022) Anomaly-based intrusion detection system for IoT networks through deep learning model. Computers Electr Eng 99:107810
38. Panigrahi R, Borah S, Pramanik M, Bhoi AK, Barsocchi P, Nayak SR et al (2022) Intrusion detection in cyber-physical environment using hybrid Naïve Bayes-Decision table and multi-objective evolutionary feature selection. Computer Commun 188:133–144
39. Yazdinejad A, Dehghantanha A, Karimipour H, Srivastava G, Parizi RM. A robust privacy-preserving federated learning model against model poisoning attacks. IEEE Transactions on Information Forensics and Security. 2024
40. Namakshenas D, Yazdinejad A, Dehghantanha A, Srivastava G(2024) Federated quantum-based privacy-preserving threat detection model for consumer internet of things. IEEE Transactions on Consumer Electronics
41. Puri A, Sharma N (2017) A novel technique for intrusion detection system for network security using hybrid svm-cart. Int J Eng Develop Res 5(2):155–161
42. Guo C, Ping Y, Liu N, Luo SS (2016) A two-level hybrid approach for intrusion detection. Neurocomputing 214:391–400
43. Khan IA, Pi D, Khan N, Khan ZU, Hussain Y, Nawaz A, et al. (2021) A privacy-conserving framework based intrusion detection method for detecting and recognizing malicious behaviours in cyber-physical power networks. Appl Intell;pp 1–16
44. Mohammadi S, Mirvaziri H, Ghazizadeh-Ahsaee M, Karimipour H (2019) Cyber intrusion detection by combined feature selection algorithm. J Inform Secur Appl 44:80–88
45. Gite P, Chouhan K, Krishna KM, Nayak CK, Soni M, Shrivastava A (2023) ML based intrusion detection scheme for various types of attacks in a WSN using C45 and CART classifiers. Mater Today: Proceed 80:3769–3776
46. Yazdinejad A, Kazemi M, Parizi RM, Dehghantanha A, Karimipour H (2023) An ensemble deep learning model for cyber threat hunting in industrial internet of things. Digital Commun Netw 9(1):101–110
47. Louk MHL, Tama BA (2023) Dual-IDS: a bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. Expert Syst Appl 213:119030
48. Wang Z, Liu J, Sun L et al (2022) EFS-DNN: an ensemble feature selection-based deep learning approach to network intrusion detection system. Secur Commun Netw 2022:2693948
49. Rashid M, Kamruzzaman J, Imam T, Wibowo S, Gordon S (2022) A tree-based stacking ensemble technique with feature selection for network intrusion detection. Appl Intell 52(9):9768–9781
50. EL Asry C, Benchaji I, Douzi S, EL Ouahidi B (2024) A robust intrusion detection system based on a shallow learning model and feature extraction techniques. Plos One 19(1):01
51. Muhammad A, Murtza I, Saadia A, Kifayat K (2023) Cortex-inspired ensemble based network intrusion detection system. Neural Comp Appl;pp 1–14
52. Assy AT, Mostafa Y, El-khaleq AA, Mashaly M. Anomaly-based intrusion detection system using one-dimensional convolutional neural network. Procedia Computer Science. 2023;220:78–85. The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40). https://doi.org/10.1016/j.procs.2023.03.013
53. Magdy ME, Matter AM, Hussin S, Hassan D, Elsaid SA (2023) Anomaly-based intrusion detection system based on Feature selection and majority voting. Indones J Electr Eng Comput Sci 30(3):1699–1706
54. Uddin MA, Aryal S, Bouadjenek MR, Al-Hawawreh M, Talukder MA (2024) usfAD based effective unknown attack detection focused IDS framework. arXiv preprint arXiv:2403.11180.

55. Nazaroff WW, Alvarez-Cohen L (2000) Environmental Engineering Science. John Wiley & Sons
56. Guo Z (2002) Review of indoor emission source models Part 1. Overview. Environ Poll 120(3):533–549
57. Guha R, Chatterjee B, Khalid Hassan S, Ahmed S, Bhattacharyya T, Sarkar R. Py_FS: a python package for feature selection using meta-heuristic optimization algorithms. In: Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2021. Springer; 2022. pp 495–504
58. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
59. Minka T (2001) Algorithms for maximum-likelihood logistic regression. Stat Tech Rep;758
60. Wang Z, Xie B, Yang S, Li D, Wang J, Chan S (2025) A deep residual SConv1D-attention intrusion detection model for industrial Internet of Things. Clust Comput 28(2):116
61. Jayalaxmi P, Saha R, Kumar G, Alazab M, Conti M, Cheng X (2023) PIGNUS: a deep learning model for IDS in industrial internet-of-things. Computers Secur 132:103315
62. Choudhary S, Kesswani N (2020) Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT. Procedia Computer Sci 167:1561–1573
63. Gyamfi E, Jurcut AD (2022) Novel online network intrusion detection system for industrial IoT based on OI-SVDD and AS-ELM. IEEE Internet Things J 10(5):3827–3839

Authors and Affiliations

Priyanka Verma[1] · Donna O'Shea[2,5] · Thomas Newe[3] · Nakul Mehta[4] · Nitesh Bharot[4] · John G. Breslin[4]

✉ Priyanka Verma
priyanka.verma@universityofgalway.ie

Donna O'Shea
donna.c.oshea@ul.ie

Thomas Newe
thomas.newe@ul.ie

Nakul Mehta
n.mehta3@universityofgalway.ie

Nitesh Bharot
nitesh.bharot@universityofgalway.ie

John G. Breslin
john.breslin@universityofgalway.ie

[1] School of Computer Science, University of Galway, University road, Galway H91TK33, Ireland

[2] Department of Computer Science, Munster Technological University, Rossa Ave, Bishopstown, Cork T12 P928, Ireland

[3] Department of Electronics and Computer Engineering, University of Limerick, Castletroy, Limerick V94 T9PX, Ireland

[4] Data Science Institute, University of Galway, University road, Galway H91TK33, Ireland

[5] Digital Engineering, University of Limerick, Castletroy, Limerick V94 T9PX, Ireland