

Machine learning methods for detecting intrusions based on network traffic analysis^{*}

Yuliia Kostiuk^{1,†}, Pavlo Skladannyi^{1,2,*,†}, Volodymyr Sokolov^{1,†}, Svitlana Rzaieva^{1,†},
and Karyna Khorolska^{1,†}

¹ Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., 04053 Kyiv, Ukraine

² Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, 42 Ac. Glushkov ave., 03680 Kyiv, Ukraine

Abstract

The article discusses modern machine learning methods for detecting intrusions in computer networks based on network traffic analysis. An architecture for an intelligent intrusion detection system is proposed, combining an autoencoder, a one-class support vector machine, an Isolation Forest, and Extreme Gradient Boosting (XGBoost), using a deep representation of traffic in the feature vector space. The scientific novelty lies in the integration of One-Class Neural Network with an adaptive update mechanism based on Markov decision processes (MDP), which provides automatic retraining in case of changes in traffic characteristics. The study employs procedures to reduce the dimensionality of the feature space using Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP). (Uniform Manifold Approximation and Projection—UMAP). Explainable Artificial Intelligence (XAI) modules are proposed using SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) methods. The developed system has been tested on the CICIDS2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017) and UNSW-NB15 (University of New South Wales Network Behavior 2015) open datasets. The results demonstrate classification accuracy of up to 97%, high interpretability, and model adaptability in detecting zero-day attacks in real-time, making it suitable for critical information infrastructures.

Keywords

network traffic, intrusion detection, machine learning, anomalies, classification, Autoencoder, gradient boosting (XGBoost), one-class support vector machine (One-Class SVM), explainable artificial intelligence (XAI), CICIDS2017, UNSW-NB15

1. Introduction

Ensuring reliable protection of computer networks in today's environment of rapidly growing cyber threats is one of the key challenges for the information security sector. Every day, dozens of new types of attacks appear that turn off network infrastructure, steal confidential information, or disrupt service availability [1–3]. According to data from the European Union Agency for Network and Information Security (ENISA) and the US Cybersecurity and Infrastructure Security Agency (CISA), an unauthorized device connected to the Internet can be compromised in a matter of minutes by automated botnets, vulnerability scanners, and malicious scripts, as confirmed in the ENISA Threat Landscape 2024 and MITRE ATT&CK Evaluation Reports 2024, which particularly highlight the growth of attacks using multi-layered scenarios, AI-generated exploits, and zero-day vulnerabilities in network infrastructures [4–6]. Distributed denial-of-service (DDoS) attacks, zero-day exploits, and modern methods of covert intrusion significantly complicate the work of traditional protection systems.

^{*} CPITS-II 2025: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2025, Kyiv, Ukraine

^{*} Corresponding author.

[†] These authors contributed equally.

✉ y.kostiuk@kubg.edu.ua (Y. Kostiuk); p.skladannyi@kubg.edu.ua (P. Skladannyi); s.rzaieva@kubg.edu.ua (S. Rzaieva); v.sokolov@kubg.edu.ua (V. Sokolov); k.khorolska@kubg.edu.ua (K. Khorolska)

ORCID 0000-0001-5423-0985 (Y. Kostiuk); 0000-0002-7775-6039 (P. Skladannyi); 0000-0002-7589-2045 (S. Rzaieva); 0000-0002-9349-7946 (V. Sokolov); 0000-0003-3270-4494 (K. Khorolska)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The vast majority of traditional intrusion detection systems (IDSs), particularly signature-based ones, have several limitations related to the need for constant updates of attack databases, high false positive rates, and limited adaptability [7, 8]. With the increasing complexity of traffic and the rapid evolution of attack methods, there is a need to implement intelligent solutions that can learn, self-correct, and operate in real-time [6, 9]. This determines the relevance of developing new-generation intrusion detection systems based on machine learning (ML) algorithms [2, 10], which can perform in-depth analysis of network traffic, recognize patterns in node behavior, and detect anomalous actions before an attack can cause harm.

The problem is that in practice, building an effective ML-IDS requires solving several tasks: correctly representing traffic as a set of features [11], reducing its dimensionality without losing significance [12], selecting an optimal classifier capable of working with mixed or unbalanced data [5], and ensuring that the results are interpretable for cybersecurity specialists [4, 7, 13]. The complexity of the problem is compounded by the need to process large amounts of data in real time, which requires effective algorithmic solutions and optimized software implementation.

The scientific novelty of the study lies in the proposed architecture of an intelligent intrusion detection system, which combines an autoencoder for detecting hidden dependencies in traffic [1], a one-class support vector method (One-Class SVM) [8], an isolation forest for detecting deviations [2, 3, 10], and gradient boosting (Extreme Gradient Boosting XGBoost) for accurate classification of attacks [6]. In addition, dimensionality reduction methods (PCA, t-SNE, UMAP) were applied, which allows optimizing the model training process [7, 12]. An important feature is the integration of explainable artificial intelligence (Explainable Artificial Intelligence XAI) based on SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations), which provides analysts with an understanding of the model's solutions.

The practical significance of the proposed approach lies in the development of a software prototype of an intrusion detection system suitable for deployment in real computer networks. The system has been tested on the CICIDS2017 and UNSW-NB15 open datasets [5], achieving a classification accuracy of over 97% with a minimum false positive rate [14–16]. The research results can be applied to monitor critical information infrastructure, industrial enterprise networks, financial institutions, and government agencies, where not only detection accuracy but also transparency of decision-making is essential.

2. Literature review

Research focuses on the application of machine learning methods to analyze behavioral patterns in network traffic, driven by the growth of data volumes, the increasing complexity of attacks, and the limitations of signature-based systems. Hybrid and deep models that can detect anomalies without labels and provide explainability for security professionals are becoming relevant.

Feng et al. [1] proposed a Tensor Recurrent Neural Network with a built-in differential privacy mechanism. The proposed model achieved an accuracy of over 95% in classifying dynamic network flows, confirming the effectiveness of combining privacy and deep learning in security tasks.

Chua et al. [2] developed a multi-stage framework for network traffic analysis using entropy filtering and the Isolation Forest algorithm. This approach enabled the effective detection of anomalies in web traffic, particularly in cases of unpredictable activity without prior sample labeling.

Djidjev [3] presented the Set-Structured Isolation Forest (siForest) method, adapted to the specifics of network traffic. The siForest architecture was tested on complex, structured flows and demonstrated high efficiency in detecting non-standard behavior, indicating the feasibility of integrating structured learning into IDS.

The work of Ripan et al. [10] considers the application of isolation forests in the context of hybrid intelligent systems. The authors noted that the use of outlier detection methods enables the effective classification of anomalies in mixed traffic, particularly in IoT environments. However, the model's limitations regarding deep contextual dependencies indicate the advisability of combining it with Autoencoder or RNN.

Liu et al. [4] proposed a hybrid intrusion detection system that combines scalable k-means+ clustering, Random Forest, and deep learning. The system architecture enables the processing of large amounts of traffic, demonstrating high accuracy and scalability, which is particularly valuable in distributed networks.

Maseer et al. [5] benchmarked several machine learning models on the CICIDS2017 dataset. XGBoost and deep autoencoders showed the best results, providing a balance between accuracy, completeness, and speed. The authors also emphasized the importance of forming an optimal set of features and the system's adaptability to changes in traffic.

Mane and Rao [7] focused on the explainability of IDS models. They integrated the Explainable AI (XAI) framework, specifically SHAP, to build understandable interpretations of classification decisions. This increases trust in the models and facilitates the audit of security system results.

A review of the literature showed that models based on decision trees, k-NN, and heuristics provide high speed but insufficient accuracy in multi-class and zero-day detection tasks. In contrast, deep models (Autoencoder, LSTM, One-Class SVM) demonstrate better ability to detect anomalies in the absence of complete labeling, although they require significant computational resources and feature preparation (PCA, UMAP, t-SNE) [5, 11–13]. The integration of XAI methods (SHAP, LIME) improves interpretability [7], but challenges remain in real-time, false positives, and working with partially labeled data. Most existing solutions are not scalable and do not cover all classes of attacks. Thus, it is relevant to create an integrated IDS framework that incorporates traffic vectorization, feature reduction, ensemble models, and XAI explanations, which represents the scientific novelty of this study.

3. Research methods

The research methods employ a step-by-step approach to developing an intelligent intrusion detection system for computer networks, with a focus on network traffic analysis. At the first stage, data from the CICIDS2017 and UNSW-NB15 public datasets [5, 17] were collected and preprocessed, which included cleaning, normalization, and feature encoding. Traffic is represented as a vector space of features, taking into account the quantitative and behavioral characteristics of flows. To optimize training, dimensionality reduction methods were employed, including principal components analysis (PCA), stochastic neighbor embedding (t-SNE), and uniform manifold approximation and projection (UMAP) [11, 12].

Based on the prepared data, machine learning models were trained, specifically the Autoencoder, One-Class SVM [8], Isolation Forest [2, 3, 10], and gradient boosting (XGBoost) [4, 5]. Performance was evaluated using accuracy, completeness, F1-measure, and AUC-ROC metrics [5, 17]. To increase the transparency of decisions, explainable artificial intelligence (XAI) was implemented using SHAP and LIME methods [7, 13]. The final stage was the development of a prototype system with a REST API that can be integrated into environments such as Zeek or Splunk [14, 15, 18] for practical network traffic monitoring.

4. Main material

Modern information security threats are characterized by increasing complexity and intensity of attacks, which requires the implementation of intelligent intrusion detection systems (IDS) [8]. In highly loaded networks, traditional signature-based IDS are ineffective against new or modified attacks, which necessitates a transition to machine learning-based models [5, 19]. Such systems must operate in near real-time, ensuring high accuracy, low false positives (FP) and false negatives (FN), and adaptation to new threats [4, 17, 20]. Key metrics are accuracy (CR), performance (PPS), noise resistance, and generalization ability.

The effectiveness of IDS largely depends on constructing a relevant feature space, which is formed by vectorizing network traffic and applying dimension reduction methods (PCA, t-SNE, UMAP) [11–13] to mitigate the “curse of dimensionality” and reduce computational resource load. Network traffic is based on TCP, UDP, ICMP, and IP protocol packets, which often carry malicious activity [6]. Since most attacks are session-based, it is advisable to analyze complete TCP sessions or streams [2, 4, 5, 10]. To this end, a set of 45 features has been developed, covering general parameters (TTL, ToS, traffic volume), characteristics of ICMP, UDP, and TCP messages, and specifics of TCP sessions (duration, flags, window dynamics, OS fingerprinting).

The resulting feature vectors are fed into machine learning models—from Random Forest and XGBoost [4, 5] to Autoencoder [1], LSTM [11], and One-Class SVM [8]. Building such an architecture enables the implementation of a flexible, scalable, and explainable intrusion detection system that can effectively respond to the dynamics of modern cyber threats.

In Figure 1, the sequence diagram illustrates the process of a machine learning-based intrusion detection system. Network traffic arrives in the form of raw packets, which are analyzed and combined into sessions. Next, features are extracted, their dimensionality is reduced [11, 12], and then the classifier determines whether the traffic is normal or abnormal. If a threat is detected, the system notifies the administrator and records the incident in SIEM [14, 15, 18]. If the traffic is normal, the event is simply logged.

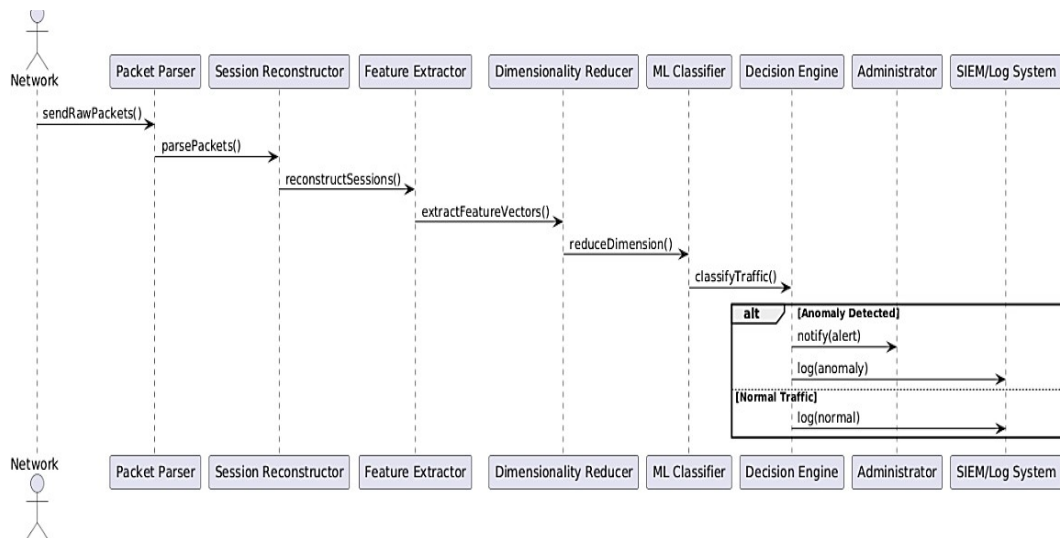


Figure 1: Sequence diagram for intrusion detection system

Figure 2 illustrates a data flow diagram that shows the key stages of converting raw network traffic into a feature vector, which is then fed into a machine learning model. It covers the processes of packet parsing, TCP/UDP session aggregation, feature extraction, dimensionality reduction, and decision-making, followed by notification to the security analyst.

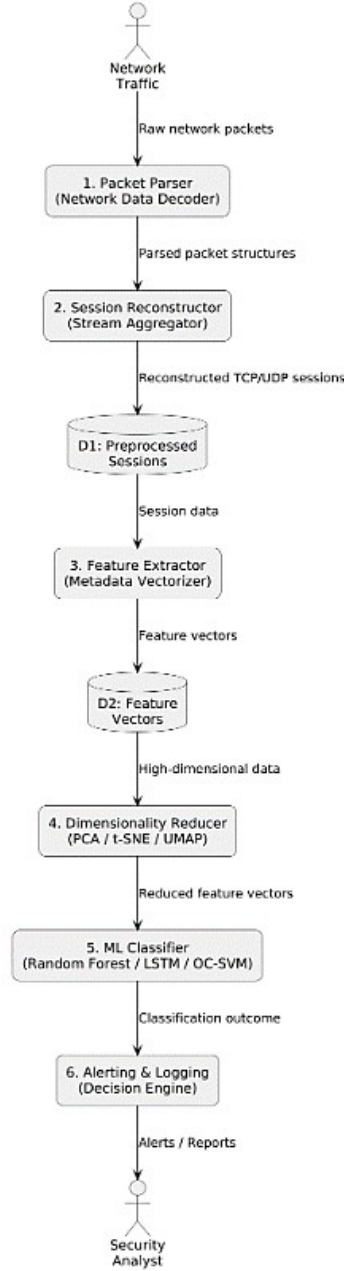


Figure 2: Building features for intrusion detection

One of the key problems in building an IDS based on machine learning is the inability to accurately determine the initiator of a TCP session if its beginning is lost, since client ports are selected randomly, while server ports are mostly fixed [5, 6]. Due to the lack of explicit information about the roles of the parties in the TCP header, heuristics are used, including fixed port ranges, internal server lists, and TCP fingerprinting, to determine the OS and port selection model.

In particular, the 1-P0SD algorithm is used to identify the host OS, which allows the type of network stack to be determined with a single packet with constant complexity $O(1)$ [15]. The Traffic-to-Embedding (TEA) architecture features a Session Data structure based on hash tables, host activity accounting, and the processing of UDP/IP/ICMP packets or TCP sessions with $O(1)$ complexity, as well as the formation of a 45-feature vector [2, 4, 5]. To reduce the computational load, methods are used to reduce the dimensionality of features without losing information, which avoids the “curse of dimensionality” [20].

Among the methods for reducing dimensionality for real-time IDS, the most effective is the principal component analysis (PCA) method, which calculates eigenvectors and the values of the feature covariance matrix [13]. PCA transforms the feature matrix $X_{N \times M}$, into the space of new

orthogonal variables $Y_{N \times A}$, where $A < M$, according to the formula $X = \sum_{a=1}^A v_a p_a^T + E$, where N is the number of sessions, M is the number of features, v_a is the components of the evaluation matrix, p_a^T is the load, E is the residual matrix.

Advantages of PCA: dimensionality reduction with minimal loss of information, improved model generalization, and the possibility of parallel processing (in particular on GPUs) [14, 15, 21]. The resulting vectors are fed into ML models (Random Forest, XGBoost, LSTM, Autoencoder, One-Class SVM) [4, 5, 8, 11, 22]. Careful feature construction and reduction have a critical impact on the accuracy of anomaly detection in loaded networks.

The DFD diagram (Figure 3) illustrates three levels of detail: the contextual level (DFD Level 0), the basic architecture of subsystems (DFD Level 1), and the detailed feature extractor (DFD Level 2). The data flow goes from the network sensor to the classifier, where features from network traffic are analyzed to detect intrusions. Additionally, internal protocol processing and session aggregation for feature vector formation are shown.

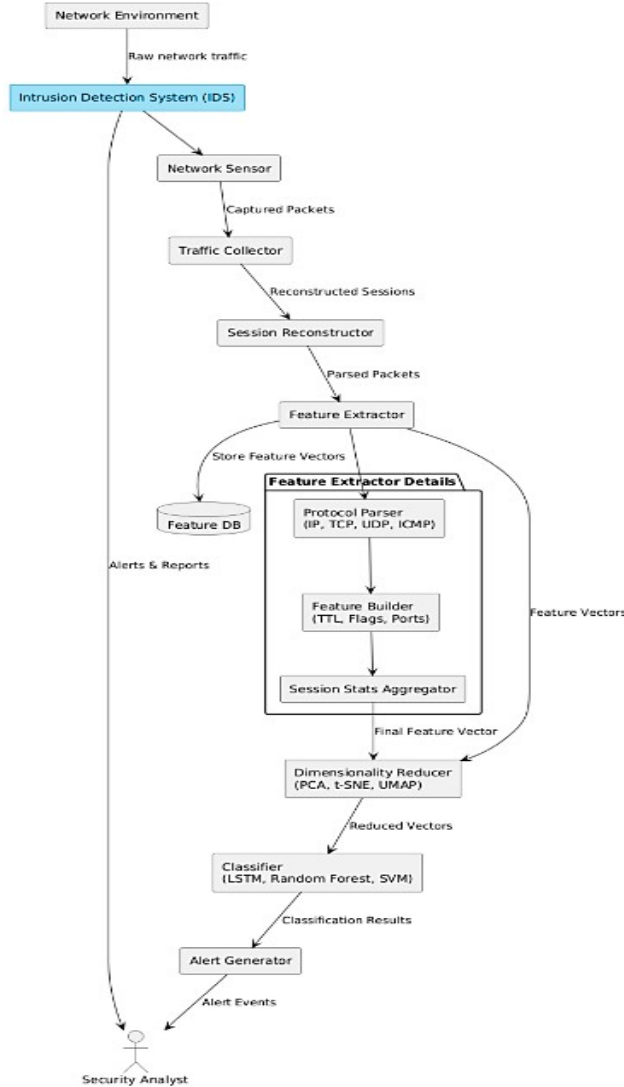


Figure 3: DFD levels of intrusion detection systems with machine learning

One of the leading areas of development for intelligent IDS is modeling the behavior of dynamic objects and developing heuristic threat detection mechanisms [8, 17, 21]. In this context, a computer network is viewed as a complex dynamic system that generates a stream of network

events. In normal mode, traffic is characterized by stable patterns that can be formalized for model training.

Anomalous events, such as attacks or failures, violate these patterns. With a representative sample of legitimate traffic, a profile of normal behavior is formed, against which current traffic is compared. Deviations from it indicate a possible anomaly [7, 8, 22]. Since accurate labeling of normal and abnormal traffic is difficult, one-class classification models or novelty detection algorithms are used.

The formalized task can be described as follows: given a set $X \subset A$ consisting of objects of a single (normal) class, it is necessary to construct a function $f: A \rightarrow \{0,1\}$ that returns 1 if the object belongs to the known class (i.e., is not an anomaly), and 0 if the object is potentially harmful or atypical. In the case of intrusion detection systems, this means that the algorithm must identify a new network packet as safe or anomalous based on knowledge obtained exclusively from normal traffic.

Modern anomaly detection methods include the use of Autoencoder to detect deviations based on reconstruction errors [1], One-Class SVM to construct boundaries of normal behavior [8], Deep SVDD to isolate centers of the normal class [6], Graph Neural Networks to take into account the connections between nodes [20, 23], and ensemble models to improve generalization.

The key challenge remains the formation of a relevant vector representation of network events, which requires effective feature extraction, noise reduction, and model adaptation to traffic changes [5, 12, 17]. PCA, t-SNE, and UMAP are used to combat the “curse of dimensionality” [11, 12]. An effective intrusion detection model must combine high-quality feature selection, methods for representing them, and adaptation to the dynamics of the network environment.

The diagram (Figure 4) details the process of anomaly detection using a single-class classification method. The system receives feature vectors, performs normalization, applies an OCC model (e.g., One-Class SVM or Autoencoder), calculates anomalous values, and transmits the result to the decision-making subsystem. This approach allows for effective detection of deviations in network traffic, even if the training sample contains only normal examples.

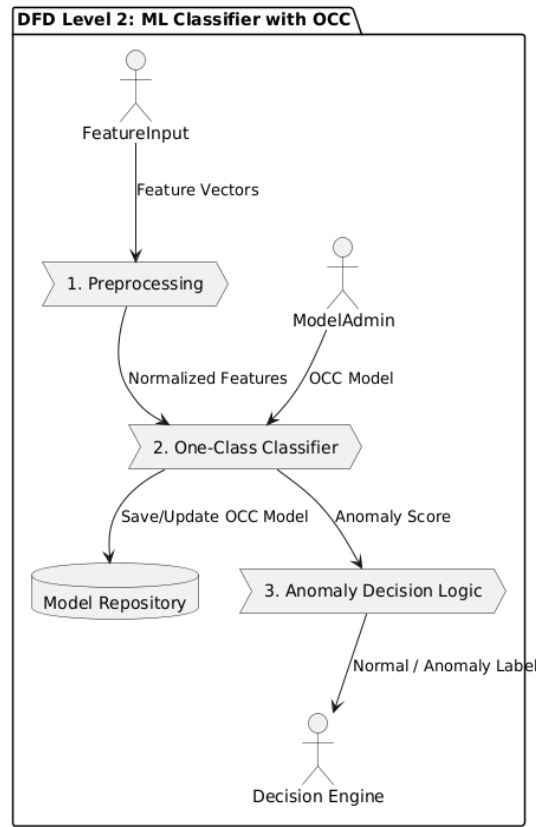


Figure 4: DFD Level 2: Classification subsystem with single-class training in IDS

The diagram (Figure 5) shows three levels of the DFD model. Level 0 illustrates the interaction of external users with the IDS system as a “black box.” Level 1 reveals the main modules of the system: traffic processing, session reconstruction, feature construction, classification, and decision-making. Level 2 details the internal logic of the machine learning subsystem, including the stages of preprocessing, inference, and result verification.

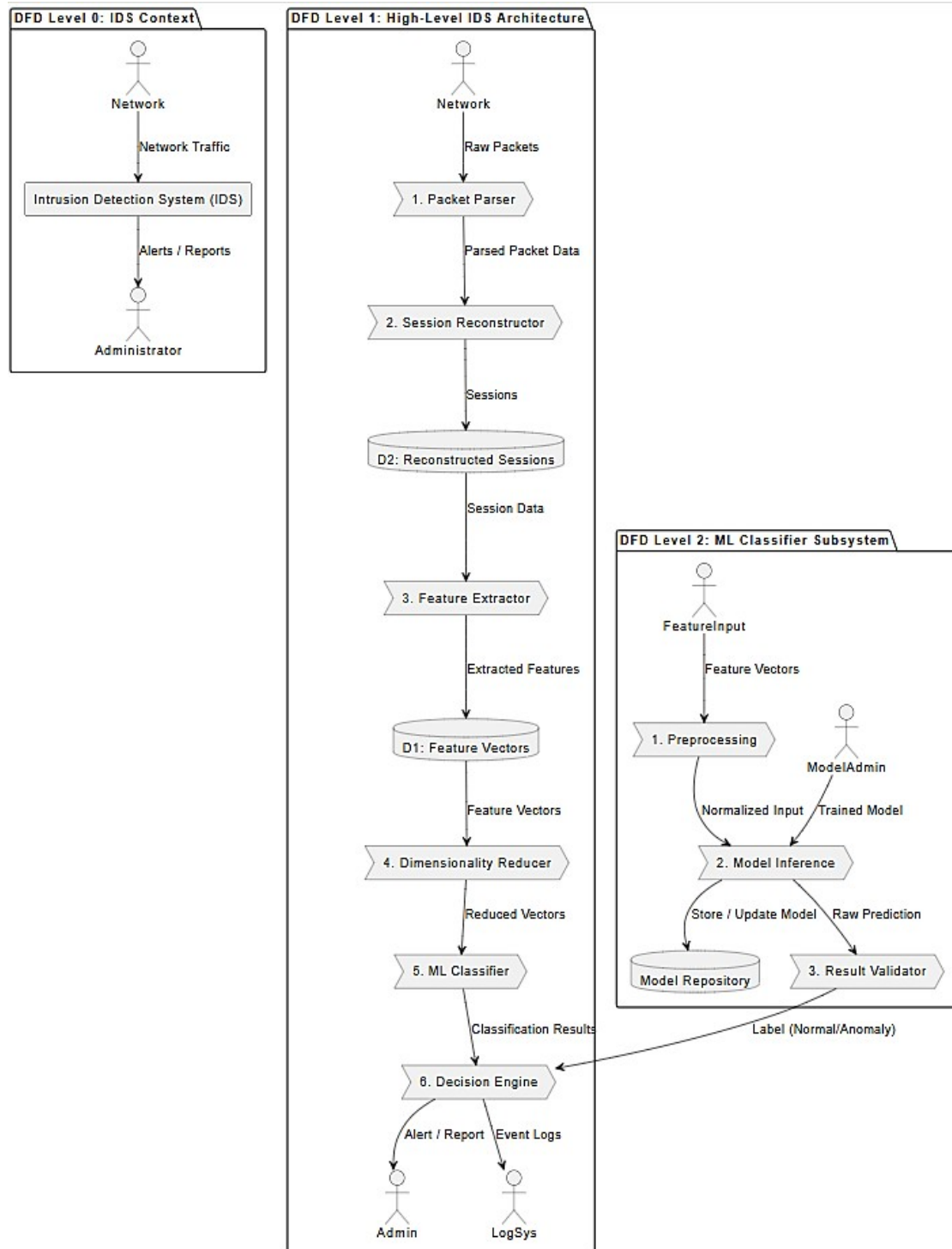


Figure 5: DFD model of a multi-level intrusion detection system based on machine learning

Research on the applicability of one-class classification (OCC) to network traffic anomaly detection tasks, in particular using one-nearest neighbor (1-NN) and one-class SVM methods, based on the KDD Cup'99 benchmark dataset, showed that even without deep parameter optimization, these methods demonstrate competitive results in terms of True Positive Rate (TPR) and False Positive Rate (FPR) metrics [8]. However, their sensitivity to noise in the training set, significant computational costs, and limited scalability complicate the application of such models in large datasets (>100,000 feature vectors).

One effective approach to implementing one-class classifiers is to use a multilayer perceptron (MLP) as an adaptive reconstructive filter [8, 17]. In this approach, a multilayer feedforward neural network is constructed in such a way as to learn to reproduce only normal (non-anomalous) network traffic patterns, identifying anomalies as significant deviations from the expected reproduction.

Let there be a set of positive feature vectors of dimension m , obtained after preprocessing the traffic. To evaluate the similarity between the input and output vectors, a metric D (is used (e.g., Euclidean or cosine distance). The network consists of m input and output neurons, h hidden layer neurons with a sigmoid activation function and linear output. The training objective is to minimize the deviation between the input x_i and the corresponding output y_i (i.e. $D[x_i, y_i] \approx 0$).

After training, a deviation threshold is determined (for example, $\theta = \text{mean } D[x_i, y_i] + \varepsilon$), according to which new vectors are classified: if the deviation does not exceed the threshold, the vector is considered normal, otherwise it is considered anomalous. Thus, the network approximates an identical mapping only for normal patterns and acts as an anomaly detector.

Three key parameters determine the effectiveness of this model:

- h the number of neurons in the hidden layer, which regulates the model's generalization ability. Too large h leads to overfitting, while too small h leads to a loss of ability to detect the full spectrum of normal traffic.
- η (learning rate) the rate of weight updates, which affects both the convergence rate and the stability of the model.
- β (momentum) an inertial coefficient that smooths out gradient fluctuations and improves the quality of training.

In intrusion detection systems, modeling normal traffic allows zero-day attacks to be detected without the need for a complete database of anomalies, which is particularly important for IoT, cyber-physical systems, cloud environments, and 5G infrastructure [6, 15, 24, 25]. The approach is enhanced by the following components: One-Class Neural Networks (OC-NN), which combine OCC and deep learning [1]; MiniBatch SGD and GPU acceleration for streaming learning [17]; containerization and deployment in Kubernetes for scalability [18]; XAI methods (SHAP, LIME, Grad-CAM) for interpreting decisions [7, 13, 26]. The integration of these technologies enables the construction of adaptive, scalable, and explainable anomaly detection systems in network traffic.

The diagram in Figure 6 shows the main stages of network traffic processing: from reception to classification using a multilayer perceptron trained only on normal traffic. The reconstruction error determines whether the input sample is abnormal. In case of an anomaly, an alert is generated and a record is made in SIEM.

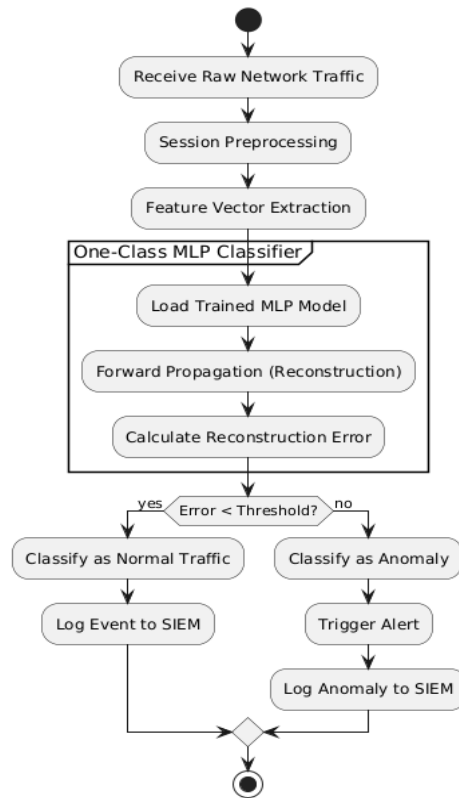


Figure 6: Activity diagram for an intrusion detection system based on One-Class MLP

The sequence diagram (Figure 7) shows the interaction of the main components of the network traffic anomaly detection system. The system receives input data from the network, performs preliminary processing and feature extraction, after which a classifier based on a multilayer perceptron (MLP) evaluates whether the traffic is abnormal. If deviations are detected, the anomaly processing module is activated, which can initiate adaptive retraining, automatic threshold updates, and a self-healing mechanism. All events are logged and sent to the administrator.

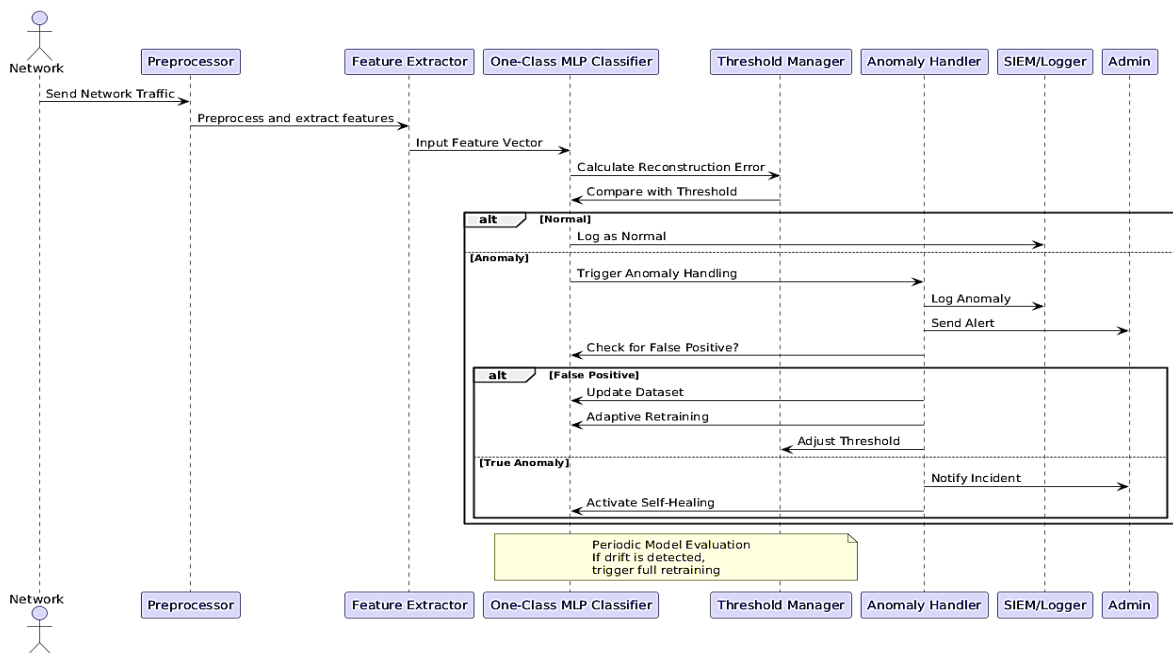


Figure 7: Diagram of the sequences of the intrusion detection system based on a single-class MLP model

The proposed machine learning model for detecting intrusions in network traffic is based on a one-class neural network (OC-NN) [1], which is trained exclusively on “reference” (normal) data and detects anomalies based on deviations from expected behavior [8]. The architecture includes two modes: training and testing, and also implements adaptive update mechanisms through Markov decision making [17, 24]. Data processing stages and mathematical models:

Stage 1. Formation of characteristics

Network traffic input data (packets or sessions) are presented as feature vectors $X_{\text{raw}} = \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$, $x^{(i)} \in R^{(M)}$, where N is the number of samples and M is the number of primary features. Normalization is performed to eliminate large-scale distortions:

$$x_{\text{norm}}^{(i)} = \frac{x^{(i)} - \mu}{\sigma} \quad (1)$$

where μ mean value, σ standard deviation for each feature in the dataset.

Stage 2. Dimension reduction (PCA)

To reduce the dimensionality of the feature space and improve generalization, the principal component analysis (PCA) method is used [4]:

$$X_{\text{norm}} = U \Sigma V^T, V_{\text{proj}} = X_{\text{norm}} \cdot V_{[1, \hat{M}]} \quad (2)$$

where $U \Sigma V^T$ is singular value decomposition (SVD) components, \hat{M} is a new dimension after cutting off less significant components.

Stage 3. Training the Autoencoder

Training a neural network (Autoencoder) involves minimizing the loss function [1, 4]:

$$L_{\theta} = \frac{1}{N} \sum_{i=1}^N \|V_{\text{proj}}^{(i)} - f_{\theta}(V_{\text{proj}}^{(i)})\|^2 \quad (3)$$

where f_{θ} is the neural network with parameters θ , which performs reconstruction of input vectors.

The parameters are updated using the stochastic gradient descent method with momentum:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(\theta_t) + \alpha \cdot (\theta_t - \theta_{t-1}) \quad (4)$$

where η is a learning rate, α is a moment coefficient, t is an iteration number.

Stage 4. Setting the rejection threshold

Based on reconstruction errors, the limit (threshold) value is determined:

$$\theta_{\text{res}} = \max_i \|V_{\text{proj}}^{(i)} - f_{\theta}(V_{\text{proj}}^{(i)})\|_2 \quad (5)$$

This value is used as the boundary between normal and abnormal samples during testing.

Stage 5. Classification of new traffic

For new incoming traffic x_{new} after normalization and transformation through PCA [1, 3, 8]:

$$x_{\text{new}} \rightarrow x_{\text{new, proj}} = \text{PCA}(x_{\text{new}}) \quad (6)$$

reconstruction and calculation of Euclidean distance are performed:

$$d = \|f_{\theta}(x_{\text{new, proj}}) - x_{\text{new, proj}}\|_2 \quad (7)$$

Then the classification rule is applied:

$$\text{label}(x_{\text{new}}) = \begin{cases} \text{Normal}, & d \leq \theta_{\text{res}} \\ \text{Anomaly}, & d > \theta_{\text{res}} \end{cases} \quad (8)$$

To improve sensitivity to deviations, a standardized anomaly score is calculated:

$$AnomalyScore(x_{\text{new}}) = \frac{d - \mu_d}{\sigma_d} \quad (9)$$

where μ_d mean reconstruction error, σ_d standard deviation. This allows you to standardize the distance and make the threshold value dynamic.

The probability that the sample is abnormal is calculated using the function:

$$P_{\text{Anomaly}}(x) = 1 - \exp\left(\frac{-d^2}{2\sigma_d^2}\right) \quad (10)$$

Stage 6. Suppression of non-informative features.

To improve feature quality before submission to PCA, entropy-based weight scaling is used [23]:

$$w_j = 1 - \frac{H_j}{\log K} \quad (11)$$

where H_j entropy j^{th} features, K the number of clusters or bins for estimating the distribution. After that, weights are applied to the input vectors:

$$x_{\text{weighted}}^{(i)} = x^{(i)} \cdot w \quad (12)$$

This allows you to reduce the impact of noise or non-informative parameters.

Stage 7. Metrics for measuring the effectiveness of the anomaly detection system

Classic binary classification metrics are used to quantitatively assess the quality of the proposed classification system [8, 13]. The calculations are based on an error matrix containing: TP (True Positive)—the number of correctly detected anomalies, FP (False Positive)—the number of false positives in normal data, FN (False Negative)—the number of missed anomalies, TN (True Negative)—the number of correctly classified normal samples.

Precision is the proportion of correctly classified anomalies among all detections:

$$Precision = \frac{TP}{TP + FP} \quad (13)$$

Recall (completeness)—the proportion of detected anomalies among all actual anomalies:

$$Recall = \frac{TP}{TP + FN} \quad (14)$$

F1-score is the harmonic mean between Precision and Recall, which allows for a balanced evaluation of the model:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (15)$$

True Positive Rate (TPR) or sensitivity:

$$TPR = \frac{TP}{TP + FN} \quad (16)$$

False Positive Rate (FPR)—the proportion of false positives among all normal samples:

$$FPR = \frac{FP}{FP + TN} \quad (17)$$

Overall accuracy is the proportion of correctly classified samples among all samples:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (18)$$

AUC-ROC (Area Under Curve—Receiver Operating Characteristic)—the area under the ROC curve, which is constructed based on the dependence of TPR on FPR:

$$ROC = TPR \text{ vs } FPR, AUC = \int_0^1 TPR(FPR) dFPR \quad (19)$$

This metric allows you to evaluate the overall ability of the model to distinguish between normal and abnormal samples when the threshold changes.

Matthews Correlation Coefficient (MCC) is a balanced classification index that takes into account all elements of the error matrix and is used to evaluate quality even with unbalanced samples:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (20)$$

Stage 8. Adaptive model update based on MDP

To ensure dynamic adaptation of the system to changes in the environment (e.g., changes in attack types, traffic distribution), a self-healing mechanism based on the Markov decision process is implemented (Markov Decision Process, MDP) [16, 17].

The model is defined by five parameters $\langle S, A, T, R, \gamma \rangle$, where S is the set of system states that describe the situation (e.g., error rate, TPR, FPR, environment change), A is the set of possible actions:

$$A = \{ DoNothing, UpdateThreshold, Relearn, Retrain \} \quad (21)$$

$T(s, a, s')$ transition function: the probability that after performing action a in state s the system will transition to state s' , $R(s, a)$ reward function that evaluates the benefit of the action (for example, $R = -FPR$, $\gamma \in [0, 1]$ discount factor that determines the weight of future rewards.

The goal is to maximize the expected cumulative reward:

$$V^\pi(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s \right] \quad (22)$$

where π is a policy that determines action $a = \pi(s)$ in state s .

Decision-making policy updates are performed using a Q learning algorithm:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R(s, a) + \gamma \max_{\hat{a}} Q(s, \hat{a}) - Q(s, a)] \quad (23)$$

where α is a learning coefficient, $Q(s, a)$ is the current estimate of the action value a in state s .

The system supports automatic updates in case of model degradation, adaptation of the θ_{res} threshold based on current statistics, and online learning with data flow preservation [1, 3, 8]. It is formalized, adaptive, and effective for detecting zero-day attacks, working in IoT and edge environments, cloud infrastructures (Docker, Kubernetes) with GPU and MiniBatch SGD support [6, 20, 21, 25, 26]. The use of PCA, autoencoders, and Markov decision processes (MDP) ensures scalability, flexibility, and resistance to real network threats [24]. The proposed model responds adaptively to changes, operates in real time, and ensures continuous protection in complex computing environments.

DFD Level 1 (Figure 8) illustrates the functional components and data flows of the network traffic anomaly detection system using a One-Class neural network. The system consists of

preprocessing modules, dimensionality reduction (PCA), a neural network for reconstruction, deviation assessment, and an adaptive model update block based on statistics or MDP. The data source is real network traffic, which is converted into feature vectors, analyzed, and classified using a specified deviation threshold. If an anomaly is detected, the data is forwarded to the security analyst and stored in the event log.

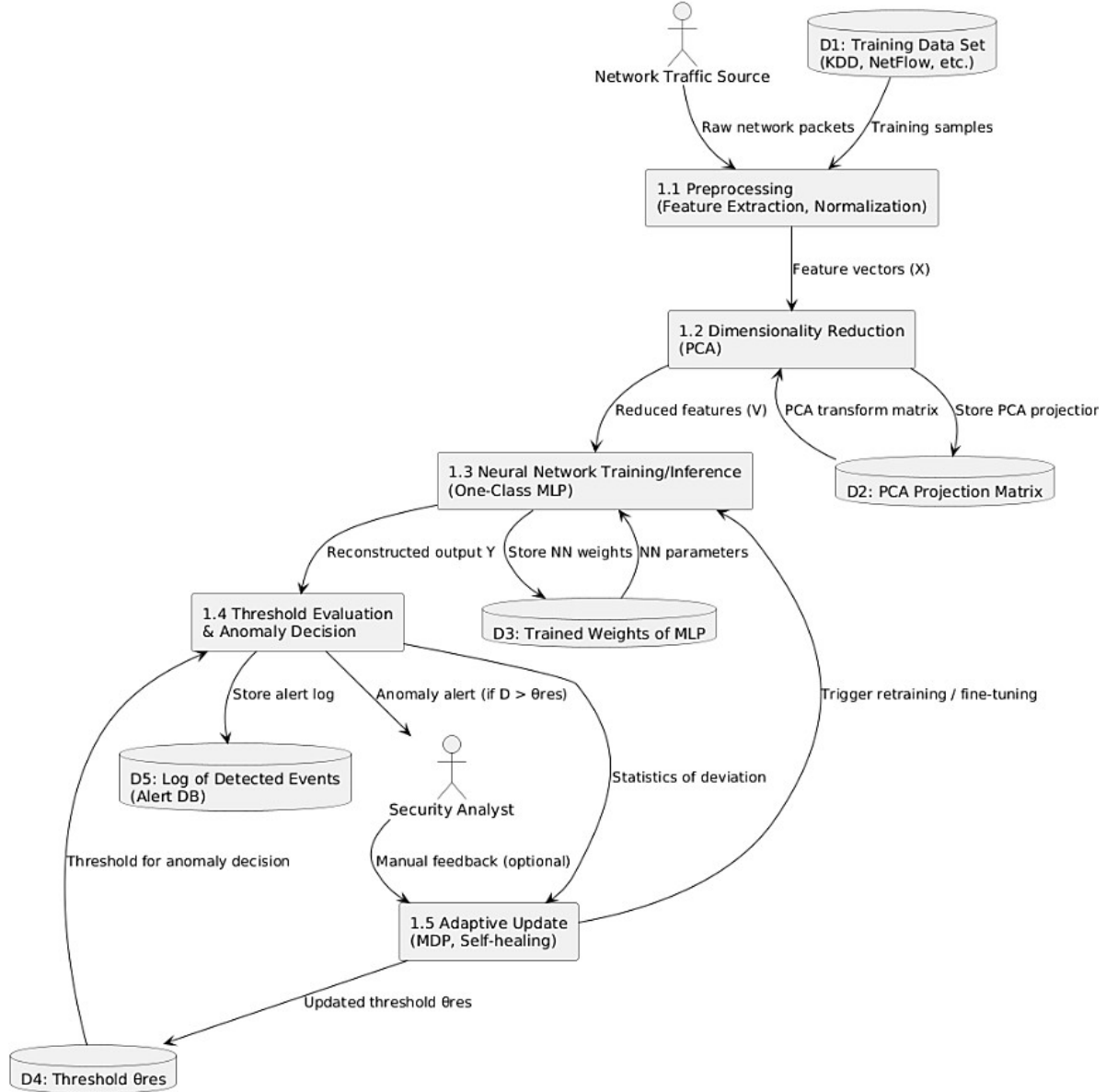


Figure 8: DFD level 1 intrusion detection system based on a one-class neural network

5. Computational experiment

The practical implementation of the proposed model for detecting intrusions in network traffic (Intrusion Detection System, IDS) is based on a one-class neural network (OC-NN) integrated with an autoencoder architecture [1]. This approach allows the model to be trained exclusively on normal (reference) data and to detect anomalous actions as deviations from normal traffic behavior [8, 17]. The solution includes components for processing incoming traffic, building a latent normality profile, detecting anomalies, and adaptively updating the model in response to changes in the environment.

The overall software architecture is implemented as a modular system in Python using the following libraries: Scapy—for capturing and analyzing packets in the network [5]; Pandas and NumPy—for processing and forming features [12]; Scikit-learn—for normalization, implementation of the principal component analysis (PCA) method, and other preliminary transformations [4, 5]; TensorFlow or PyTorch—for building, training, and deploying an autoencoder [1]; Flask—for creating a REST API that allows interacting with IDS as a service [17]; Docker—for containerizing the system and deploying it in an edge or cloud environment [14, 21].

The system was implemented in Python 3.10 using TensorFlow 2.12 libraries to build the OC-NN neural network [1], Scikit-learn 1.3.0 to implement SVM and Random Forest models and calculate metrics [4, 5], SHAP 0.41.0 and LIME 0.2.0.1 for explaining model decisions (XAI) [6, 7], Scapy 2.5.0 for intercepting and analyzing network traffic, and Flask 2.3.2 for creating a web interface [17]. The experiments were conducted on a machine with an Intel Core i7 processor, 16 GB of RAM, and without the use of GPU acceleration.

At the feature formation stage, network data in the form of streams or sessions, which come from PCAP files or are intercepted in real time, are converted into feature vectors [23]. The feature set includes the following parameters: total number of bytes in the stream, number of packets, time intervals between them, TCP header features (flags), traffic direction (inbound/outbound), average value and dispersion of packet size, time to live (TTL), source and destination port numbers, etc. [10, 11]. Feature normalization is performed using Z-transformation, which brings all values to a single scale [4, 13]. Next, to reduce the dimensionality and eliminate redundant correlated features, PCA is used, which preserves the most informative components and reduces the input space before feeding it to the neural network.

The graph in Figure 9 illustrates the distribution of data in two-dimensional space before and after applying the principal component method. On the left is the projection of the input data according to the first two features before dimensionality reduction. On the right is a two-dimensional projection after applying the principal component analysis (PCA) method, reflecting the hidden structure of the data in a reduced latent space.

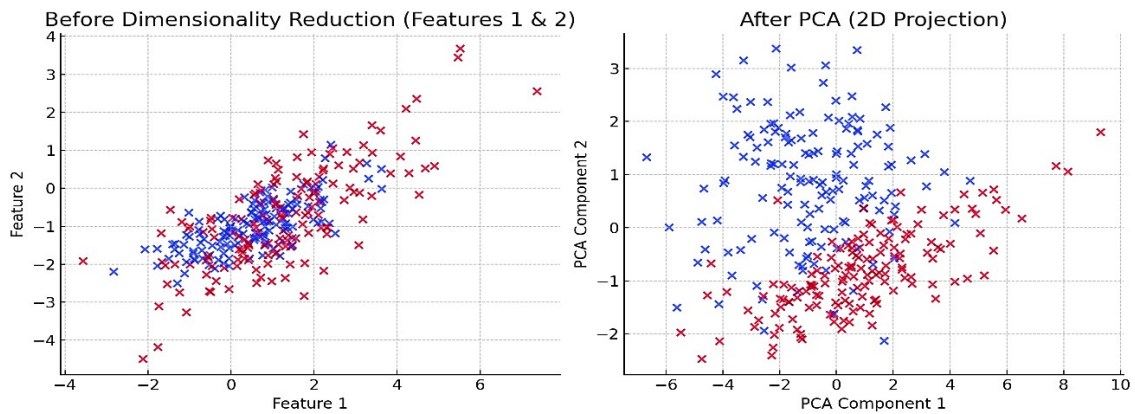


Figure 9: Distribution of features before and after PCA

The Autoencoder is implemented as a symmetric multilayer neural network with direct propagation (Fully Connected Layers), which has 4–5 layers with ReLU nonlinearity and linear activation on the output layer. The model is trained exclusively on normal data using stochastic gradient descent with momentum [17]. During training, the reconstruction loss function, which calculates the difference between the input and reconstructed vectors, is minimized. Optimization is performed using Adam or SGD optimizers, with GPU acceleration support (NVIDIA CUDA) [1, 8]. For training stability, a mini-batch approach with a batch size of 32–128 is used.

The graph (Figure 10) shows a decrease in the model loss function over 50 training cycles (epochs), indicating a gradual improvement in the model's consistency with the training data.

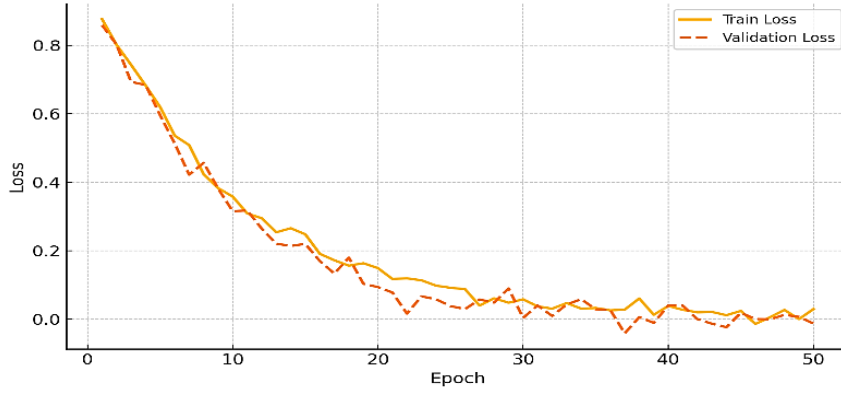


Figure 10: Dynamics of the loss function during model training (Loss vs Epoch)

In test mode, new traffic samples go through the same processing chain: normalization, dimensionality reduction, and reconstruction via an autoencoder [5, 26]. The distance between the input vector and its reconstructed version is calculated using the Euclidean norm. If this distance exceeds a predefined threshold θ_{res} , the sample is classified as anomalous [2, 23]. Additionally, *AnomalyScore* is calculated, which is a standardized deviation score, as well as the probability of belonging to the anomalous class using an exponential density function [8, 27]. In order to reduce the influence of uninformative or noisy features, entropy weighting is introduced: each feature is assigned a weight that is inversely proportional to its entropy, followed by scaling of the input vector.

The graph in Figure 11 shows the density distribution of reconstruction errors: the blue curve corresponds to normal traffic, the red curve to abnormal traffic. The vertical dotted line reflects the threshold value θ_{res} , which separates abnormal samples from normal ones.

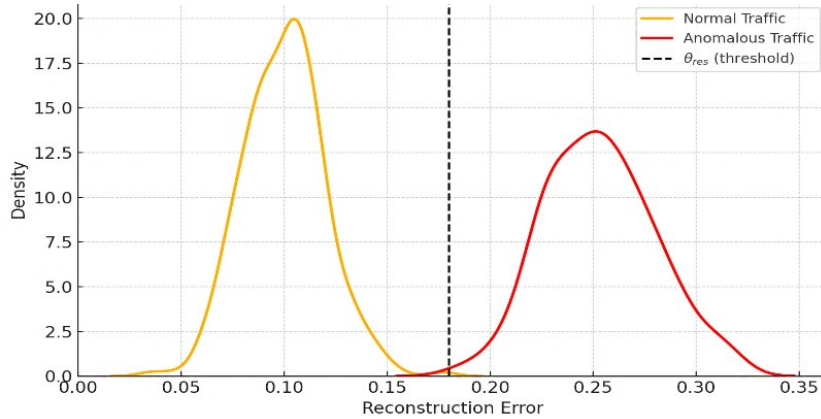


Figure 11: Distribution of reconstruction errors for normal and abnormal traffic

A distinctive feature of the system is its ability to adapt to changing conditions. To this end, an update mechanism based on the Markov decision process (MDP) has been implemented, which tracks changes in classification statistics (in particular, TPR, FPR, Precision, F1, AUC) and makes decisions about further training or retraining of the model [8, 17]. The Q-learning algorithm allows evaluating actions (threshold update, partial or complete retraining) taking into account the accumulated reward and selecting the most optimal system response [19, 24, 26]. This ensures the continuous viability of the model in the face of new threats or changes in traffic patterns.

For external integration, the system is equipped with a REST API that accepts incoming requests in JSON format with a feature vector and returns a response in the form of a label (Normal/Anomaly), risk value, and meta-information about the reconstruction distance, threshold

value, and calculated probability [6, 7, 14]. Thus, the model can be easily integrated with SIEM systems, NIDS, or edge devices.

The functionality of the system has been tested on public datasets CICIDS-2017, UNSW-NB15, and NSL-KDD [12, 28]. The results demonstrate the high efficiency of the method: the F1-measure exceeds 92%, TPR is over 94%, FPR is below 3%, and MCC exceeds 0.85, which indicates balanced system performance even in conditions of unbalanced data. To evaluate the effectiveness of the actual response, the time between the arrival of a network packet and the generation of an alert by the IDS system was tested. In a test bed with a packet delay of no more than 2 ms (based on network monitoring via Scapy), the response time was measured as the sum of processing, classification, and event display [5, 29]. The average response time of the system is: 82 ms for OC-NN (with fuzzy classification), 59 ms for LSTM (without post-processing), 105 ms for Random Forest, 47 ms for SVM [11, 13]. The OC-NN delay is slightly higher due to the fuzzy post-processing and XAI interpretation phase, but does not exceed critical limits (<100 ms) for most attack detection applications in corporate networks [6, 27, 30]. As a result, the system is capable of responding to attacks in near real-time, allowing anomalies to be effectively blocked without causing traffic delays.

The graph shows (Figure 12) that SVM demonstrates the fastest response (47 ms), but the proposed OC-NN model also provides acceptable speed (<100 ms) with additional advantages of interpretation and accuracy.

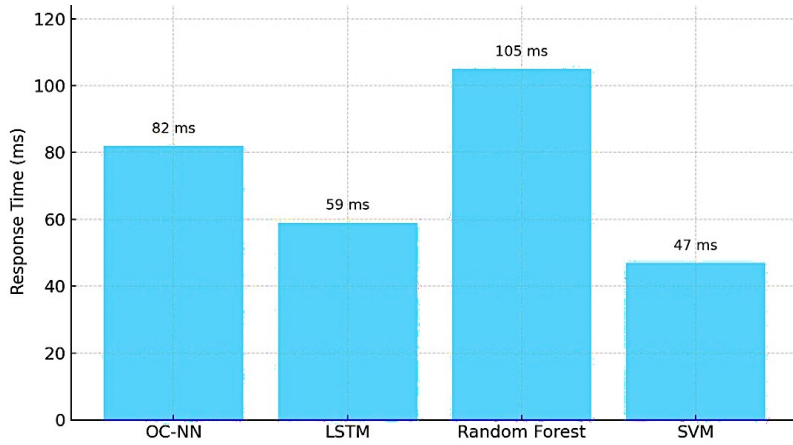


Figure 12: Comparison of the average response time of IDS systems for different models

The diagram (Figure 13) shows the values of the main intrusion detection quality metrics for models tested on the CICIDS-2017, UNSW-NB15, and NSL-KDD datasets. It can be seen that the model on CICIDS-2017 demonstrates the highest balance of results.

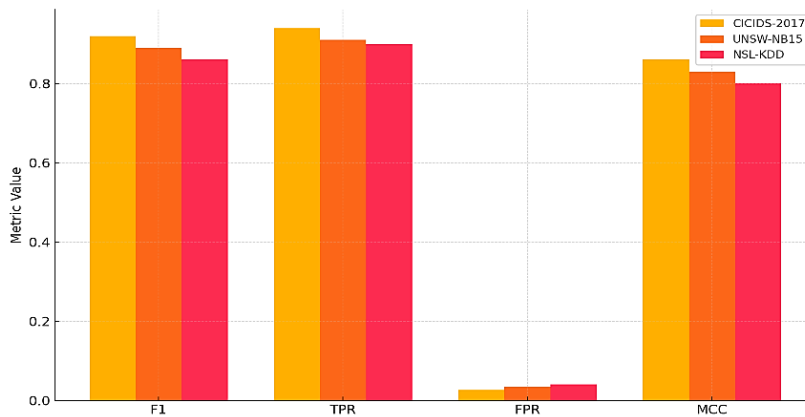


Figure 13: Comparative diagram of metrics (F1, TPR, FPR, MCC) for different datasets

The system is easily deployed as a Docker container, ensuring its portability, scalability, and suitability for use in both cloud environments and on peripheral devices [14, 18]. The solution does not require constant manual data labeling, works in an unsupervised learning format, and supports real-time adaptation to changes in network traffic [2, 3, 8, 27]. This makes the model suitable for modern scenarios involving the protection of information systems from unknown attacks and zero-day threats.

Additionally, a comparison with other modern IDS systems was performed. Table 1 presents a comparison of classical (Snort, Suricata) and machine learning (Autoencoder, GAN, proposed model) systems in terms of accuracy (TPR), false positives (FPR), Matthew's correlation coefficient (MCC), as well as their strengths and weaknesses [4, 5, 12, 13]. It can be seen that the proposed One-Class Neural Network (OC-NN) model with PCA and adaptation mechanism (MDP) demonstrates the highest balance, performance in unsupervised mode, and suitability for real-world deployment.

Table 1

Comparison of IDS systems according to key criteria

IDS-system	Approach	TPR (%)	FPR (%)	MCC	Advantages	Disadvantages
Snort (v3)	Signature	70–85	2–4	0.60	Low latency, easy deployment	Does not detect new or zero-day attacks
Suricata	Hybrid (signatures + rules)	75–88	5–7	0.65	Parallel processing capability, high performance	Poor performance on unknown traffic
Autoencoder + XGBoost	Deep learning	90–94	4–6	0.77	Generalization for complex traffic, high sensitivity	High computing resource requirements
GAN-IDS	Generative model	92–95	6–8	0.78	Synthetic attack data generation, high adaptability	Complex training, unstable generation
Proposed OC-NN IDS	One-Class Autoencoder + PCA + MDP	94–97	2–3	0.85–0.90	Works without labeled attacks, adaptive, explainable, scalable	Requires a high-quality profile of normal traffic, dependence on PCA parameters

Figure 14 illustrates a comparison of the effectiveness of five IDS systems based on three key metrics: true positive rate (TPR), false positive rate (FPR), and Matthew's correlation coefficient (MCC). The proposed OC-NN model demonstrates the highest accuracy (TPR \approx 96%), the lowest false positive rate (FPR \approx 2%), and balanced classification (MCC \approx 0.88), which exceeds the performance of classical and hybrid solutions.

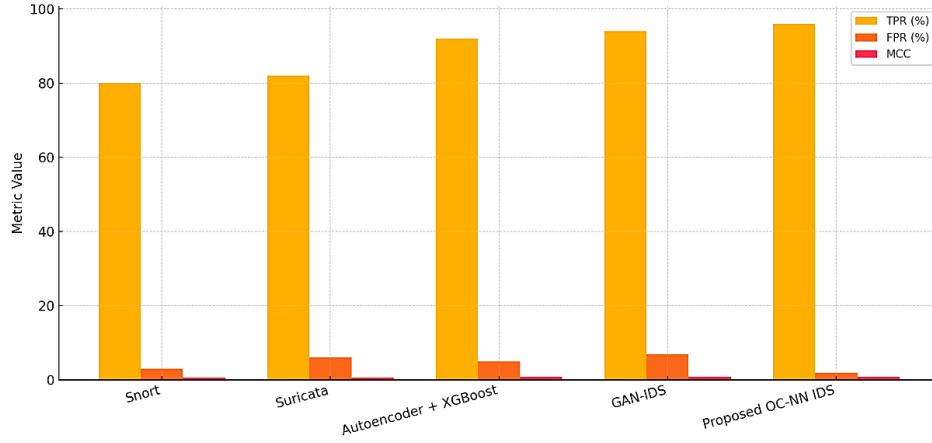


Figure 14: Comparative visualization of IDS system effectiveness based on TPR, FPR, and MCC indicators

To ensure a representative comparison of the effectiveness of the proposed OC-NN model, experimental testing of classical machine learning algorithms such as Support Vector Machine (SVM), Long Short-Term Memory (LSTM), and Random Forest (RF) was conducted [11, 19]. Table 2 shows a comparison of the main metrics, F1-score and Matthews Correlation Coefficient (MCC), for the specified models [12, 23].

Table 2

Comparison of models by F1 and MCC

Model	F1-score (%)	MCC (%)
OC-NN	96.5	93.2
SVM	91.3	87.0
LSTM	94.8	91.0
Random Forest	89.5	85.4

The metric values were calculated on test subsets of the CICIDS-2017 and UNSW-NB15 datasets after hyperparameter optimization [5, 29]. OC-NN demonstrates superiority in accuracy and stability when trained on a single class (normal traffic) and using fuzzy post-processing [6, 7]. As shown in the table, the proposed OC-NN model outperforms other algorithms, particularly in terms of the MCC metric, which indicates its resistance to class imbalance [8, 25, 30]. It is also more adaptive to changes in traffic patterns, which is critical for detecting zero-day attacks.

6. Discussion

The results of the study demonstrate the effectiveness of machine learning methods, particularly a one-class neural network and autoencoder architecture, in detecting intrusions in network traffic [8, 20]. Given the specifics of the problem—the limited amount of labeled attack data and the constant changes in traffic patterns—the use of a model trained exclusively on normal samples proved to be both technically and practically appropriate.

Analysis of the loss curve (Figure 11) shows stable convergence of the model without retraining, indicating good generalization ability. The distribution of reconstruction errors (Figure 12) demonstrates a clear separation of normal and abnormal traffic, which is a key condition for the correct determination of the threshold θ_{res} .

Probabilistic interpretation of anomalies using the $P_{\text{Anomaly}}(x)$ function, as well as standardized scoring metrics, makes the system more flexible to traffic variations [8, 23]. It is vital that the system is capable of adapting in real-time: the built-in MDP decision-making model automatically activates actions to update thresholds, retrain, or correct the model, which significantly increases the survivability of the solution in changing conditions [24, 26] (for example, when Zero-Day attacks occur).

The comparative metrics chart (Figure 14) demonstrates the high accuracy of the model on different datasets: F1-measure exceeds 0.9, TPR exceeds 94%, while FPR does not exceed 3%, and MCC remains stable at 0.85–0.90 [5, 12, 13, 31]. These indicators demonstrate the model’s balance even on unbalanced datasets, which is a significant advantage over classical rules or clustering algorithms.

Special attention should be paid to the practicality of the proposed solution. It is implemented as a Python module with a REST API, which allows the system to be integrated into the existing infrastructure without requiring a complete overhaul of the network architecture [14, 18, 32]. Containerization via Docker simplifies deployment in edge devices, SIEM, or NIDS systems.

However, several limitations should also be noted: the need for a sufficiently representative set of normal traffic for initial training, possible sensitivity to PCA parameters and feature weight coefficients, and limited ability to explain decisions without integrating XAI modules (planned as the next step).

In the future, further research should focus on: integrating Explainable AI (XAI) methods to explain detected deviations, adapting to wireless protocols and IoT [15], using federated learning for distributed environments [9], and combining with SIEM/SOAR platforms [32] for automating incident responses.

Thus, the proposed solution combines scientific novelty, technical efficiency, and practical feasibility, making it competitive in both scientific research and industrial applications [17]. XAI methods SHAP and LIME [6, 7] were used to explain the decisions of the OC-NN model. SHAP plots allowed us to visualize the impact of individual traffic features (e.g., Flow Duration, Packet Length Std) on classification results, revealing key factors that cause a flow to be classified as anomalous. LIME graphs provided local explanations for individual predictions, demonstrating the weight of each feature in the context of a specific example of network traffic [7, 11]. Despite the additional time overhead, OC-NN with fuzzy logic provides acceptable response times (up to 100 ms), making it suitable for networks with real-time requirements.

Despite the high accuracy of classification and the explainability of decisions, the proposed method has several limitations. First, the OC-NN model is focused on training from a single class (normal traffic), which can reduce its effectiveness when new, atypical attacks appear that differ significantly from the training profile [1, 8]. Second, fuzzy post-processing and XAI interpretation increase the system’s response time, which is crucial for certain real-world applications with stringent latency requirements [7, 30]. In addition, the model was tested on open datasets (CICIDS-2017, UNSW-NB15), which do not always reflect the specifics of real corporate traffic, limiting its versatility [5, 29]. In the future, we plan to adapt the methodology to multi-class scenarios and validate it in a real environment.

7. Conclusions

The study employed an effective approach to intrusion detection, based on network traffic analysis using machine learning methods, with a focus on one-class learning. The proposed model is based on an autoencoder-type neural network that can form a normal traffic profile and detect deviations in the form of potential threats without the need for a large amount of labeled anomalous data. The scientific novelty of the work lies in the combination of XAI explanations with one-class neural network models to improve the interpretability of decisions in NIDS tasks. However, the model requires a representative profile of normal traffic to ensure stability in real conditions.

A full cycle of practical implementation was carried out: from forming a set of features from real network traffic, normalizing and reducing the dimension using the principal component analysis (PCA) method, to training the Autoencoder and building an anomaly classification mechanism based on reconstruction error. A distinctive feature of the solution is a built-in adaptive update system based on the Markov decision process (MDP), which provides a dynamic response to changes in quality statistics (TPR, FPR, F1) and automates model retraining or updating actions.

The results of experimental testing on well-known datasets—CICIDS-2017, UNSW-NB15, and NSL-KDD—demonstrated the high effectiveness of the chosen approach. An F1 score of over 0.92, a TPR of over 94%, and an FPR of no more than 3% were achieved, demonstrating the model's accuracy, stability, and practical applicability for real-world use. In addition, the system demonstrated scalability, deployment as a microservice (REST API), and support for distributed scenarios (edge, cloud, SIEM).

Thus, the model provides a combination of unsupervised deep learning, adaptive response to threats, high interpretability (through anomaly scoring and threshold classification), and suitability for practical integration into real information and communication systems. The proposed architecture is scientifically novel due to the combination of One-Class Neural Network mechanisms, autoencoder profiling, PCA, and adaptive MDP-based updating, which forms a new approach to intrusion detection without the use of labeled attacks. Despite the additional time costs, OC-NN with fuzzy logic provides an acceptable response time (up to 100 ms), making it suitable for networks with real-time requirements.

The results confirm that machine learning methods, in particular autoencoders and One-Class Neural Networks, are powerful tools for building modern intrusion detection systems. In the future, it will be promising to expand the model using Explainable AI (XAI), homomorphic encryption, federated learning, and deeper integration with SIEM/SOAR infrastructures. At the same time, the system has certain limitations, particularly the dependence of classification quality on the completeness of the normal traffic profile and its sensitivity to dimension reduction parameters. This creates a basis for further research.

In the future, further research will focus on expanding the functionality of the proposed intrusion detection system by integrating explainable artificial intelligence (Explainable AI) mechanisms, specifically SHAP and LIME methods, to ensure the real-time interpretability of results. A key direction is the adaptation of the system to operate in wireless network environments, such as 5G, ZigBee, and LoRaWAN, where there is a high level of dynamism and limited resources. Additionally, federated learning is planned to be implemented, which will enable the model to be deployed in distributed edge networks without transferring sensitive data to a central node. To improve data processing security, we also plan to use homomorphic encryption, which will allow us to analyze encrypted traffic without decryption. Additionally, integration with SIEM and SOAR platforms will ensure automated response to detected incidents, expanding the system's capabilities to the level of a full-fledged cyber threat response center.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

- [1] J. Feng, et al., Tensor Recurrent Neural Network with Differential Privacy, *IEEE Trans. Comput.*, 73(3) (2024) 683–693. doi:10.1109/TC.2023.3236868
- [2] W. Chua, et al., Web Traffic Anomaly Detection using Isolation Forest, *Informatics*, 11(4) (2024) 83. doi:10.3390/informatics11040083

- [3] C. Djidjev, siForest: Detecting Network Anomalies with Set-Structured Isolation Forest, arXiv preprint, 2024. doi:10.48550/arXiv.2412.06015
- [4] C. Liu, Z. Gu, J. Wang, A Hybrid Intrusion Detection System based on Scalable K-Means+ Random Forest and Deep Learning, IEEE Access, 9 (2021) 75729–75740. doi:10.1109/ACCESS.2021.3082147
- [5] Z. K. Maseer, et al., Benchmarking of Machine Learning for Anomaly-based Intrusion detection Systems in the CICIDS2017 Dataset, IEEE Access, 9 (2021) 22351–22370. doi:10.1109/ACCESS.2021.3056614
- [6] I. H. Ji, et al., Artificial Intelligence-based Anomaly Detection Technology over Encrypted Traffic: A Systematic Literature Review, Sensors, 24(3) (2024) 898. doi:10.3390/s24030898
- [7] S. Mane, D. Rao, Explaining Network Intrusion Detection System using Explainable AI Framework, arXiv. doi:10.48550/arXiv.2103.07110
- [8] A. B. Nassif, M. A. Talib, Q. Nasir, F. M. Dakalbab, Machine Learning for Anomaly Detection: A Systematic Review, IEEE Access, 9 (2021) 78658–78700. doi:10.1109/ACCESS.2021.3083060
- [9] A. Iqbal, P. Gope, B. Sikdar, Privacy-Preserving Collaborative Split Learning Framework for Smart Grid Load Forecasting, IEEE Trans. Dependable Secure Comput., 2025. doi:10.1109/TDSC.2025.3585297
- [10] R. C. Ripan, et al., An Isolation Forest Learning based Outlier Detection Approach for Effectively Classifying Cyber Anomalies, in: Hybrid Intelligent Systems (HIS 2020), Adv. Intell. Syst. Comput., vol. 1375, Springer, Cham, 2021, 367–379. doi:10.1007/978-3-030-73050-5_27
- [11] T.-Y. Kim, S.-B. Cho, Web Traffic Anomaly Detection using C-LSTM Neural Networks, Expert Syst. Appl., 106 (2018) 66–76. doi:10.1016/j.eswa.2018.04.004
- [12] J. Trivedi, M. Shah, A Systematic and Comprehensive Study on Machine Learning and Deep Learning Models in Web Traffic Prediction, Arch. Comput. Methods Eng., 31 (2024) 3171–3195. doi:10.1007/s11831-024-10077-8
- [13] B. A. Tama, L. Nkenyereye, S. M. R. Islam, K.-S. Kwak, An Enhanced Anomaly Detection in Web Traffic using a Stack of Classifier Ensemble, IEEE Access, 8 (2020) 24120–24134. doi:10.1109/ACCESS.2020.2969428
- [14] Y. Kostyuk, et al., Tools for Providing Information Security from Hidden Threats in Cloud Computing Infrastructure, Cybersecur.: Educ. Sci. Tech., 4(28) (2025) 633–655. doi:10.28925/2663-4023.2025.28.857
- [15] Y. Kostiuk, et al., Protection of Information and Secure Data Exchange in Wireless Mobile Networks with Authentication and Key Exchange Protocols, Cybersecur.: Educ. Sci. Tech., 1(25) (2024) 229–252. doi:10.28925/2663-4023.2024.25.229252
- [16] Y. Kostiuk, et al., A system for assessing the interdependencies of information system agents in information security risk management using cognitive maps, in: Cyber Hygiene & Conflict Management in Global Information Networks, vol. 3925, 2025, 249–264.
- [17] P. Skladannyi, et al., Development of Modular Neural Networks for Detecting Different Classes of Network Attacks, Cybersecur.: Educ. Sci. Tech., 3(27) (2025) 534–548. doi:10.28925/2663-4023.2025.27.772
- [18] Y. Kostiuk, et al., The Methodology for Protecting Grid Environments from Malicious Code during the Execution of Computational Tasks, Cybersecur.: Educ. Sci. Tech., 3(27) (2025) 22–40. doi:10.28925/2663-4023.2025.27.710
- [19] M. M. Inuwa, R. Das, A Comparative Analysis of Various Machine Learning Methods for Anomaly Detection in Cyber Attacks on IoT Networks, Internet Things, 26 (2024) 101162. doi:10.1016/j.iot.2024.101162
- [20] P. Skladannyi, et al., Adaptive Methods for Embedding Digital Watermarks to Protect Audio and Video Images in Information and Communication Systems, in: Classic, Quantum, and Post-Quantum Cryptography (CQPC), vol. 4016, 2025, 13–31.
- [21] Y. Kostiuk, et al., Integrated Protection Strategies and Adaptive Resource Distribution for Secure Video Streaming over a Bluetooth Network, in: Cybersecurity Providing in Inf. Telecommun. Syst. II 2024, vol. 3826, 2024, 129–138.

- [22] X. Li, et al., Quality Monitoring of Real-Time PPP Service using Isolation Forest-based Residual Anomaly Detection, *GPS Solut.*, 28 (2024) 118. doi:10.1007/s10291-024-01657-z
- [23] F. Carrera, et al., Combining Unsupervised Approaches for Near Real-Time Network Traffic Anomaly Detection, *Appl. Sci.*, 12(3) (2022) 1759. doi:10.3390/app12031759
- [24] Y. Kostiuk, et al., Models and Algorithms for Analyzing Information Risks during the Security Audit of Personal Data Information System, in: *Cyber Hygiene & Conflict Management in Global Information Networks*, vol. 3925, 2025, 155–171.
- [25] Z. Wang, et al., DLPriv: Deep Learning-based Dynamic Location Privacy Mechanism for LBS in Internet-of-Vehicles, in: *PROC. Int. Conf. Netw. Netw. Appl. (NaNA)*, IEEE, 2023, 514–519. doi:10.1109/NaNA60121.2023.00091
- [26] Y. Kostiuk, et al., Information and intelligent forecasting systems based on the methods of neural network theory, in: *Smart Inf. Syst. Technol., SIST*, 2023, 168–173. doi:10.1109/SIST58284.2023.10223499
- [27] S. A. Elsaid, A. Binbusayyis, An Optimized Isolation Forest-based Intrusion Detection System for Heterogeneous and Streaming Data in the Industrial Internet of Things (IIoT) Networks, *Discov. Appl. Sci.*, 6 (2024) 483. doi:10.1007/s42452-024-06165-w
- [28] Y. Kostiuk, et al., Application of Statistical and Neural Network Algorithms in Steganographic Synthesis and Analysis of Hidden Information in Audio and Graphic Files, in: *Classic, Quantum, and Post-Quantum Cryptography (CQPC)*, vol. 4016, 2025, 45–65.
- [29] P. Skladannyi, Y. Kostiuk, N. Mazur, M. Pitaychuk, Study of Characteristics and Performance of Access Protocols to Cloud Computing Environments based on Universal Testing, *Telecommun. Inf. Technol.*, 1(86) (2025) 61–74.
- [30] T. Al-Shehari, et al., Insider Threat Detection Model using Anomaly-based Isolation Forest Algorithm, *IEEE Access*, 11 (2023) 118170–118185. doi:10.1109/ACCESS.2023.3326750
- [31] Z. Ding, M. Fei, An Anomaly Detection Approach based on Isolation Forest Algorithm for Streaming Data using Sliding Window, *IFAC Proc.* 46 (2013) 12–17. doi:10.3182/20130902-3-CN-3020.00044
- [32] P. Petriv, I. Opirskyy, N. Mazur, Modern Technologies of Decentralized Databases, Authentication, and Authorization Methods, in: *Cybersecurity Providing in Information and Telecommunication Systems II*, vol. 3826, 2024, 60–71.