

Detection of attacks on computer networks using machine learning – predictive model

Julia Stępień¹, Imed El Fray^{1,2*} 

¹ Faculty of Computer Sciences and Information Technology, West Pomeranian University of Technology in Szczecin, Al. Piastów 17, Szczecin, Poland

² Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences, Nowoursynowska 159, Warsaw, Poland

* Corresponding author's e-mail: ielfray@zut.edu.pl

ABSTRACT

Intrusion detection systems based on machine learning achieve high effectiveness under optimal testing conditions. However, real environments come with limitations such as changing data characteristics, previously unknown attacks or low and delayed label availability. To address these challenges, this paper proposed an approach of continuous adaptation to streaming data in the form of active incremental learning. This solution further incorporates a semi-supervised learning approach to reduce expert involvement. The effectiveness of the approach was assessed under conditions simulating real-time system operation. The results of the performed tests show that selective use of expert assistance maintains high classification performance while minimising labelling costs. An F1 score of 0.9930 was achieved using 6% of labels, and 0.9803 with approximately 2%, indicating that a limited number of expert labels is sufficient for effective model adaptation. These findings confirm the effectiveness of the proposed adaptation mechanism for intrusion detection in streaming environments.

Keywords: intrusion detection system, incremental learning, active learning, cybersecurity.

INTRODUCTION

The rapid growth of computer networks has led to an increased demand for effective defence against cyber threats. Traditional security methods based on rules and signatures, such as firewalls, IDS and IPS systems, have limited adaptability in the face of new types of attacks and changes in network traffic patterns.

In response to these limitations, intrusion detection systems (IDS) based on machine learning methods are becoming an increasingly popular solution. These systems can automatically analyse network traffic and detect attacks based on patterns derived from the data, thus eliminating the need for manual rule definition [1].

However, one of the key problems in the practical application of machine learning methods in IDS systems is the limited availability of labelled data (ground truth). In real-world network environments, labelling network traffic is time-consuming and often possible only for a small fraction of observed events [2]. Nevertheless, many research studies assume the full availability of labels, which significantly simplifies the learning process, but this deviates from real-world conditions.

This paper aimed to design and implement a machine learning-based IDS system that considers the streaming nature of data, variability in data distributions over time (concept drift) and limited real-world label availability. This paper

utilised the Adaptive Random Forest incremental algorithm [3] and proposed a semi-supervised learning mechanism that uses pseudo-labels [4] as well as limited administrator intervention.

The innovative approach involves adapting the designed system continuously to streaming data through active incremental learning. Furthermore, this approach uses semi-supervised learning, thereby reducing the need for expert involvement. The proposed approach effectively detects attacks under conditions that simulate real-time system operation. The experimental results obtained are consistent with the latest literature [5, 6] and validate the direction of the research.

RELATED WORKS

IT systems are exposed to a number of threats, including unauthorised access, privilege abuse and cyber-attacks. In practice, detected incidents are often recorded centrally via a Security Information and Event Management (SIEM) system [7]. SIEM systems collect data from various sources, including logs and alerts from security systems such as IDS [8, 9] and IPS [9], as well as firewalls. This allows events to be correlated and analysed in a single, centralised location. Complex incidents can be identified this way, and alerts can be sent to the administrator based on the relationships between events [10]. This paper focused on intrusion detection systems, the primary task of which is to detect potential threats based on data analysis. IPS and SIEM systems are outside the scope of this paper.

Among the threat detection techniques used by IDSs, two dominant approaches stand out: signature-based detection [9, 11, 12] and anomaly-based detection [11]. The chosen technique determines the method of data analysis and the types of threats that can be detected. The detection technique also influences the accuracy of detection and the number of false alarms.

In the case of SIDS, the IDS attack database must be regularly updated with new signatures. Without these patterns, SIDS cannot recognise threats, leaving the network or system vulnerable to zero-day attacks [13]. Anomaly-based detection (AIDS) involves creating a baseline model of normal behaviour and comparing it with monitored data. This baseline model can be constructed using statistical methods or machine learning [14]. Based on historical data,

it represents the range of normal data, with any deviation marked as an anomaly. However, the model must be continuously updated due to changes in normal behaviour over time (concept drift) [15, 16]. While this system is more effective at detecting unknown threats, it results in a higher false alarm rate.

The ability of the anomaly-based approach to detect unknown threats is what makes it a key component of intrusion detection systems (NIDS). When combined with network traffic analysis in NIDS systems [17] and machine learning models for prediction and adaptation, it can provide an effective solution. For this reason, this paper focused primarily on ML-based anomaly detection in network IDS systems.

Unlike traditional rule- and signature-based methods, machine learning (ML) models learn from data and can autonomously generate rules for identifying attacks. They can identify relationships between network traffic characteristics, and because they can generalise, they can classify unknown attacks.

Machine learning methods can be divided into two main types:

- A supervised model receives input data with assigned labels and learns patterns based on which it predicts the appropriate output value. These are divided into two categories: classification and regression. Typical algorithms include support vector machine (SVM), Random Forest (RF) and logistic regression (LR).
- An unsupervised model receives only unlabelled data and learns independently to detect patterns using statistical dependencies between samples. Popular techniques include clustering (e.g. k-means and DBSCAN), principal component analysis (PCA), and autoencoders – neural networks that reconstruct data to detect anomalies. In the context of an IDS, the model detects anomalies by identifying behaviour that differs from normal network traffic. While these models can adapt to new types of attacks, their lack of labels limits their effectiveness, resulting in a higher number of false alarms than with supervised models [18].

In practice, semi-supervised learning is also employed because it combines the approaches of the first two methods. Labelling all network traffic samples is often impossible or costly [2]. This approach strikes a balance between detection

effectiveness and the availability of training data, which is consistent with the aims of this work.

Machine learning applications are also associated with significant limitations that impact the effectiveness and stability of IDS systems in real-world environments. The datasets commonly used in research, such as KDD Cup 99 [17], NSL-KDD [19–21], UNSW-NB15 [19, 22, 23], CIC-IDS2017 [24, 25] and CSE-CIC-IDS2018 [26], do not accurately reflect modern network traffic conditions. They contain redundant records and outdated categories and attack methods, and exhibit low data diversity. This limits the ability of models trained on such data to generalise.

Furthermore, researchers sometimes fail to remove metadata features, leading to ‘shortcut learning’, whereby the model predicts results based primarily on these features (e.g. destination port). These features depend on the network configuration and conditions present when the dataset was generated and do not reflect real-world traffic.

In practice, network traffic is highly dynamic, with both its structure and the attack methods changing over time. This leads to a gradual decline in the effectiveness of the offline-trained models [27]. Another issue is the limited availability of labelled data, which hinders the use of supervised methods.

One identified issue is concept drift, whereby the relationship between input data and class labels changes over time [28]. In the context of IDS, this means that network traffic patterns are not static, with new types of attacks emerging and old ones evolving. Offline-trained models are static and do not adapt to traffic dynamics; consequently, the number of misclassifications increases over time [26, 29].

Previous research on ML-IDS has primarily focused on comparing the performance of classifiers using standard metrics such as accuracy, precision and sensitivity [30]. Many publications suggest that the models perform well on test datasets [26]. However, these solutions have not been tested under real-world conditions, which can create a false sense of optimism about the models’ actual effectiveness. This is due to limitations in the data used, as well as a lack of mechanisms for adapting models to changing network traffic. To accurately assess the real-world performance of models, additional metrics such as training time, memory usage, and false alarms must also be considered [29].

Another challenge lies in integrating models with existing security infrastructure and updating them without interrupting system operation. A high rate of IDS false alarms can result in a loss of trust in the system and reduce its usability [9]. Machine learning models should operate in near real-time to minimise the delay between threat detection and alert generation. Using machine learning models also increases computational complexity, which hinders their deployment in real-world environments [17].

Additionally, the literature notes that many existing machine learning-based IDS systems employ a one-off training approach on a static dataset. This approach does not account for the continuous generation of network data or variability in traffic chartover time, which reduces its usefulness in real-world environments [31]. To address these limitations, IDS systems based on stream processing have been proposed [17, 27, 32, 33]. These systems utilise incremental and adaptive algorithms that enable the model to be updated while the system is running, often using only a small number of labelled examples.

PROBLEM ANALYSIS, METHODOLOGY AND DATA PREPARATION

Characteristics of the IDS datasets

Network traffic is characterised by a large number of features and an imbalance between normal and malicious traffic instances. The CIC-IDS2017, CSE-CIC-IDS2018 and UNSW-NB15 datasets, which are discussed in Section 2, are widely used in ML-IDS research. Each dataset has its own limitations. CIC-IDS2017 and CSE-CIC-IDS2018 closely resemble real-world traffic, but suffer from significant class imbalance. In contrast, UNSW-NB15 features a more balanced data distribution; however, the inclusion of synthetic data means that it may not accurately reflect real-world network traffic. Model testing is typically conducted on a separate subset of data from the same dataset where the distributions do not differ significantly. Therefore, in this work, CIC-IDS2017 was used for model training and CSE-CIC-IDS2018 for evaluation due to the possibility of observing concept drift between two datasets with similar structures. The UNSW-NB15 dataset was not used for comparison because its different data generation process

makes analysing concept drift more difficult than with the CIC datasets.

Selection and justification of ML algorithms

The choice of algorithms in IDS depends on the nature of the data, the type of task (i.e. binary or multi-class classification) and the required interpretability of the model. In [34], the authors stated that the classifiers based on supervised learning are more effective at predicting both known and unknown attacks. The literature notes that the RF trained offline often outperforms other classical learning methods in tests on IDS datasets [26, 35]. RF performs well in the face of network traffic challenges due to its logarithmic-linear training time complexity, high prediction speed and resistance to class imbalance [35]. It can also reduce overfitting by aggregating multiple trees, and it handles missing values in the data well [36]. However, offline-trained IDS assumes that the data distribution remains constant, which is a flawed assumption in the context of network traffic. In [5], the authors investigated the performance of static models on new datasets. The RF models trained on CICIDS2017 and tested on CSE-CIC-IDS2018 showed a significant decline in prediction quality. One potential solution to this problem is incremental learning, which enables the model to be updated in real time based on individual incoming samples. This maintains the stability and quality of predictions without the need for full retraining. In this case, the input data enables continuous expansion of knowledge about the observed phenomena [37].

This article therefore presents the adaptive Random Forest (ARF) algorithm, which is an extension of the classic RF designed to operate under conditions of changing distributions. The selected model operates in online mode, updating its decision trees based on the input data. Each tree contains a mechanism for detecting model drift and issuing warnings. Upon receiving a warning, new trees are trained in the background. When a change in the data distribution is detected, the existing tree is replaced with one that has been trained since the warning was received, thereby maintaining prediction stability [37]. ARF enables an evaluation of the impact of the adaptive approach on classification performance under new conditions (CIC-IDS2017 and CSE-CIC-IDS2018). This evaluation was conducted across datasets (i.e. a cross-dataset

evaluation) as research on the generalisation of NIDS models indicates that the very high results obtained when training and testing on the same dataset do not necessarily translate into effectiveness on other datasets [5, 6, 38, 39]. This justifies the need to verify the models' ability to adapt to the data from a different environment with a similar structure.

Data preparation

Cleaning and merging datasets

Due to the structure of the CIC-IDS-2017 dataset, which consists of eight files, the data preparation process began with the preliminary cleaning of each file's data.

The following operations were performed:

- column names and class labels were standardised;
- empty rows were removed;
- the duplicate column 'Fwd.Header.Length.1' was removed;
- columns containing metadata were removed;
- infinite values were replaced with NaN.

In intrusion detection research datasets, there are sometimes columns containing metadata, such as destination port, IP addresses or timestamps. This information results from how traffic is generated into the dataset rather than from behavioural characteristics. Research indicates that the models trained on such features demonstrate high accuracy by learning an unnatural relationship between metadata and the label. These models do not reflect the true nature of attacks and achieve high performance on the test set because a given attack class occurs in the dataset under specific conditions. Using such features leads to unnatural correlations (shortcut learning) [40].

For this reason, six columns containing metadata were removed in order to obtain a model capable of generalising under real-world conditions. The names of the selected columns and the reasons for their removal are detailed in Table 1.

Before merging, all files were checked for empty rows. A total of 288,602 such rows were found in the 'Thursday-WorkingHours-Morning-WebAttacks.csv' file and were removed. The data were then merged into a single data structure and further analysis was performed on the entire dataset. Columns with zero variance were identified where the only unique value was 0, as can be seen from the column statistics presented in

Table 1. Metadata columns and reasons for their removal

Column name	Characteristics / reason for removal
flow_id	The traffic identifier does not exist under real-world conditions.
source_ip	IP addresses are specific to the laboratory environment and models learn to assign attacks to specific hosts.
destination_ip	The same as above applies.
timestamp	Timestamps reflect the attack generation schedule.
source_port	In practice, the source port is random.
destination_port	In the strongest case of <i>shortcut learning</i> , across many datasets, the port independently predicts the class with high accuracy [41].

Table 2, where the mean, minimum and maximum values are all 0. Thus, eight columns that did not contain meaningful information were removed. Next, 596,376 duplicates were removed from the dataset using the `drop_duplicates()` function from the pandas library. Finally, an analysis of missing values was conducted to identify the columns with missing values. Two columns were found to have a missing value rate of 0.1013%: 'flow_bytes_s' and 'flow_packets_s'. Due to feature distributions that did not follow a Gaussian distribution, the missing values were imputed using the median [41].

Exploratory data analysis

As part of the exploratory data analysis, the following were examined:

- the percentage distribution of classes,
- the distribution of attack types,
- statistical analysis of selected features,
- a correlation matrix to assess the interdependence between attributes,
- graphs of key feature characteristics,
- analysis of outliers using the 1.5xIQR rule (IQR – interquartile range).

The results are presented in Figures 1, 2 and 3:

Figure 1 shows the percentage distribution of classes. After the initial data clean-up, normal traffic accounted for 85% of the data. The second-largest class in the dataset is the DoS Hulk attack,

accounting for around 7.7% of the total. This confirms a significant imbalance between normal traffic and attacks.

Figure 2 shows the frequency of attack classes in the dataset. The least frequent class is Heartbleed, accounting for just 0.00049% of labels in the CIC-IDS2017 dataset — only 11 instances compared to the 172,849 DoS Hulk attacks. This further indicates an imbalance between the different attack types. Such a data structure hinders the effective training of multi-class models and may result in rare classes being undetected [42], which is why binary classification was chosen instead.

Figure 3 presents a correlation heatmap illustrating the relationships between input features. Groups of highly correlated features are visible as red clusters, indicating information redundancy in the dataset. Some features describe similar network traffic properties: for instance, `fwd_iat_mean`, `fwd_iat_std`, `fwd_iat_max` and `fwd_iat_min` all relate to the time interval between successive data packets arriving in the network. Feature selection is therefore justified.

Table 3 lists the 14 features for which the outlier rate exceeds 20%. The presence of outliers is characteristic of network data [43]. These values have not been removed in order to preserve the realistic nature of the data stream.

Table 2. Features with zero variance

Parameter	count	mean	std	min	25%	50%	max
bwd_psh_flags	2,830,743.0	0.0	0.0	0.0	0.0	0.0	0.0
bwd_urg_flags	2,830,743.0	0.0	0.0	0.0	0.0	0.0	0.0
fwd_avg_bytes_bulk	2830743.0	0.0	0.0	0.0	0.0	0.0	0.0
fwd_avg_packets_bulk	2,830,743.0	0.0	0.0	0.0	0.0	0.0	0.0
fwd_avg_bulk_rate	2,830,743.0	0.0	0.0	0.0	0.0	0.0	0.0
bwd_avg_bytes_bulk	2830743.0	0.0	0.0	0.0	0.0	0.0	0.0
bwd_avg_packets_bulk	2830743.0	0.0	0.0	0.0	0.0	0.0	0.0
bwd_avg_bulk_rate	2830743.0	0.0	0.0	0.0	0.0	0.0	0.0

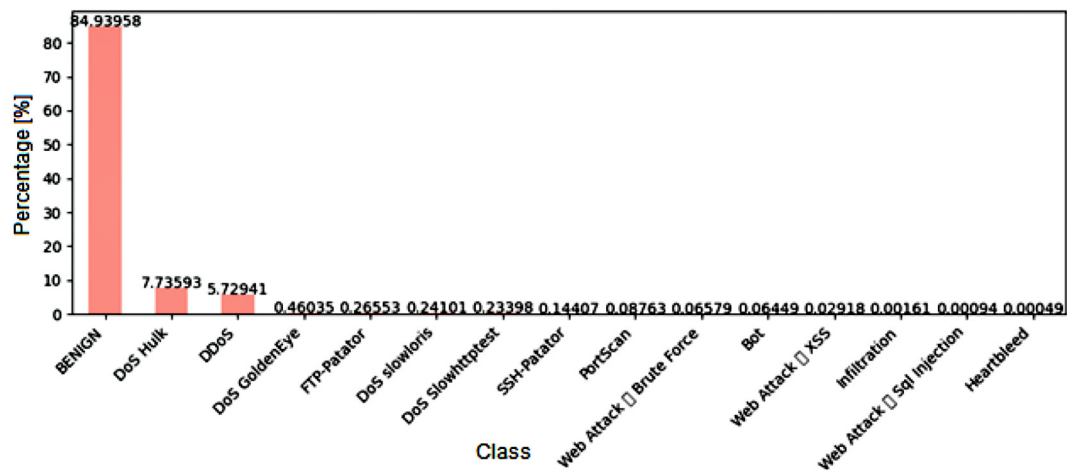


Figure 1. Percentage distribution of classes in the dataset

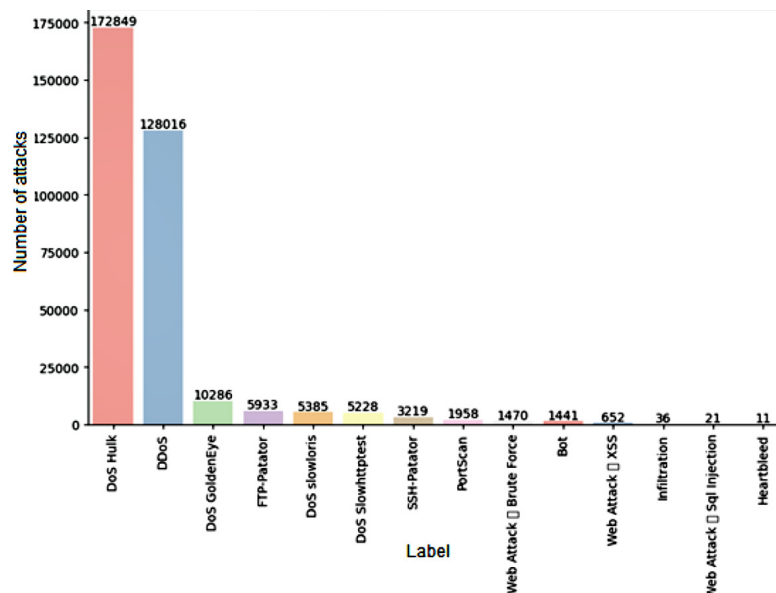


Figure 2. Distribution of attack types by frequency

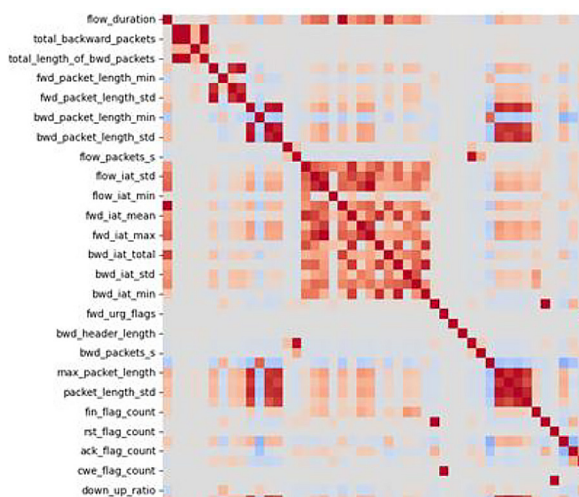


Figure 3. Feature correlation matrix in heatmap form

Feature selection

Feature selection was performed using the recursive feature elimination (RFE) method, with a Random Forest [20] classifier acting as the estimator. This method was chosen based on the research results presented in the literature, which indicate its high effectiveness for the CIC-IDS2017 dataset [39, 44].

Ultimately, 14 features were selected to reduce the computational complexity of the model, as shown in Table 4.

Ultimately, 14 features were selected, as shown in Table 4. Reducing the feature space from 78 to 14 core attributes not only improves

Table 3. Number and percentage of outliers for individual traits – above 20%

Feature	Number of outliers	Percentage of outliers [%]
flow_duration	463,068	20.72
flow_bytes_s	487,114	21.80
flow_packets_s	516,793	23.13
flow_iat_min	466,115	20.86
fwd_iat_total	454,646	20.35
fwd_iat_min	466994	20.90
bwd_iat_total	487004	21.80
bwd_iat_mean	486873	21.79
bwd_iat_max	496822	22.24
fwd_packets_s	510697	22.86
bwd_packets_s	482992	21.62
active_mean	554857	24.83
active_max	554866	24.83
active_min	554423	24.81

model generalisation, but also lowers the mathematical operations required per incoming flow, directly decreasing inference latency during online stream ingestion.

Preparation of training and test sets

The dataset was divided into training and test sets at an 80:20 ratio using random sampling, with class proportions preserved (stratification). The training set was used to train the models and the test set was used to conduct a preliminary evaluation under stable conditions.

To further verify the models' resilience to data variability and adaptability, two additional test datasets were prepared from the CSE-CIC-IDS2018 dataset. The data from February 16th and 21st 2018 were selected from the available files in the dataset. The first file contains DoS attacks (DoS-SlowHTTPTest and DoS-Hulk) [45], which are also included in CIC-IDS2017. The second file includes the DDoS attacks (DDoS-LOIC-UDP and DDoS-HOIC) representing new patterns of network traffic.

This selection of test sets allows for a comparison of model behaviour in two scenarios:

- detection of attacks known from the training process but occurring under different network conditions; and:
- detection of new attacks.

This enables evaluating the sensitivity of the models to changes caused by differences in the network environment, as well as their ability to adapt to new types of attacks. This approach is consistent with the concept of cross-dataset evaluation, which, as previously noted, is a more reliable solution than testing on a separate portion of the same dataset [5, 6].

For all test datasets, feature names were mapped to a format consistent with the training dataset. An identical data cleaning process was applied to ensure consistency across experiments. Table 5 presents the final model parameters used, including the number of trees, depth, distribution change detector settings and other hyperparameters.

Table 4. Descriptions of selected features

Selected trait	Description
total_fwd_packets	Total number of packets sent in the forward direction
total_length_of_bwd_packets	Sum of the lengths of all packets in the backward direction
bwd_packet_length_max	Maximum packet length in the backward direction
bwd_packet_length_mean	Average packet length in the backward direction
bwd_packet_length_std	Standard deviation of packet lengths in the backward direction
bwd_iat_mean	Mean interval between consecutive packets in the backward direction
fwd_iat_max	Maximum interval between packets in the forward direction
bwd_packets_s	Number of packets sent in the backward direction per second
max_packet_length	Maximum packet length
packet_length_mean	Average packet length
packet_length_std	Standard deviation of packet length
packet_length_variance	Variance of packet length
average_packet_size	Average packet size
avg_bwd_segment_size	Average backward data segment size

Table 5. Adaptive Random Forest model parameters

Parameter	Value
Number of trees (n_models)	7
Poisson parameter (λ)	4
Maximum tree depth	20
Alert detector	ADWIN ($\delta = 0.01$)
Drift detector	ADWIN ($\delta = 0.001$)
Training mode	Streaming (incremental)
Library	River

EXPERIMENT AND RESULTS

Architecture of the proposed IDS system

A typical intrusion detection system (IDS) architecture consists of components responsible for collecting, processing and analysing network traffic, and reporting detected incidents. Network sensors monitor traffic and collect raw data, which is preprocessed to extract specific features. The processed data is then fed into detection mechanisms. The detection results are then aggregated and transmitted as alerts, which can be analysed further by an administrator or higher-level systems [17].

An extended IDS system was developed based on the architecture above [17], which is designed for operation in streaming environments. This system allows for model adaptation during runtime. The detection stages described were supplemented with semi-supervised learning mechanisms and a user (expert) interaction layer.

The system monitors the network data stream and determines whether each sample belongs to the normal or attack class. The general data flow in the system includes the following stages (Figure

4): packet capture; flow construction; preparation of flow features; class prediction by a machine learning model; prediction uncertainty assessment; a decision on whether to perform pseudo-labelling or consult an expert; and incremental model update.

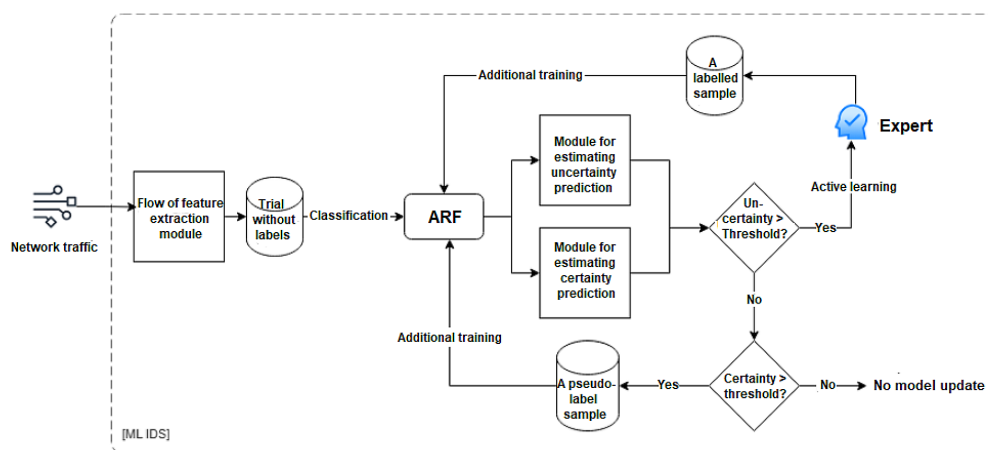
The system receives a stream of samples describing network traffic in the form of feature vectors as input. Each sample represents the features of a single flow. The data is processed sequentially without assuming access to the entire training set, corresponding to the operating conditions of actual IDS systems.

The requirement to construct network flows originates from the data format contained within the CIC datasets employed for training and evaluating the predictive model. Additionally, creating a bidirectional flow from raw traffic data reduces the amount of data while preserving the information needed for attack detection [32, 46].

The main component of the system is the ARF predictive model. For each incoming sample, the model generates a probability of belonging to an attack class, forming the basis for classification decisions and further uncertainty analyses.

The function $U(p) = 1 - 2|p - 0.5|$ was selected to estimate uncertainty. This function is symmetric about the point $p = 0.5$, so it reaches its highest value for predictions close to the decision boundary. This approach is consistent with classical measures of uncertainty used in active learning [47].

Based on the calculated uncertainty, the system decides how to use the sample in the learning process. If the uncertainty value exceeds the set threshold, $\theta U = 0.1$, the sample is sent to an expert to obtain the true label. This threshold was empirically selected as a compromise between


Figure 4. Data flow diagram in the proposed IDS system

the number of queries to the expert and the stability of the classification metrics.

For high-confidence predictions, a pseudo-labelling mechanism is used. Prediction confidence is determined based on a threshold derived from the distribution of confidence values from the last 500 samples. The pseudo-labelling threshold is set as the 90th percentile of the current distribution. The percentile was chosen to ensure that the model only trains on the most confident predictions. Using an adaptive threshold rather than a fixed value enables the model to adapt to the level of uncertainty in the data stream.

The change detection mechanisms built into the ARF algorithm enable it to respond to statistical changes in the data stream, but labels are required. To enable model adaptation, ARF is updated based on labelled samples, both expert- and pseudo-labelled.

System implementation in a research environment

The proposed solution was implemented as a prototype to test the algorithm's behaviour on historical data. Research experiments were conducted in the Google Colab environment. This environment was chosen due to the ease with which experiments can be reproduced, the availability of computational resources, and integration with popular machine learning libraries.

The experiments were conducted on a Google Compute Engine backend using a Python 3 interpreter and CPU-based computations. The environment parameters during the experiments were as follows:

- available RAM: approx. 12.7 GB,
- disk space: approx. 107 GB,
- no GPU acceleration.

The following libraries were used to implement the system:

- pandas (v2.3.3) – for reading, filtering, and preprocessing data,
- numpy (v2.0.2) – for calculations and statistical operations,
- matplotlib (v3.10.0) – for visualising experiment results,
- scikit-learn (v1.6.1) – for auxiliary tasks such as data splitting,
- river (v0.23.0) – a library for implementing machine learning in streaming mode and

including an ARF implementation, evaluation metrics, and drift detection mechanisms,

- pickle – for saving and loading the model state.

The River library was chosen due to its support for incremental learning and the ability to update the model sample by sample, which aligns with the design principles of a real-time IDS system.

Tests and discussion of results

A series of tests were conducted to evaluate the proposed solution. A set of scenarios was developed to simulate the system's various operating conditions and demonstrate its practical performance. The tests were designed to:

- assess the model's robustness to changes in data distribution,
- analyze the trade-off between prediction quality and the cost of obtaining labels.

Test 1: ARF with full label availability (reference scenario)

In this scenario, only the ARF model is tested. The model was trained incrementally using the CSE-CIC-IDS2018 data stream (100,000 samples of data as of February 16th).

The test was conducted under the assumption that a true label was available for each sample. This scenario is an ideal research setting that is frequently encountered in the literature [31].

It served as a baseline for evaluating the model's performance under fully supervised conditions.

Test 2: ARF with pseudo-labelling and no expert involvement

In this test, prepared by the author, the ARF model is trained incrementally on a CSE-CIC-IDS2018 data stream (100,000 samples from the February 16th dataset). The model is updated solely based on pseudo-labels generated by its own predictions. There are no queries to an expert and the model has no access to actual labels during the update.

The aim of the test was to assess the impact of erroneous pseudo-labels accumulating on the model's stability and effectiveness under the conditions of complete unsupervised learning.

Test 3: Proposed algorithm

This scenario involves an algorithm that was designed as part of this work and utilises the ARF model in streaming mode.

The model is updated based on pseudo-labels and expert labels obtained through selective queries. The queries sent to the expert are selected based on a measure of uncertainty, so the tests were conducted for various uncertainty thresholds. Data streams comprising 100,000 samples, collected on February 16th and 21st respectively, from the CSE-CIC-IDS2018 dataset were tested separately. The proposed scenario allows an analysis of the trade-off between the number of queries to the expert and the quality of predictions. The quality of detection of completely new attacks can also be evaluated.

Test 4: Scalability of the solution

In this scenario, the authors' proposed algorithm was tested on a dataset of 500,000 CSE-CIC-IDS2018 samples. The purpose of the test was to evaluate the stability of the algorithm's performance and how the metrics behave during long-term operation.

Test 5: Delayed expert Labels

This scenario involves testing the algorithm's performance under conditions similar to those in the real world. A delay in the availability of expert labels was introduced, whereby the true label becomes available for training 1,000 samples later [48]. The algorithm was tested on a stream of 500,000 samples from CSE-CIC-IDS2018. The objective of this test was to evaluate the stability of the solution when the model does not receive an immediate response from an expert.

Experimental results and analysis

The results were analysed by comparing metrics and confusion matrices, as well as the

supervision cost, which is expressed as the number of queries made to the expert.

Reference model – Test 1

The model trained under full supervision achieves stable, very high classification performance. According to Table 6, the accuracy rate exceeds 99.6% for both days, with F1 scores of 0.9921 and 0.9946 on February 16th and 21st, respectively. Analysis of the confusion matrix (Table 7) shows a low number of errors.

For the 16th February data, the number of false positives (FP) is 343 out of 75,298 negative-class samples, representing a low proportion of false positives. The number of false negatives (FN) is 46, which translates to a very high sensitivity of 0.9981. In contrast, the data from 21st February shows a higher proportion of false negatives (FN) at 331, resulting in a decrease in sensitivity to 0.9908. However, due to a reduction in FPs to 55, precision increases to 0.9985.

Impact of lack of supervision – Test 2

The results of Test 2 (Table 8) clearly indicate a degradation of the learning process under self-pseudo-labelling conditions. Accuracy drops to 75.64% and the F1 score for the attack class is zero.

The confusion matrix (Table 9) shows that the model incorrectly classifies all attack samples as negative (false negatives (FN) = 24,359, true positives (TP) = 0). This results in a loss of attack detection capability, which is why the false positive rate (FPR) is zero (true negatives (TN) = 75,641, false positives (FP) = 0). This is caused by the accumulation of erroneous pseudo-labels.

The impact of the uncertainty threshold and pseudo-labels on the proposed algorithm – Test 3

The results of Test 3 (Table 10) show that the uncertainty threshold affects the trade-off between supervision cost and prediction quality. For the 16 February data, a threshold of 0.1 yields an F1 score of 0.9913 and an AUC of 0.9965, which are very close to the reference model's scores of 0.9921 and 0.9968, respectively (see Table 6.2). The number of queries to the expert is 6,695, accounting for 6.7% of the stream samples.

Table 6. Quality metrics – full supervision

Metrics	February 16	Feb. 21
Accuracy	0.9961	0.9961
F1	0.9921	0.9946
Precision	0.9861	0.9985
Sensitivity	0.9981	0.9908
AUC	0.9968	0.9949

Table 7. Confusion matrix

Forecast class – 16-02		
Actual class	Negative (0)	Positive (1)
Negative (0)	75,298	343
Positive (1)	46	24,313
Forecast class – 21-02		
Actual class	Negative (0)	Positive (1)
Negative (0)	64,120	55
Positive (1)	331	35,494

Table 8. Confusion matrix

Metric	16.02
Accuracy	0.75641
F1	0.0
AUC	0.5
Proportion of incorrect pseudo-labels	0.24359

Table 9. Confusion matrix

Forecast class – 16-02		
Actual class	Negative (0)	Positive (1)
Negative (0)	75,641	0
Positive (1)	24,359	0

Analysis of the confusion matrix (Table 11) indicates a low error rate: the number of false positives is 378 and the number of false negatives is 50. This translates to high sensitivity (0.9979) and high precision (0.9847). This means that expert queries effectively limit the propagation of erroneous pseudo-labels.

However, increasing the threshold to 0.2 reduces the number of expert queries to 2,664 but results in a deterioration in detection quality. The number of false negatives rises to 3,124 (see Table 11), leading to a decrease in sensitivity to 0.8718 and an decrease in the area under the curve (AUC) to 0.9330. Pseudo-label error rises to 1.91%, indicating greater instability in the learning process.

For the data from February 21st, a threshold of 0.1 is inadequate: the number of expert queries

drops to 25 and the model deteriorates in a manner analogous to Test 3. Lowering the threshold to 0.05 stabilises the learning process. According to Table 10, the F1 score increases to 0.9977 and the AUC increases to 0.9979, while the number of false negatives drops to 125 (see Table 11).

Scalability of the solution – Test 4

The results of Test 4 (see Table 12) confirm that the algorithm is stable when operating on a stream of 500,000 samples over a prolonged period. With a threshold of 0.1, the F1 score is 0.9930 and the area under the curve (AUC) is 0.9975. The number of queries to the expert accounts for approximately 6% of the samples.

The confusion matrix (Table 13) shows a very low error rate: false negatives account for just 84 out of 122,625 attack samples, giving a sensitivity of 0.9993. The low pseudo-label error rate suggests that errors do not accumulate over time.

Increasing the threshold to 0.2 reduces the number of queries to 11,424, but also increases the number of false negatives to 3,161 (see Table 13). This results in a decrease in sensitivity to 0.9742 and in the AUC to 0.9850. These results confirm that the trade-off between monitoring cost and prediction quality is also evident in large-scale data settings.

Delayed expert labels – Test 5

The results of Test 5 (see Table 14) show that introducing a delay of 1.000 samples in the availability of expert labels does not affect the stability of the detection and adaptation processes. However, the quality of predictions deteriorates compared to the scenario without delays (Test 4). The uncertainty threshold for the test was set to 0.1. An accuracy of 0.9947 and an F1 score of 0.9893 were achieved, which is a decrease compared to Test 4 with a threshold of 0.1 (F1 = 0.9930; see Table 6.11).

Table 10. Quality metrics

Data / Threshold	Accuracy	F1	Precision	Sensitivity	AUC	Number of queries	Pseudo-label error
16.02/ 0.1	0.9957	0.9913	0.9847	0.9979	0.9965	6695	0.0006
16.02/ 0.2	0.9644	0.9227	0.9799	0.8718	0.9330	2664	0.0191
02/21/0.1	0.6417	0.0000	0.0000	0.0000	0.4999	25	0.3324
02/21/0.05	0.9983	0.9977	0.9989	0.9965	0.9979	3154	0.0015

Table 11. Confusion matrix

Forecast class – 16-02, threshold 0.1		
Actual class	Negative (0)	Positive (1)
Negative (0)	75,263	378
Positive (1)	50	24,309
Forecast class – 16-02, threshold 0.2		
Actual class	Negative (0)	Positive (1)
Negative (0)	75,207	434
Positive (1)	3124	21,235
Forecast class – 21-02, threshold 0.05		
Actual class	Negative (0)	Positive (1)
Negative (0)	64,134	41
Positive (1)	125	35,700

According to Table 15, the number of false positives increases from 1.633 in Test 4 to 2,275. The number of false negatives is 362, representing a significant increase compared to Test 4 (FN = 84). This results in a decrease in precision to 0.9817, though sensitivity remains similar at 0.9970.

Despite the label delay, the AUC is 0.9955, indicating that the model distinguishes between classes effectively. The deterioration in metrics is negligible. Overall, the results indicate that the proposed algorithm effectively handles the introduced delay in expert labels.

DISCUSSION

The experiments evaluated the effectiveness of the IDS system designed based on the Adaptive Random Forest algorithm, which incorporates an active learning mechanism, pseudo-labelling and selective expert involvement. There are limited possibilities for direct comparison with the literature. No results were found in the publicly available studies for an ARF model trained on one dataset and adapted to another in streaming mode. For this reason, the analysis presented here is qualitative in nature.

Test 1 shows that the model is highly resistant to concept drift when true labels are continuously available. However, it should be noted that the full availability of labels in research studies is treated as a simplifying assumption [48]. The real-world

Table 13. Confusion matrix

Forecast class – threshold 0.1		
Actual class	Negative (0)	Positive (1)
Negative (0)	375,742	1633
Positive (1)	84	122,541
Forecast class – threshold 0.2		
Actual class	Negative (0)	Positive (1)
Negative (0)	375,741	1634
Positive (1)	3161	119,464

cost of obtaining them is often overlooked. The results of this work show that such a scenario sets the upper limit of the adaptive model's effectiveness; however, its practical implementation in IDS systems is questionable. The results also highlight the issue of maximising metrics without considering real-world conditions [6].

The literature emphasised that the high effectiveness of models or entire detection algorithms evaluated on a single dataset does not translate into real-world effectiveness [6]. Cantone et al. [5] demonstrated that static models trained on a CICIDS-2017 subset and tested on the CSE-CIC-IDS-2018 dataset achieved F1 scores in the range 25–40%, despite achieving very good results on the CIC-IDS-2017 test set. This indicates that classical, statically trained models lose their ability to distinguish between normal activity and attacks when the environment changes. The authors emphasised that standard evaluation overestimates the actual quality of the solution.

In light of this research, it was assumed that the model must adapt to the data in practice. The results of Tests 3 and 4 demonstrate that querying an expert selectively enables high classification quality to be maintained at a limited labelling cost. An F1 score of 0.9930 was achieved with 6% of labels, and an F1 score of 0.9803 was achieved with approximately 2% (Test 4), suggesting that a limited number of expert labels allows for effective model adaptation.

Gupta et al. analysed the adaptation between CIC-IDS-2017 and CSE-CIC-IDS-2018 using various active learning strategies, including an uncertainty-based method.

Table 12. Quality metrics – 500,000 samples

Date / Threshold	Accuracy	F1	Precision	Sensitivity	AUC	Number of queries	Pseudo-label error
0.1	0.9966	0.9930	0.9869	0.9993	0.9975	30094	9.16e-05
0.2	0.9904	0.9803	0.9865	0.9742	0.9850	11424	0.00256

Table 14. Quality metrics

Accuracy	F1	Precision	Sensitivity	AUC	Number of queries	Pseudo-label error
0.9947	0.9893	0.9817	0.9970	0.9955	27,739	1.28e-04

Table 15. Confusion matrix

Actual class	Forecast class	
	Negative (0)	Positive (1)
Negative (0)	375,100	2275
Positive (1)	362	122,263

With a label budget of 1%, this method achieves an F1 score of 0.74. However, the comparison is indicative due to the lack of results for both solutions with the same budget. However, the F1 score of over 0.98 achieved in the conducted study using a larger, though still limited, proportion of labels (2.2%) indicates the high effectiveness of the proposed adaptation mechanism.

In order to deploy the developed system in high-speed network environments, it is necessary to minimise computational overhead. The proposed semi-supervised framework addresses real-time streaming constraints by making three architectural design choices. Applying Recursive Feature Elimination (RFE) reduces the feature space from 78 raw attributes to 14 core features, minimising operations per incoming flow and lowering processing latency. The semi-supervised active learning strategy only invokes model updates and expert queries when prediction uncertainty surpasses adaptive thresholds. This avoids the high computational cost of continuously updating decision trees on high-confidence samples. The underlying Adaptive Random Forest algorithm ensures that memory growth remains bounded during drift adaptation. The ADWIN mechanism initialises background trees after detecting distribution shifts and prunes stale tree nodes once drift stabilises [49]. Although hardware-level throughput and latency profiling depend heavily on the acceleration and deployment hardware of the Network Interface Card, these algorithmic design choices ensure that the online pipeline remains computationally lightweight for edge and forwarder integration.

In summary, the results suggest that the proposed solution is a viable alternative to static IDS models and more complex active learning methods. It is particularly significant that high

classification quality is achieved with limited expert involvement, even when adapting to a new dataset. This validates the solution under conditions closer to real-world scenarios. The conducted tests confirmed that a machine learning-based IDS system can be designed that operates online and is capable of adaptation without requiring full availability of labelled data. Using the model's most confident predictions for further training enables adaptation to changing network traffic characteristics. Active learning supports the stability of the learning process under dynamic conditions by obtaining labels for the most uncertain samples.

Despite the promising detection performance results, the proposed solution has certain limitations.

One of the main limitation is that the pseudo-labelling is based on the naive assumption that a model's confident prediction is correct. For example, when the characteristics of the data change suddenly or new types of attacks emerge, the model may generate incorrect predictions with high confidence, which will then be used for further training [4].

The results indicated the need to select an uncertainty threshold (tests 3 and 4). This threshold should be determined through experimentation based on an evaluation of the balance between labelling costs and prediction quality. The threshold must be adjusted according to the specific characteristics of the analysed environment and the availability of an expert.

The experiments were conducted on historical data, simulating a data stream, rather than in a real production environment. While the datasets used are commonly employed in IDS system research, real-world network traffic may be characterised by greater variability, as well as noise and incomplete data.

CONCLUSIONS

As part of this project, a machine learning-based intrusion detection system was developed and tested. The analysis focused on network security methods and the limitations of intelligent IDS

systems, particularly in the context of streaming data and limited label availability. A detection algorithm utilising incremental learning, active label acquisition and learning from its own predictions was then proposed. Based on this, a system architecture was designed. A prototype of the solution was implemented and experiments were conducted. The results obtained enabled an evaluation of the system's behaviour under the conditions close to real-world scenarios. The analysis confirmed that the proposed approach maintains detection effectiveness under changing conditions. The system can adapt to changes in network traffic and detect new attacks with minimal expert intervention.

Overall, the results demonstrate that the proposed approach is a viable alternative to static IDS models and more complex active learning methods. It achieves high detection performance with limited expert involvement, even when adapting to new datasets, indicating its suitability for realistic deployment scenarios. The experiments confirm also that an online machine learning-based IDS can adapt to changing network traffic without requiring fully labelled data by combining pseudo-labelling of high-confidence predictions with active learning for uncertain samples. However, the approach has several limitations. Pseudo-labelling assumes that highly confident predictions are correct, which may propagate errors when traffic characteristics change or new attacks appear. In addition, selecting an appropriate uncertainty threshold requires empirical tuning to balance labelling costs and prediction quality. Finally, the evaluation was conducted on simulated data streams from historical datasets, rather than in a real production environment, where network traffic is likely to be more variable, noisy, and incomplete.

REFERENCES

1. Murta E. et al. Real-Time Intrusion Detection via Machine Learning Approaches. Proceedings of the Ital-IA Intelligenza Artificiale – Thematic Workshops. Vol. 3762. CEUR Workshop Proceedings, 2024. Naples, Italy, <https://ceur-ws.org/Vol-3762/582.pdf>
2. Guerra J.L. et al. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security*. 2022. <https://doi.org/10.1016/j.cose.2022.102810>
3. Gupta R. et al. Generative active adaptation for drifting and imbalanced network intrusion detection. arXiv preprint arXiv:2503.03022, 2025 - arxiv.org
4. Alam M.T. et al. ADAPT: A pseudo-labeling approach to combat concept drift in malware detection. arXiv preprint arXiv:2507.08597, 2025 - arxiv.org
5. Cantoneet M. et al. On the cross-dataset generalization of machine learning for network intrusion detection. arXiv preprint arXiv:2402.10974, 2024 - arxiv.org
6. D'hooge L. et al. Investigating the generalized performance of data-constrained supervised machine learning models on novel, related samples in intrusion detection. *Sensors* 2023; 23(4): 1846. <https://doi.org/10.3390/s23041846>
7. Bassey C. et al. Building a scalable security operations center: a focus on open-source tools. *Journal of Engineering Research and Reports*. 2024; 26(7): 196–209. <https://doi.org/10.9734/jerr/2024/v26i71203>
8. Kukielka P., Kotulski Z. Systemy wykrywania intruzów wykorzystujące metody sztucznej inteligencji”. *Przegląd telekomunikacyjny*, 2011, <https://www.ippt.pan.pl/repository/open/o324.pdf>
9. Ashoor A.S., Gore S. Difference between Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). In: Wyld, D.C., Wozniak, M., Chaki, N., Meghanathan, N., Nagamalai, D. (eds) *Advances in Network Security and Applications*. CNSA 2011. Communications in Computer and Information Science, vol 196. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22540-6_48.
10. Lopez Velasquez J.M., Martínez Monterrubio S.M., Sanchez Crespo L.E. et al. Systematic review of SIEM technology: SIEM-SC birth. *Int. J. Inf. Secur.* 2023; 22: 691–711. <https://doi.org/10.1007/s10207-022-00657-9>
11. Kumar Jain Dr A. Comparative analysis of signature-based and anomaly-based IDS. *International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)*. 2025; 1(3). <https://doi.org/10.63345/ijarcse.v1.i3.204>
12. Abdulganiyu, O.H., Ait Tchakoucht, T., Saheed, Y.K. A systematic literature review for network intrusion detection system (IDS). *Int. J. Inf. Secur.* 2023; 22: 1125–1162. <https://doi.org/10.1007/s10207-023-00682-2>
13. IBM. What is an Intrusion Detection System (IDS)?. 2000, <https://www.ibm.com/think/topics/intrusion-detection-system/>
14. Sah S. Machine learning: A review of learning types, Preprints (2020). <https://doi.org/10.20944/preprints202007.0230.v1>
15. Yang S. et al. ReCDA: Concept Drift Adaptation with Representation Enhancement for Network Intrusion Detection. *KDD '24: Proceedings of the*

- 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024; 3818–3828. <https://doi.org/10.1145/3637528.36720>
16. Shyaa M.A. et al. Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems. *Engineering Applications of Artificial Intelligence*, 2024; 137: 109143–109143. <https://doi.org/10.1016/j.engappai.2024.109143>
17. Hozouri A., Mirzaei A., Effatparvar M. A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning, and emerging cybersecurity challenges. *Discover Artificial Intelligence*. 2025, <https://doi.org/10.1007/s44163-025-00578-1>
18. Mane D. Machine learning algorithms for intrusion detection in cybersecurity. *International Journal for Research in Applied Science and Engineering Technology* 2023; 11(5): 6315–6321. <https://doi.org/10.22214/ijraset.2023.52788>
19. Choudhary S., Kesswani N. Analysis of KDD-Cup'99, NSL-KDD, and UNSW-NB15 datasets using deep learning in IoT. *Procedia Computer Science* 2020; 167: 1561–1573. <https://doi.org/10.1016/j.procs.2020.03.367>
20. Dymora P. et al. Comparative analysis of machine learning models for multiclass anomaly detection in internet of things network traffic using the RT-IoT2022dataset. *Advances in Science and Technology Research Journal*. 2026; 20(4): 45–59. <https://doi.org/10.12913/22998624/213807>
21. Kuraś P., Bolanowski M., Łoza M. Optimization of machine learning models for effective anomaly detection in industrial IoT systems. *Advances in Science and Technology Research Journal*. 2026; 20(1): 203–221. <https://doi.org/10.12913/22998624/210686>
22. Moustafa N., and Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS)*, 2015. IEEE, 2015.
23. Ahmed U. et al. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering. *Scientific Reports*. 2025, <https://doi.org/10.1038/s41598-025-85866-7>
24. University of New Brunswick. Intrusion detection evaluation dataset (CIC-IDS-2017). <https://www.unb.ca/cic/datasets/ids-2017.html>
25. Panigrahi R., Borah S. A detailed analysis of the CICIDS-2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology* 2018; 7(3.24): 479–482. <https://doi.org/10.15664/jjt.v97i17.2039/ResearchGate> ID: 329045441
26. Leevy J.L., Khoshgoftaar T.M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*. 2020. <https://doi.org/10.1186/s40537-020-00382-x>
27. Adewole K.S. et al. Empirical analysis of data streaming and batch learning models for network intrusion detection. *Electronics* 2022; 11(19): 3109; <https://doi.org/10.3390/electronics11193109>
28. Pelosi D. et al. Explainability and interpretability in concept and data drift: A systematic literature review. *Algorithms* 2025; 18(7): 443; <https://doi.org/10.3390/a18070443>
29. Jacob S.L., Habibullah P.S. A systematic analysis and review on intrusion detection systems using machine learning and deep learning algorithms. *Journal of Computational and Cognitive Engineering*. 2024. <https://doi.org/10.47852/bonview/JCCE42023249>
30. Maseer Z.K. et al. Benchmarking of machine learning for anomaly-based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access*. 2021. <https://doi.org/10.1109/ACCESS.2021.3056614>
31. Miani R. et al. A survey of data stream-based intrusion detection systems. *IEEE Access*. 2025. <https://doi.org/10.1109/ACCESS.2025.3561105>
32. Hajizadeh M., Barua S., Golchin P. FSA-IDS: A Flow-based self-active intrusion detection system. *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. <https://doi.org/10.1109/NOMS56928.2023.10154343>
33. Ajjaj S. et al. Incremental online machine learning for detecting malicious nodes in vehicular communications using real-time monitoring. *Telecom*. 2023. <https://doi.org/10.3390/telecom4030028>
34. Laskov P., Düssel P., Schäfer C., Rieck K. Learning Intrusion Detection: Supervised or Unsupervised?. In: Roli, F., Vitulano, S. (eds) *Image Analysis and Processing—ICIAP2005*. ICIAP2005. *Lecture Notes in Computer Science*, 2005; 3617. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11553595_6
35. Angelo P., Drummond A.C. A survey of random forest-based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)*, 2018; 51(3), 48: 1–36. <https://doi.org/10.1145/3178582>
36. Thakkar A., Lohiya R. A review on challenges and future research directions for machine learning-based intrusion detection systems. *Archives of Computational Methods in Engineering*. 2023. <https://doi.org/10.1007/s11831-023-09943-8>
37. Alqahtani A.H. An incremental hybrid adaptive network-based IDS in Software Defined Networks to detect stealth attacks. *arXiv preprint arXiv:2404.01109*. 2024 - arxiv.org
38. Talpini J., Sartori F., Savi M. Enhancing trustworthiness in ML-based network intrusion detection with uncertainty quantification. *Journal of*

- Reliable Intelligent Environments. 2024. <https://doi.org/10.1007/s40860-024-00238-8>
39. Silva J.V.V. et al. A statistical analysis of intrinsic bias in network security datasets for training machine learning mechanisms. *Annals of Telecommunications*. 2022; <https://doi.org/10.1007/s12243-021-00904-5>
40. D'hooge L. et al. Establishing the Contaminating Effect of Metadata Feature Inclusion in Machine-Learned Network Intrusion Detection Models. *Detection of Intrusions and Malware, and Vulnerability Assessment*. Edited by Lorenzo Cavallaro. Springer, Cham: Springer International Publishing, 2022. <https://doi.org/10.1007/978-3-031-09484-2>
41. Bacevicius M. et al. Comparative analysis of perturbation techniques in LIME for intrusion detection enhancement. *Machine Learning and Knowledge Extraction*. 2025. <https://doi.org/10.3390/make7010021>
42. Abdulboriy A., Shin J.S. An Incremental majority voting approach for intrusion detection system based on machine learning. *IEEE Access*. (2024). <https://doi.org/10.1109/ACCESS.2024.3361041>
43. Howard C., Jananthan H., Kepner J. From Bits to Insights: Exploring Network Traffic, Traffic Matrices, and Heavy-Tailed Data". *IEEE MIT Undergraduate Research Technology Conference (URTC)*. 2023; <https://doi.org/10.1109/URTC60662.2023.10535002>
44. Habibi A. et al. Characterization of Tor Traffic Using Time Based Features. *Proceeding of the 3rd International Conference on Information System Security and Privacy*, SCITEPRESS, Porto, Portugal, 2017. <https://doi.org/10.5220/0006105602530262>
45. University of New Brunswick. CSE-CIC-IDS2018 on AWS. <https://www.unb.ca/cic/datasets/ids-2018.html>
46. Rodríguez M. et al. Evaluation of machine learning techniques for traffic flow-based intrusion detection. *Sensors*. 2022. <https://doi.org/10.3390/s22239326>
47. Nguyen V.-L., Shaker M.H., Hüllermeier E. How to measure uncertainty in uncertainty sampling for active learning. *Machine Learning*, 2022. <https://doi.org/10.1007/s10994-021-06003-9>
48. Gomes H.M. et al. Adaptive random forests for evolving data stream classification. *Machine Learning*. 2017. <https://doi.org/10.1007/s10994-017-5642-8>
49. Bifet A., Gavaldà R. (2007). Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*. 2007; 7: 443–448. <https://doi.org/10.1137/1.9781611972771.42>