

Article

Neural Network Method for Detecting UDP Flood Attacks in Critical Infrastructure Microgrid Protection Systems with Law Enforcement Agencies' Rapid Response

Serhii Vladov ^{1,2,*} , Łukasz Ścisło ^{3,*} , Anatoliy Sachenko ^{4,5} , Jan Krupiński ³ , Victoria Vysotska ^{2,6} , Maksym Korniienko ⁷, Oleh Uhrovetskyi ⁸, Vyacheslav Krykun ⁹, Kateryna Levchenko ¹⁰ and Alina Sachenko ¹¹

- ¹ Department of Scientific Activity Organisation, Kharkiv National University of Internal Affairs, 27, L. Landau Avenue, 61080 Kharkiv, Ukraine
- ² Department of Combating Cybercrime, Kharkiv National University of Internal Affairs, 27, L. Landau Avenue, 61080 Kharkiv, Ukraine; victoria.a.vysotska@lpnu.ua
- ³ Department of Automation and Computer Engineering, Cracow University of Technology, 24, Warszawska, 31-155 Cracow, Poland; jan.krupinski@pk.edu.pl
- ⁴ Research Institute for Intelligent Computer Systems, West Ukrainian National University, 11, Lvivska Street, 46009 Ternopil, Ukraine; as@wunu.edu.ua
- ⁵ Department of Teleinformatics, Casimir Pulaski Radom University, 29, Malczewskiego Street, 26-600 Radom, Poland
- ⁶ Information Systems and Networks Department, Lviv Polytechnic National University, 12, Bandera Street, 79013 Lviv, Ukraine
- ⁷ Odesa State University of Internal Affairs, 1 Uspenska Street, 65014 Odesa, Ukraine; oduvs2015@ukr.net
- ⁸ Laboratory of Theoretical Studies, International, Editorial-Publishing, and Scientific-Methodological Activities, National Scientific Center «Hon. Prof. M.S. Bokarius Forensic Science Institute», 8-A, Zolochivska Street, 61177 Kharkiv, Ukraine; oleg0023@i.ua
- ⁹ Department of Civil Law Disciplines, Institute of Law and Security, Odesa State University of Internal Affairs, 1 Uspenska Street, 65014 Odesa, Ukraine; krykunviacheslav@gmail.com
- ¹⁰ Ministry of Social Policy, Family and Unity of Ukraine, Government Commissioner for Gender Policy, 21/8, Instytutaska Street, 01008 Kyiv, Ukraine; levchenkokateryna@yahoo.com
- ¹¹ Owl Land of Illinois Inc., 2130 W Corporate Drive, Addison, IL 60101, USA; alinasachenko0@gmail.com
- * Correspondence: serhii.vladov@univd.edu.ua (S.V.); lscislo@pk.edu.pl (Ł.Ś.)

Abstract

This article develops a hybrid neural network method for detecting UDP flooding in critical infrastructure microgrid protection systems. This method combines sequential statistics (CUSUM) and a multimodal convolutional 1D-CNN architecture with a composite scoring criterion. Input features are generated using packet-aggregated one-minute vectors with metrics for packet count, average size, source entropy, and HHI concentration index, as well as compact sketches of top sources. To ensure forensically relevant incident recording, a greedy artefact selection policy based on the knapsack problem with a limited forensic buffer is implemented. The developed method is theoretically justified using a likelihood ratio criterion and adaptive threshold tuning, which ensures control over the false alarm probability. Experimental validation on traffic datasets demonstrated high efficiency, with an overall accuracy of 98.7%, a sensitivity of 97.4%, an average model inference time of 5.3 ms (2.5 times faster than its LSTM counterpart), a controlled FPR of 0.96%, and a reduction in asymptotic detection latency with an increase in intensity from 35 to 12 s. Moreover, with a storage budget of 10 MB, 28 priority bins were selected (their total size was 7.39 MB), ensuring the approximate preservation of 85% of the most informative packets for subsequent examination. This research contribution involves the creation of a ready-to-deploy, resource-efficient detector with low latency, explainable statistical layers, and a built-in mechanism for generating a standardized evidence package to facilitate rapid law enforcement response.



Academic Editors: Sen Tan,
José Matas and Jorge El Mariachet

Received: 27 November 2025

Revised: 23 December 2025

Accepted: 27 December 2025

Published: 30 December 2025

Copyright: © 2025 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: critical infrastructure microgrid protection systems; cybersecurity; UDP flood attacks; neural network; CUSUM; knapsack algorithm; hybrid detector; law enforcement agencies

1. Introduction

Today, UDP flooding remains one of the simplest yet most effective forms of distributed denial of service (DDoS), particularly dangerous for critical infrastructure systems (energy, water supply, transportation, and healthcare) [1]. It is due to the fact that essential infrastructure facilities often contain highly specialized equipment and legacy components, which are critical to high availability and traffic predictability. Even a short-term disruption to communications or a delay in control can lead to consequences that extend beyond a digital incident and affect the safety of people and processes [2]. Moreover, attacks on the network flow layer, which predominantly use UDP packets, are more difficult to classify as malicious without the aid of contextual analytics. This is because UDP is a connection-less protocol that can be easily forged and generated from a large number of distributed sources [3]. Figure 1 shows an example of a UDP traffic time series with characteristic bursts during flood attacks. It clearly illustrates these threats and the need for prompt monitoring and preservation of artefacts for forensic analysis.

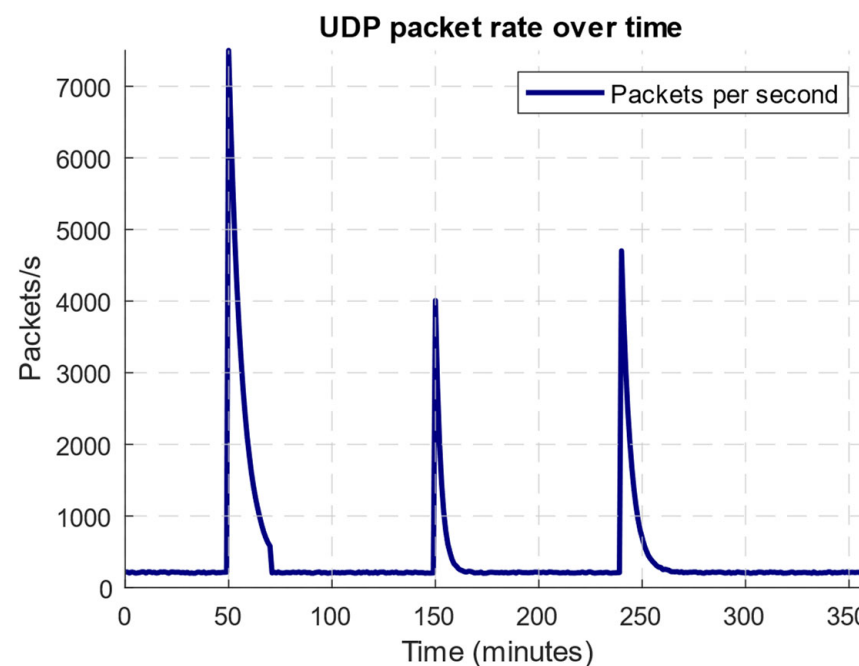


Figure 1. Time series with three attack periods.

Figure 1 illustrates a combination of steady background traffic and several short-term, intense bursts, which simple thresholding methods can easily detect. In contrast, low-intensity, prolonged increases in traffic almost blend into the background and require more sophisticated statistical and behavioral analytics for reliable detection.

Figure 2 illustrates the distinguishing characteristic between regular and attack traffic. The mode shift and concentration of small (or uniform) attack sizes facilitate feature engineering for classifiers and fast filters. These distinctive packet size profiles reduce the number of false positives in high-volume IoT traffic while simultaneously generating evidentiary artefacts (statistical signatures, histograms, and anomalous sample hashes) for reporting to law enforcement.

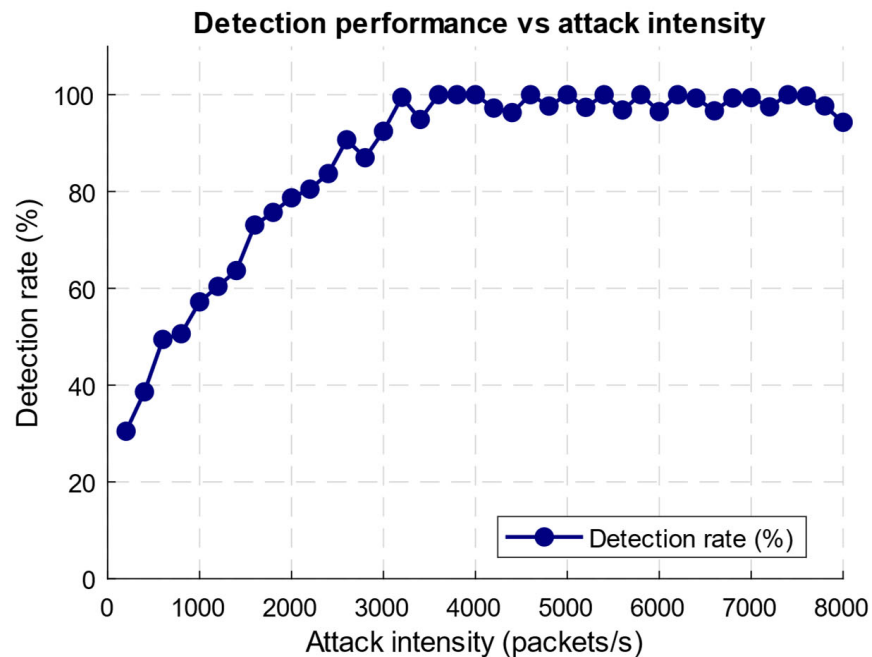


Figure 2. Histogram of packet size distribution in regular traffic and under attack.

Figure 3 shows that the number of successfully detected attacks increases significantly in proportion to the growing intensity. That is, at very low intensities, detection remains low and noisy, confirming the system's vulnerability to "silent" long-term floods. It highlights the need for a hybrid approach, combining the temporal patterns from Figure 1 and the packet size features from Figure 2, along with adaptive thresholds and source correlation, to detect low-intensity attacks while reducing false positives and preserving the evidence base for law enforcement.

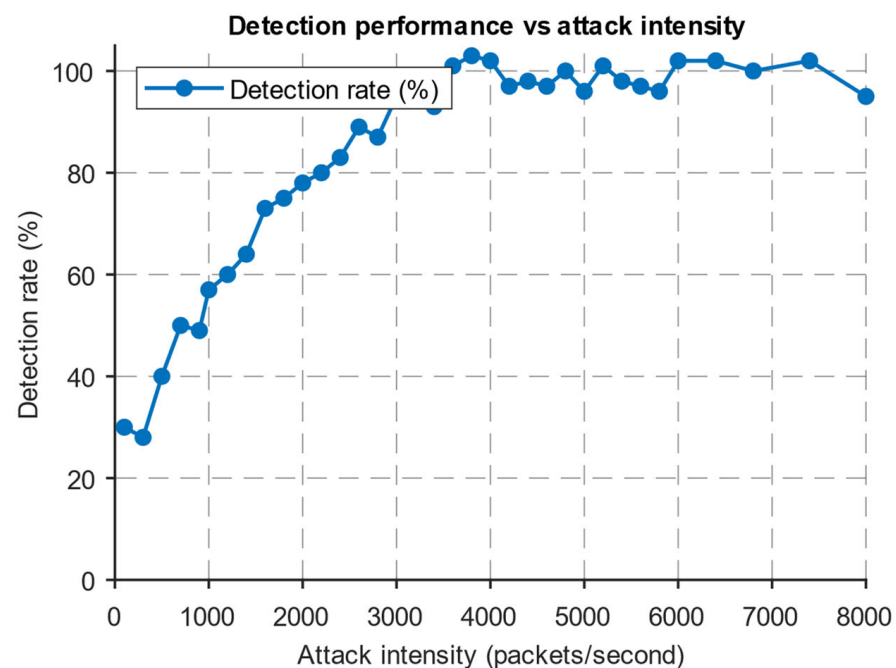


Figure 3. Detection efficiency diagram as a function of attack intensity.

Figure 4 illustrates a classic UDP flood scenario, where multiple clients (or fake sources) forward UDP packets through a router to a server, resulting in the server being overloaded with requests and unable to support legitimate traffic.

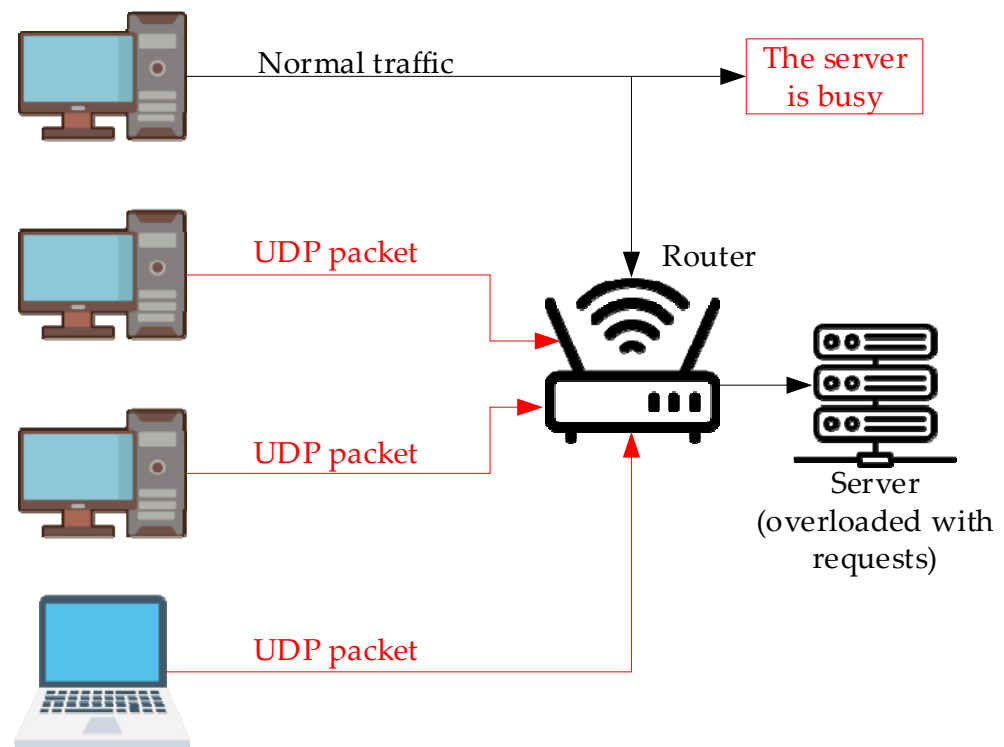


Figure 4. UDP flood attack implementation diagram.

Figure 4 is the core of the above. While Figure 1 illustrates the actual time profile of traffic spikes, Figure 2 demonstrates the structural cause of such spikes and explains how distributed or spoofed sources create a denial-of-service effect. For critical infrastructure cybersecurity systems, this means that even low-power but synchronized flows from multiple nodes can cause equipment failure or disrupt control processes. Therefore, a detector must be able to not only detect anomalous spikes (as in Figure 1) but also correlate events by source, AS path, and packet size profile (Figures 2 and 3), as well as promptly preserve evidence and generate a materials package for transfer to law enforcement agencies.

Thus, Figures 1–4 clearly illustrate the need to combine rapid monitoring with in-depth forensics and source correlation before handing over data to law enforcement. In this context, this topic’s relevance is underscored by two interrelated challenges:

- The changing nature of traffic and the growth of IoT devices increase background network noise, making it challenging to detect low-intensity but long-lasting UDP floods.
- The necessary integration of the detection mechanism with law enforcement agencies’ operational responses and infrastructure operators’ systems places special demands on the accuracy, explainability, and response speed of detection systems.

Systems tuned for sensitivity to sudden bursts often generate false positives in real-world conditions, while conservative thresholds allow low-intensity attacks to “slip through” [4,5]. Therefore, research focuses not only on improving detection but also on reducing false positives and on attribution mechanisms (forensics) for transmitting data to law enforcement agencies [6]. Within this framework, the development of methods capable of identifying characteristic statistical and UDP flooding behavioral markers (temporal patterns, packet size distribution, source spectra, etc.), as well as the creation of hybrid algorithms combining a fast network filter and in-depth flow analysis for incident confirmation, is a pressing scientific and practical task. Furthermore, a standardized protocol for notification and transmission of artefacts (logs, session dumps, suspicious sample hashes)

to law enforcement agencies is an essential element, which requires a balance between efficiency, confidentiality, and the evidentiary value of the obtained data.

Thus, this study makes the following contribution to the subject area:

1. A hybrid network traffic anomaly detection method has been developed that combines sequential statistical tests (CUSUM, GLR) with deep temporal convolutional models and feature metrics (entropy, HHI), thereby increasing sensitivity to a variety of UDP flood scenarios while maintaining robustness to background fluctuations.
2. A compact flow representation scheme (Count-Min Sketch with top-k) is proposed in combination with a resource-efficient neural network model architecture, which provides scalable real-time flow processing and is suitable for deployment on edge platforms with limited memory and latency.
3. A formalized protocol for ensuring data integrity and admissibility (chain of custody, cryptographic hashes, artefact versioning, and access control in accordance with ISO/IEC 27037; Information technology – Security techniques – Guidelines for the identification, collection, acquisition, and preservation of digital evidence. ISO: Geneva, Switzerland, 2012.) has been introduced, which ensures the reproducibility of the experiments and the legal significance of the results obtained.
4. A practical mechanism for adapting the model to data drift and evolving attacks (automatic drift detection, active learning for labelling, incremental fine-tuning, and canary releases) is described, which improves the long-term reliability and maintainability of the proposed approach.

2. Existing Research Review

Classic signature-based systems (Snort, Suricata, etc.) [7,8] remain widely used for detecting known UDP flood patterns, as they are effective at recognizing known patterns and are easy to deploy. However, it has been noted that such systems perform poorly against new or modified floods, fail to detect low-intensity and polymorphic attacks, and generate a large number of false positives in environments with an unstable background (IoT networks).

Statistical methods [9,10] and threshold detectors (entropy-based [11,12], rate-threshold [13,14], and moving average [15]) utilize the behavioral characteristics of traffic and are easily understood by operators. However, in these methods, the threshold choice is sensitive to network variability. In general, these methods are either too sensitive (resulting in many false positives) or too conservative (missing “silent” attacks), and they also do not scale well for high-speed flows without hardware acceleration.

Flow-based machine learning (NetFlow and IPFIX classifiers, including Random Forest [16], SVM [17], and XGBoost [18]) demonstrates high performance in the presence of labels and an informative feature set (packet number, session duration, and size distribution). However, applying these methods in real-world conditions requires high-quality labelling; the models are sensitive to distribution drift, and their real-time implementation requires optimization for latency and memory, which is problematic in resource-constrained CI environments.

Deep learning-based approaches (CNN [19,20], RNN [21,22], LSTM [23–25], attention [26,27]) are capable of automatically extracting complex temporal and spatial dependencies in network traffic and have demonstrated improved detection of low-intensity attacks in studies. However, deep learning-based methods have significant drawbacks, including the need for large labelled datasets, computational costs, difficulty interpreting findings, and the risk of overfitting on synthetic datasets.

Autoencoders [28,29] and unsupervised training methods (anomaly detection via autoencoders [30,31] and isolation forests [32,33]) are helpful in the absence of labels, as

such models learn the regular traffic patterns and flag abnormalities. The limitations of these methods include the difficulty of controlling alarm thresholds, the high likelihood of false positives due to seasonal traffic fluctuations, and the need for periodic retraining to account for changes.

Stream [34] and sketch algorithms (Count-Min Sketch [35], Bloom filters [36], and lossy counting [37]) are used for cost-effective tracking of high-speed streams and detecting “heavy hitters.” However, these methods lose detailed information (approximate counts), poorly detect distributed low-intensity attacks, and do not produce explainable artefacts for forensics.

SDN-oriented solutions [38–40] offer dynamic traffic shaping (or redirection) in the control plane, simplifying policy implementation and rapid response. However, they presuppose an SDN infrastructure and trust in the existence of the controller. Furthermore, critical infrastructure often utilizes irreplaceable, proprietary devices, where SDN intervention is either impossible or risky.

Hybrid architectures (rule-based with machine learning methods) [41–43] combine the detection speed at the simple rule level with the flexibility of machine-learning-based classifiers for incident confirmation. The hybrid architectures’ main disadvantages are the orchestration complexity, the rules’ risk and model desynchronization, and increased operational costs for maintenance (rule updates, model retraining).

Methods of collective defense and information exchange between providers and ISPs (blackholing [44], source filtering [45], and collaborative BGP-based mitigation [46]) are effective in combating large-scale flood attacks. However, for CI purposes, coordination delays, legal and privacy barriers to data transfer, and the legitimate sources being blocked due to inaccurate decisions are significant risks.

Forensic methods [47,48] and attribution techniques (IP traceback [49], correlation [50], and flow correlation across sensors [51,52]) are crucial for disseminating evidence to law enforcement. Challenges include UDP spoofing, NAT, and proxies, which complicate accurate attribution, as well as evidence storage and integrity, which require secure channels and standards that are not always considered in early research.

CI-focused studies, such as [53–55], emphasize the environment specifics, namely, limited computing resources, certification requirements, long hardware lifecycles, and the high cost of downtime. However, the methods developed in [53–55] were tested on public synthetic datasets (CAIDA, CIC), which poorly reflect real-world CI workloads, so their portability to real-world systems remains challenging.

Research on explainable AI (XAI) [56] and the legal admissibility of detections [57] is still in its infancy. Even if a model detects an anomaly, operators and investigators require clear explanations and an evidence chain. Most modern neural network solutions do not provide guaranteed interpretability and auditability, which is critical when reporting incidents to law enforcement.

Based on the above, Figure 5 illustrates the comparative suitability of various approaches for detecting UDP flooding in critical infrastructure environments, and Table 1 provides a summary of the methods considered, along with their main shortcomings.

Thus, existing approaches demonstrate that there is no universal solution that simultaneously satisfies the requirements for early detection of low-intensity UDP floods, limited CI resources, explainability for operators, and evidentiary requirements for reporting to law enforcement. Open challenges include the reliable detection of “silent” attacks amid noisy IoT traffic, modeling robustness to distribution drift and attack polymorphism, minimizing latency and computational load in real-time, and ensuring interpretable, auditable conclusions and secure storage of forensic artefacts. Therefore, it is necessary to develop a neural network method optimized for the CI specifics (compact architectures, online

(incremental) training, explainability mechanisms, and a built-in evidence chain), as well as an integrated protocol for interaction with law enforcement agencies for prompt and legally correct incident response.

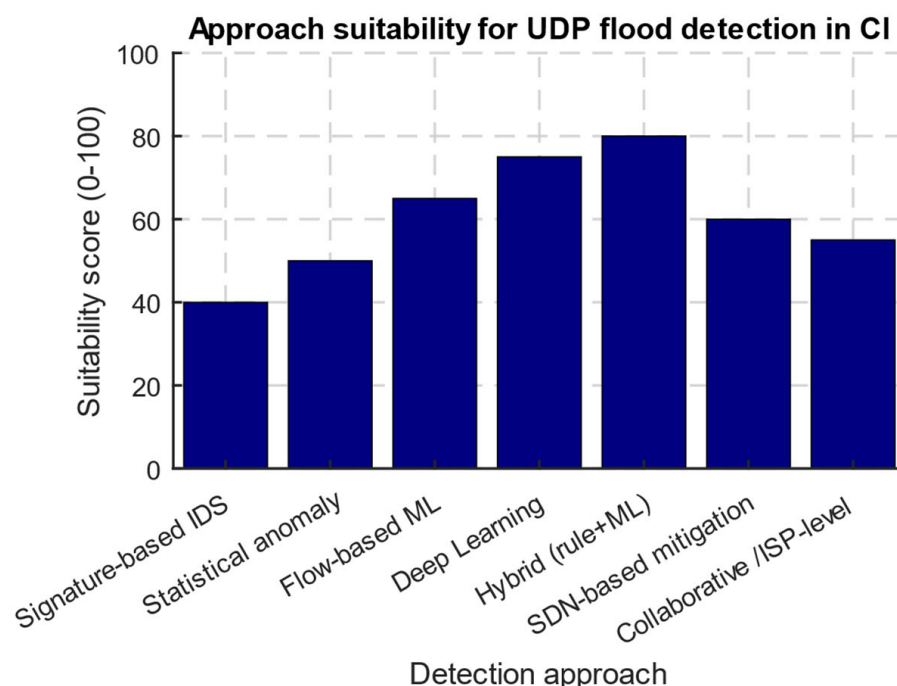


Figure 5. Comparative suitability of approaches for detecting UDP flooding in critical infrastructure environments.

Table 1. Summary of the approaches considered for detecting UDP floods and their main drawbacks (results based on analysis).

Number	Approach	Limitations
1	Signature-based IDS	Does not detect new polymorphic floods, has high sensitivity to obfuscation, and has false positives in dynamic backgrounds.
2	Statistical and threshold methods	Sensitive to threshold selection, it is not suitable for low-intensity attacks and requires customization for a specific network.
3	Flow-based machine learning (RandomForest, XGBoost)	It requires tags, is subject to drift, and has resource requirements for real-time.
4	Deep learning (CNN, RNN, LSTM)	Big data and resources, weak explainability, risk of overfitting on synthetic datasets.
5	Autoencoders unsupervised	Ambiguous thresholds, increased false positives when profile changes, and periodic retraining.
6	Sketching and streaming algorithms	Approximate data, poor detection of distributed low-intensity attacks, and few forensic artefacts.
7	SDN-based mitigation	Requires SDN infrastructure, risks network stability, and is incompatible with legacy CI equipment.
8	Hybrid approach (machine learning with rules)	Complexity of integration, operational load, and risk of desynchronization.
9	Collaborative (ISP-level)	Legal and privacy barriers, coordination delays, and the risk of blocking legitimate services.

Table 1. Cont.

Number	Approach	Limitations
10	Forensics traceback	Spoofing, NATs interfere with accurate attribution, storage requirements and evidence integrity.
11	Explainable AI for legal readiness	Insufficient interpretability and auditability, unpreparedness for legal use of results.

3. Materials and Methods

3.1. Development of a Theoretical Framework for Detecting UDP Flooding for the Security of Critical Infrastructure Cybersecurity Systems

It is assumed that the observed flow of UDP packets in the form of a packet counter $N(t)$ is the number of packets arriving in the interval $[0, t]$. As part of the theoretical model development for detecting UDP flooding, observations in discrete equal windows of length $\Delta > 0$ are considered; that is, for $k \in N$, it is assumed that the number of packets in the k -th time bins X_k corresponds to the expression:

$$X_k := N(k \cdot \Delta) - N((k - 1) \cdot \Delta), \quad (1)$$

where N is the incoming packets' counter during the time interval $[0, t]$ and Δ is the discrete windows' (bin) length in the same units of time ($\Delta > 0$).

In this case, for each i -th packet in the bin, a fixed feature set is defined, consisting of the size $S_i \in \mathbb{R}_+$ (in bytes), source A_i (IP, ASN), and arrival time t_i . For the k -th bin, a feature vector is determined (empirical size distribution, source distribution, etc.). It is assumed that $p_0(\bullet)$ denotes the "normal" (background) traffic model, and $p_1(\bullet)$ denotes the model in the UDP flood presence.

To construct a stochastic traffic model, a basic background traffic model is considered as a Poisson process with intensity $\lambda_0(t)$ (in the simplest case, a stationary λ_0). To account for seasonality and slow drift, it is assumed that

$$\lambda_0(t) = \bar{\lambda}_0 + \eta(t), \quad |\eta(t)| \ll \bar{\lambda}_0, \quad (2)$$

where $\eta(t)$ is a slowly changing deterministic (or weakly stochastic) deviation of the background intensity, $\lambda_0(t)$ is the background (base) flow intensity (in a simple case, constant λ_0). The attack is modelled as an additional process with intensity $\lambda_A(t) \geq 0$, so that the total intensity:

$$\lambda(t) = \lambda_0(t) + \lambda_A(t), \quad (3)$$

where $\lambda_A(t)$ is the additional intensity created by the attack (a non-negative function of time).

It is noted that typical λ_A models are:

- Pulse explosions:

$$\lambda_A(t) = \sum_j a_j \cdot \delta(t - \tau_j), \quad (4)$$

- Long "quiet" floods (smooth function):

$$\lambda_A(t) = \sum_j A_j \cdot h(t - \tau_j), \quad h(u) = \mathbf{1}_{u \geq 0} \exp(-\alpha \cdot u), \quad (5)$$

- Self-exciting Hawkes process:

$$\lambda(t) = \mu(t) + \int_0^t g(t-s) dN(s), \quad (6)$$

where g is the kernel (e.g., $g(u) = \beta \cdot \exp(-\alpha \cdot u)$), β is the excitation functions' (impulse contribution) scale (amplitude), and α is the velocity attenuation coefficient of the kernel (in the example of an exponential kernel). The Hawkes model describes cascading effects and synchronization of distributed sources.

For analytics, based on [56], it makes sense to move on to intensity in bins:

$$\Lambda_k := \int_{(k-1) \cdot \Delta}^{k \cdot \Delta} \lambda(t) dt = \Lambda_{0,k} + \Lambda_{A,k}, \quad (7)$$

then, under the Poisson assumption, $X_k \sim \text{Poisson}(\Lambda_k)$.

Temporal (count-based) detection is based on the classical test, that is, comparing X_k with an adaptive threshold T_k for a fixed window Δ . For this, a normalized statistic of the form is introduced:

$$T_k^{(c)} := \frac{X_k - \hat{\Lambda}_{0,k}}{\sqrt{\hat{\Lambda}_{0,k}}}, \quad (8)$$

where $\hat{\Lambda}_{0,k}$ is the estimate of background intensity in the k -th bin (can be a sliding window, exponential smoothing, or a model). Under the null hypothesis H_0 : $\Lambda_{A,k} = 0$, we have approximately $T_k^{(c)} \xrightarrow{d} \mathcal{N}(0, 1)$ for large $\hat{\Lambda}_{0,k}$, which corresponds to the central limit theorem [57]. To achieve this, a normalized statistic is introduced, that is:

$$\mathbb{P}_{H_0}(T_k^{(c)} > u) \approx 1 - \Phi(u) = \alpha, \quad (9)$$

where u represents the threshold for the normalized statistic, selected to match the acceptable false positive rate.

However, the proposed approaches' application is limited to low-intensity, long-term floods, so a multivariate statistical combination is required.

In this context, to determine packet size statistics, it is assumed that the k -th bin contains a set of sizes $\{S_{k,1}, \dots, S_k, X_k\}$. It is also believed that $F_0(s)$ is a known (or a priori estimated) cumulative distribution function of the background size, and $f_0(s)$ is the density. Consider the empirical distribution— $\hat{F}_k(s)$, and use the Kullback–Leibler (KL) distance [58]:

$$D_{KL}(\hat{f}_k \| f_0) = \int \hat{f}_k(s) \cdot \log \frac{\hat{f}_k(s)}{f_0(s)} ds, \quad (10)$$

When discretizing the sizes into m bins, the χ^2 and DKL statistics are defined as:

$$\chi_k^2 = \sum_{i=1}^m \frac{(n_{k,i} - X_k \cdot p_{0,i})^2}{X_k \cdot p_{0,i}}, \quad D_k = \sum_{i=1}^m \hat{p}_{k,i} \cdot \log \frac{\hat{p}_{k,i}}{p_{0,i}}, \quad (11)$$

where p_j is the background proportion (probability) in the j -th size basket, and q_j is the empirical proportion in the j -th basket.

It is noted that large values of χ_k^2 or χ_k^2 indicate a change in the packet size profile (for example, concentration on small sizes during UDP flooding).

For the correlation by sources (forensic score), it is assumed that Q_k is the empirical distribution of sources (IP, ASN) in the k -th bin. Information entropy is defined as:

$$H(Q_k) = - \sum_a q_{k,a} \cdot \log(q_{k,a}). \quad (12)$$

It is noted that distributed flooding often involves an increase in the number of unique sources (fake sources) or, conversely, a concentration on a small set of bot nodes. Based on [59], introducing a concentration index (Herfindahl–Hirschman) of the form:

$$HHI_k = \sum_a q_{k,a}^2, \quad (13)$$

To assess synchronicity across sources. For forensics, the “attribution-weight” metric is important, defining the correlation statistics between bins as:

$$\rho_{k,\ell} = \text{corr}(\mathbf{1}_{A_i \in \mathcal{S}} \text{ in bins } k, \ell). \quad (14)$$

It is known that [60] the “attribution-weight” metric is vital for forensics:

$$F_k := \sum_a \omega_a \cdot q_{k,a}, \quad \omega_a = \phi(\text{ASN}(a), \text{reputation}(a)), \quad (15)$$

where ϕ is the function to convert ASN (or reputation) to weight when calculating “attribution-weight”, for example, converts ASN to weight in the range [0, 1].

Based on the above, the combined detector function consists of combining three components (counts, sizes, sources) into a single score, that is:

$$S_k := \alpha_1 \cdot g_c(T_k^{(c)}) + \alpha_2 \cdot g_s(D_s) + \alpha_3 \cdot g_f(F_k), \quad (16)$$

where $g_\bullet$ are monotonic normalization transformations (e.g., sigmoid or standardization) and $\alpha_i \geq 0$ are weights selected based on the optimization criterion. The detection rule is determined by the presence of an alarm signal in the k -th bin if $S_k > h$, where h is the adaptive threshold. It is noted that threshold adaptation takes into account the estimated background variability and storage resources.

A sequential test is used to reduce latency [61]. It is assumed that under the alternative in the k -th bin, the observation distribution changes from p_0 to p_1 . For simplicity, we combine the input information into a single observation Y_k (e.g., a vector $(X_k, \hat{p}_{k,\bullet}, Q_k)$) with densities $f_0(Y)$ and $f_1(Y)$. The classical logarithm of the likelihood ratio is defined as:

$$\ell_k := \log \left(\frac{f_1(Y_k)}{f_0(Y_k)} \right), \quad (17)$$

and the CUSUM statistic W_k is defined recursively:

$$W_0 = 0, \quad W_k = \max(0, W_{k-1} + \ell_k). \quad (18)$$

An event $W_k > H$ detects a change, and the threshold H is chosen by limiting the average run length (ARL) to false alarms. For incomplete knowledge of f_1 , the generalized likelihood ratio (GLR) is applied [62]:

$$\ell_k = \sup_{\theta \in \Theta} \log \left(\frac{f_\theta(Y_k)}{f_0(Y_k)} \right). \quad (19)$$

where Θ is the alternative parameter space, for stationary independent observations with the true alternative f_1 , the logarithm of the likelihood ratio has the mean:

$$\mathbb{E}_1[\ell_k] = D_{KL}(f_1 \| f_0) =: D. \quad (20)$$

For $D > 0$, the sum $\sum_{i=1}^n \ell_k$ —grows linearly. In this context, according to the law of large numbers:

$$\frac{1}{n} \cdot \sum_{i=1}^n \ell_k \xrightarrow{\text{a.s.}} D, \quad (21)$$

then the probability of not detecting a change before n observations (the lag) is exponentially small. More formally, for an arbitrary threshold H , the expected detection time τ behaves as

$$\mathbb{E}_1[\tau] \approx \frac{H}{D}, \quad (22)$$

and the error risk β (false negative) decreases $\sim \exp(-n \cdot D)$ (via the Chernoff bound), which determines a vital measure of “detectability”, which is the local rate of information content per unit time, that is, the KL divergence peak.

To model slow changes and interactions with defense mechanisms, it is convenient to introduce the deterministic dynamics of the attack intensity $a(t)$ (for example, the average intensity of the attacking pool), subject to the ordinary differential equation:

$$\frac{da}{dt} = -\alpha \cdot a(t) + u(t), \quad (23)$$

where $\alpha > 0$ is the average intensity (averaged across the attacker pool), and $u(t)$ is the attacker’s control signal (synchronizing pulses). Solution (23) is represented as:

$$a(t) = a(0) \cdot \exp(-\alpha \cdot t) + \int_0^t \exp(-\alpha \cdot (t-s)) \cdot u(s) ds. \quad (24)$$

Detector dynamics (for example, threshold $h(t)$ with exponential background adaptation):

$$\frac{dh}{dt} = -\gamma \cdot (h(t) - h_0) + \kappa \cdot \sigma_t, \quad (25)$$

where h_0 is the base threshold, γ is the return rate, σ_t is the standard deviation’s local estimate of the S_t statistic, and κ is the adaptation coefficient. Model (25) guarantees threshold stabilization and response to background drift.

In the optimization setting, resource constraints and forensics are taken into account; as in CI [53], the need to store evidence (dumps, hashes) is essential, given the limited storage capacity C and throughput B . It is assumed that $z_k \in \{0, 1\}$ is the decision to store the k -th bin artefact. The storage cost c_k (bytes) and the forensics utility are defined as:

$$\max_{z_k \in \{0,1\}} \sum_k u_k \cdot z_k \text{ s.t. } \sum_k c_k \cdot z_k \leq C, \sum_k b_k \cdot z_k \leq B, \quad (26)$$

which is a two-dimensional knapsack problem. The solution prioritizes bins with the highest ratio— $\frac{u_k}{c_k}$ —and takes into account the notification rate and its legal significance. For online operation, greedy schemes with a threshold $u_k \geq \tau_C$ are used, where τ_C adapts to the current storage capacity:

$$\tau_C(t) = \tau_0 + \eta \cdot \frac{U(t)}{C}, \quad (27)$$

where τ_C is a threshold function that adapts to the current storage volume C (used in the greedy policy).

An approximation for high-speed flows using sketch models and error estimation is that when using Count-Min Sketch [35] to count the number of packets by source, we obtain an approximate estimate of the counters $\tilde{n}_{k,a}$, with an upper bound:

$$\tilde{n}_{k,a} \leq n_{k,a} + \varepsilon \cdot \|\mathbf{n}_k\|_1, \quad (28)$$

where ε is the Count-Min Sketch error characteristic (the counter estimates’ error upper limit), and it is noted that the sketch parameters (width/height) determine ε and the

probability of error. Then, the Herfindahl–Hirschman Index (*HHI*) and entropy can be estimated with errors, which translates into a threshold correction:

$$\widetilde{HHI}_k \leq HHI_k + \mathcal{O}(\varepsilon). \quad (29)$$

The h_k threshold should be increased, taking into account the error to prevent false alarms due to sketch errors:

$$h_k \leq h_k^{(0)} + c_\varepsilon(\varepsilon), \quad (30)$$

where c_ε is a coefficient selected based on the false alarm-controlled levels' property.

Weights $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ are subject to training on historical data or online optimization. Let the labelling contain labels $y_k \in \{0, 1\}$, according to which α is selected as a solution to the regularized risk minimization problem:

$$\min_{\alpha \geq 0} \frac{1}{N} \cdot \sum_{k=1}^N \ell(\sigma(S_k(\alpha) - h), y_k) + \lambda \cdot \|\alpha\|_2^2, \quad (31)$$

where ℓ is the logistic loss function, σ is the sigmoid, and h is the bias. Then, the gradient is defined as:

$$\frac{\partial}{\partial \alpha_i} = \frac{1}{N} \cdot \sum_{k=1}^N (\sigma(S_k(\alpha) - h), y_k) \cdot \sigma'(S_k(\alpha) - h) \cdot g_i(\bullet) + 2 \cdot \lambda \cdot \alpha_i. \quad (32)$$

Statement (31) and (32) naturally transitions to a neural network model, where g_i is the feature layer and α are the linear output layers' parameters.

To determine the statistical guarantees and asymptotics, we consider a GLR-based detector [63,64] with a threshold, H . Under regular conditions and for $D > 0$ (the average log-likelihood value under the alternative), we have the asymptotics:

$$\mathbb{E}_1[\tau_H] \sim \frac{H}{D}, \log \mathbb{E}_\infty[\tau_H] \sim H, \quad (33)$$

where $\tau_H = \inf\{k: W_k > H\}$ and $\mathbb{E}_\infty$ is the expected value of the constant (average run length, ARL) [65], which gives the H choice to perform the ARL:

$$H \approx \log \mathbb{E}_\infty[\tau_H]. \quad (34)$$

To determine the lower bound of the detection rate, it is assumed that the alternative process creates an additional intensity Λ_A per unit time; then, the KL divergence per unit time for Poisson processes [66] is equal to:

$$d = \Lambda \cdot \log \left(\frac{\Lambda}{\Lambda_0} \right) - (\Lambda - \Lambda_0), \quad (35)$$

where $\Lambda = \Lambda_0 + \Lambda_A$. If Λ_A is small, the binomial expansion yields

$$d \approx \frac{\Lambda_A^2}{2 \cdot \Lambda_0} + \mathcal{O}(\Lambda_A^3). \quad (36)$$

From this, it can be seen that for weak attacks— $\Lambda_A \ll \Lambda_0$, the detection power decreases quadratically with respect to the signal-to-noise ratio. For a given acceptable false alarm rate α and background variance estimate $\hat{\sigma}_k^2$, an adaptive threshold with false alarm control is selected according to the expression:

$$h_k = \mu_{S,k} + z_{1-\alpha} \cdot \hat{\sigma}_{S,k}, \quad (37)$$

where $\mu_{S,k}$ and $\hat{\sigma}_{S,k}$ —are the mean and standard deviations' estimates of the rate S (exponential smoothing), and $z_{1-\alpha}$ is the normal distribution quantile. If the distribution is heavy-tailed, then replace z with the corresponding quantile of the bootstrap estimate.

To formalize the rapid detection with forensics problem, it is assumed that the aim is to minimize the average detection time $\mathbb{E}_1[\tau]$ with a limit on the false alarm level $\mathbb{E}_\infty[\tau] \geq \Gamma$ and a limit on the amount of stored artefacts:

$$\min_{\text{detector}, z} \mathbb{E}_1[\tau] \text{ s.t. } \mathbb{E}_\infty[\tau] \geq \Gamma, \mathbb{E} \left[\sum_{k=1}^{\tau} c_k \cdot z_k \right] \leq C, \quad (38)$$

where Γ is the lower bound of the mean time between false positives (ARL minimum value), and C is the available storage amount for artefacts. Solution (38) approximately gives a CUSUM-type procedure for detection and a greedy policy for storing until C is reached.

Based on the above, three mathematical theorems are formulated and proven: Theorem A1, “The Neyman-Pearson theorem for a bin or Poisson detector”, Theorem A2, “Exponential error estimate and detection threshold”, Theorem A3, “Heuristic statement with an optimal CUSUM delay estimate in a discrete Poisson context” (Appendix A).

Thus, the presented model combines input process stochastics (Poisson or Hawkes models), packet size information (Kulban–Leibner divergence, χ^2), source correlation (entropy, HHI), sequential statistics (CUSUM or GLR), resource constraints (knapsack problem), and threshold adaptation algorithms (ODE for $h(t)$). The resulting analytical expressions and theorems provide rigorous guidelines for detector tuning and for formulating the loss function and feature architecture of the neural network model that will implement windowed and online fusion of these signals.

3.2. Development of a Neural Network Method for Detecting UDP Flooding for the Critical Infrastructure Cybersecurity Systems Security

A neural network-based UDP flood detection method is developed based on stochastic models of packet bin counts, KL estimates of packet size distributions, HHI estimates, and source concentration indices, as well as sequential CUSUM procedures for rapid detection and control of the false-positive rate [67,68]. The developed theoretical foundations are used as a priori statistical layers and criteria for the hybrid detector architecture and forensic artefact preservation policy. Figure 6 shows a diagram of a developed method.

It is assumed that in discrete bins of length Δ , there is for bin k :

$$\begin{aligned} \text{count: } X_k &:= N(k \cdot \Delta) - N((k-1) \cdot \Delta), \\ \text{packet sizes: } \mathcal{S}_k &= \{s_{k,1}, \dots, s_k, x_k\}, \\ \text{sources: IP/ASN: } Q_k &= \{a_{k,1}, \dots, a_{k,X_k}\}, \end{aligned} \quad (39)$$

in this case, the vector of observations Y_k is a compact representation of the bin:

$$Y_k = \left(X_k, p_k^{(m)}, H_k, G_k, \text{sketch}_k, t_k \right), \quad (40)$$

where $p_k^{(m)} \in \mathbb{R}^m$ —is the histogram (m bins) of the bin size distribution, H_k is the entropy of sources (calculated according to (12)), G_k is the Herfindahl–Hirschman index (calculated according to (13)), and sketch_k is the count-min sketch vector of approximate counters by flow keys (calculated according to (28)) [69–71].

The preprocessor is a windowing and online aggregation module that generates compact features (incremental statistics, temporal, and spectral characteristics) in real-time and in batches, with normalization, smoothing, and gap recovery for transfer to

the predictor, reconstructor, or detector. Sliding aggregation of length M is performed as follows [72]:

$$\bar{Y}_k^{(M)} = \frac{1}{M} \cdot \sum_{i=k-M+1}^k Y_i. \quad (41)$$

For instant adaptation and cheap memory, an online exponential moving average (EMA) [72] of the form is used:

$$Z_k = (1 - \beta) \cdot Z_{k-1} + \beta \cdot Y_k, 0 < \beta \leq 1. \quad (42)$$

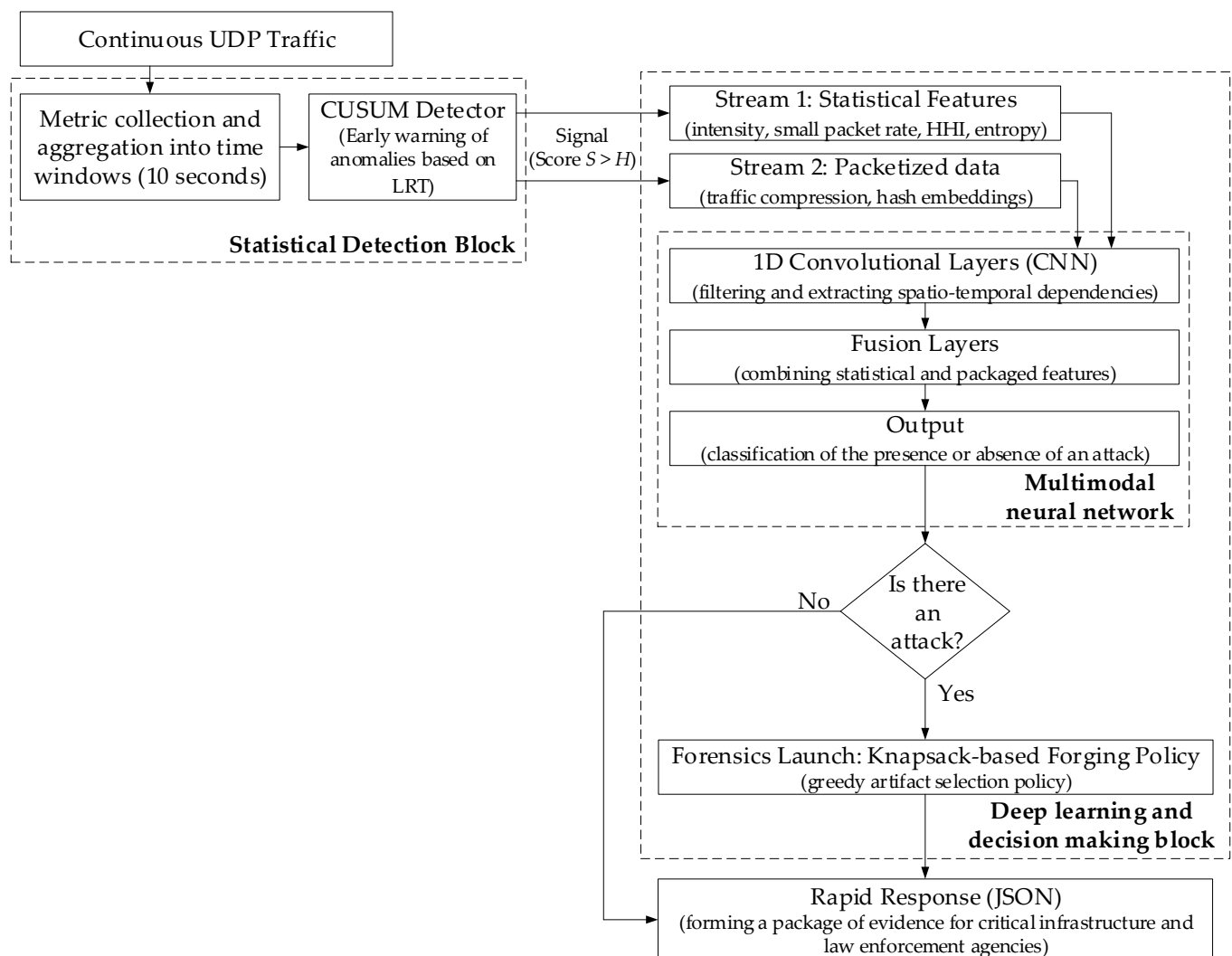


Figure 6. Block diagram of the developed neural network method for detecting UDP flooding.

This pair of mechanisms implements a “two-speed” strategy, in which $\bar{Y}_k^{(M)}$ —captures stable trends (protection against noise spikes), while Z_k represents rapid changes (early spikes). When using sketches, the HHI estimates are adjusted to account for the error ε as in (29) and (30), i.e., the thresholds are adjusted [73]:

$$h_k \leftarrow h_k + c_\varepsilon \cdot \varepsilon. \quad (43)$$

Encoder is a formal layer definitions module that defines a parameterized transformations family (fully connected, convolutional, recurrent or LSTM, transformer blocks, etc.) that, by affine transformations, nonlinear activations, normalization, and attention mech-

anisms composition, map input aggregated features $x_k \in \mathbb{R}^d$ into compact latent vectors $z_k \in \mathbb{R}^h$ with explicit specifications of tensor sizes, initialization schemes, regularization (dropout, L2), residual connections, capacity constraints, and an interface for online or batch training and subsequent feeding to a predictor, reconstructor, or detector.

CountEncoder implements normalization and FNN for the counter, that is [74]:

$$\begin{aligned} u_k^{(c)} &= \text{LayerNorm}(\log(1 + N_k)), \\ e_k^{(c)} &= \phi(W^{(c)} \cdot u_k^{(c)} + b^{(c)}). \end{aligned} \quad (44)$$

SizeEncoder is a module in which the histogram— $p_k^{(m)}$ —undergoes 1D convolution with a filter set to extract local size patterns, followed by nonlinear activation, normalization and (optional) pooling (or aggregation), producing a compact vector of size features for feeding to latent layers, i.e., [75]:

$$e_k^{(c)} = \text{SmoothReLU}\left(\text{Conv1D}\left(p_k^{(m)}; K_s\right)\right) \in \mathbb{R}^{d_s}, \quad (45)$$

where SmoothReLU is a modification of the ReLU activation function developed in [75] and implemented in many neural network methods and models, for example [76,77].

It is noted that an alternative is to calculate a density embedding projection of the form:

$$e_k^{(s)} = W^{(s)} \cdot p_k^{(m)} + b^{(s)}. \quad (46)$$

SourceEncoder is a module that hashes IP addresses or ASNs into a fixed dictionary, followed by embedding into a latent space, where the decoder memory provides compact storage and recovery of source semantic features. For approximate streams, the count-min sketch vector (sketch_k) is used, and the embed is given [75]:

$$e_k^{(q)} = \text{SmoothReLU}\left(W^{(q)} \cdot \text{HashEmbed}(Q_k) + b^{(q)}\right) \in \mathbb{R}^{d_q}. \quad (47)$$

Formally, HashEmbed is implemented as the top- r keys' embeddings' summation; that is, if u is the top- r IP in the sketch, then [75]

$$\text{HashEmbed}(Q_k) = \sum_{j=1}^r \text{Emb}(u_j). \quad (48)$$

Temporal fusion is a module that performs causal convolutions to extract multi-scale temporal patterns while preserving the order of events. It applies an attention mechanism for dynamic weighted aggregation of short- and long-term contexts, providing an interpretable fusion of temporal information before feeding it to a predictor or detector. In Temporal fusion, the modalities assemble into a single vector:

$$E_k = \left[e_k^{(c)}, e_k^{(s)}, e_k^{(q)}, H_k, G_k \right] \in \mathbb{R}^{d_E}, \quad (49)$$

for which causal dilated Conv1D is applied over the L window (temporal depth) to extract temporal patterns (important for “quiet” long-term floods):

$$T_k = \text{Conv1D}_{\text{causal}}(\{E_{k-L+1}, \dots, E_k\}; K_t). \quad (50)$$

In parallel, self-attention provides a weighted aggregation of modalities and temporal positions, that is:

$$\begin{aligned} Q &= T_k \cdot W^Q, \quad K = T_k \cdot W^K, \quad V = T_k \cdot W^V, \\ \text{Attention}(T_k) &= \text{softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d_{\text{att}}}}\right) \cdot V. \end{aligned} \quad (51)$$

The hybrid neural network module is a hybrid model in which the autoencoder reconstructs the background, and the discriminative head classifies anomalies. The inputs flow through “encoder $\rightarrow$ temporal conv $\rightarrow$ attention”, resulting in a latent representation R_k , that is:

$$R_k = \text{LayerNorm}(\text{Attention}(T_k)). \quad (52)$$

Autoencoder (unsupervised path) is trained on regular traffic to evaluate reconstruction error as an anomaly feature:

$$\hat{R}_k = \text{Dec}(R_k), L_{rec}(k) = \|R_k - \hat{R}_k\|_2^2. \quad (53)$$

The classification part (semi-supervised) is the “MLP $\rightarrow$ sigmoid” type discriminative head, trained on labels or pseudo-labels and outputting a scalar score $S_k \in (0, 1)$, interpreted as the anomaly probability (or degree) for threshold detection and subsequent fusion logic. Thus,

$$z_k = \text{SmoothReLU}\left(W_z \cdot \left[R_k, \hat{R}_k, \bar{Y}_k^{(M)}, Z_k\right] + b_z\right), \quad (54)$$

$$S_k = \sigma(\omega^\top \cdot z_k + b).$$

The classification part interpretation is that the vector z_k contains both local (online) and window features, as well as a mismatch feature (reconstruction error), which helps to simultaneously catch sharp bursts and hidden long-term deviations.

The CUSUM Wrapper [77] is a cumulative mechanism that sequentially aggregates signals from the discriminative head (logit estimates) to highlight persistent deviations from background behavior. When the accumulated counter steadily increases, exceeding a predetermined threshold triggers a blocking rule, which consists of immediately isolating or filtering the suspicious flow and logging the event for further investigation. The CUSUM Wrapper uses the logit estimate as an approximation to the log-likelihood ratio (which corresponds to (17)–(19)). Denoting

$$\ell_k := \log\left(\frac{S_k + \epsilon}{1 - S_k + \epsilon}\right), \quad (55)$$

$$W_k = \max(0, W_{k-1} + \ell_k - c),$$

where c is the shift (or compensation) associated with the false alarms and ARL control (which corresponds to (18) and (34)), the detector is triggered when $W_k > H$. To solve the false alarms and attribution problem, a two-level threshold is introduced:

- When $S_k > S_{fast}$, instant blocking (fast path) is performed only if additional conditions match (for example, $\text{HHI} > \text{threshold}$ or a specific ASN in the suspicious list);
- When $S_{slow} < S_k \leq S_{fast}$ is placed in the confirmation queue (confirmatory path) with in-depth analysis of several consecutive bins and verification of the reconstruction error L_{rec} .

The policy for preserving forensic artefacts (knapsack) consists of the fact that when triggered, a decision $z_k \in \{0, 1\}$ is formed about preserving the dump according to the principle (26) and (27), that is, the selection of bins with the maximum forensic score F_k is carried out under the volume limit C . In practice, a greedy algorithm is used with a threshold u_C , adapting to the storage capacity.

Training the developed neural network involves optimizing a combined objective function that combines reconstruction loss, a discriminative (classification) component, and

regularizers with adjustable weights for balancing tasks and controlling model behavior in online or batch modes. The overall loss function for a batch, b , in mixed mode is defined as:

$$L = \lambda_{cls} \cdot L_{cls} + \lambda_{rec} \cdot L_{rec} + \lambda_{ctr} \cdot L_{ctr} + \lambda_{reg} \cdot \|\Theta\|^2, \quad (56)$$

where L_{cls} is the binary cross-entropy on the available labels y_k , defined as:

$$L_{cls} = -\frac{1}{|B|} \cdot \sum_{k \in B} (y_k \cdot \log(S_k) + (1 - y_k) \cdot \log(1 - S_k)), \quad (57)$$

and L_{rec} is defined as:

$$L_{rec} = \frac{1}{|B|} \cdot \sum_k L_{rec}(k), \quad (58)$$

and L_{ctr} is an optional contrastive term for separating norms and anomalies in the latent (if there are labels), Θ are the neural networks' parameters, and the gradient step is carried out according to the expression:

$$\Theta \leftarrow \Theta - \eta \cdot \nabla_{\Theta} L. \quad (59)$$

It is noted that for online retraining, we use a limited replay buffer and a small learning rate η_{online} , as well as the “elastic weight consolidation” mechanism or regularization to update old parameters, preventing catastrophic forgetting under drift conditions.

The threshold H is chosen according to the desired ARL Γ according to the CUSUM asymptotics (which corresponds to Theorem A3, (34) and (A45)):

$$H \approx \log(\Gamma) + O(1). \quad (60)$$

The shiftable threshold in the CUSUM recursion through c allows one to specify the desired false alarm rate and sensitivity. The built-in statistics $\mu_{S,k}$, $\sigma_{S,k}$ (exponential smoothing) are used for adaptive normalization and calculation of z-statistics (according to (25) and (37)):

$$\begin{aligned} \mu_{S,k} &= (1 - \gamma) \cdot \mu_{S,k-1} + \gamma \cdot S_k, \\ \sigma_{S,k}^2 &= (1 - \gamma) \cdot \sigma_{S,k-1}^2 + \gamma \cdot (S_k - \mu_{S,k})^2. \end{aligned} \quad (61)$$

Thus, the developed neural network's architecture is optimized for compactness and low computational costs. In this context, the number of parameters in 1D convolutions is limited due to narrow filters and a small number of channels, embedding tables are implemented as small hash-embedding tables, and sketch structures and EMA (exponential moving average) minimize the maintained state and amount of memory consumption. Moreover, the autoencoder has a narrow latent space of dimension $d \approx 32 \dots 128$, which further reduces the computational load and storage requirements. Logit scores ℓ_k approximate the log-likelihood ratio between models f_1 and f_0 (according to (17)), and sequential summation W_k implements CUSUM (18), which ensures asymptotic optimality in detection rate for a given ARL in accordance with the theorems presented in the theoretical section. The autoencoder reconstruction error serves as an estimate of the deviation from $p_0(\bullet)$ (which corresponds to DKL in (10) and (11)), which improves the low-intensity long floods detection when the counter X_k deviates little, but the source size distribution is modified. Thus, based on the above, the training algorithm for the developed neural network is presented in Table 2.

Table 2. The developed neural networks' training algorithm (the results are based on mathematical justification).

Stage Number	Stage Name	Input	Output	Key Actions
1	Data preparation	PCAP or flow, labels (if any)	Y_k window, sketch, histograms	Collection, augmentation, and cutting of “normal” windows for AE minimization of L_{rec}
2	AE pretraining	Normal traffic windows	Pretrained decoder and encoder	Minimization of L_{rec}
3	Supervised fine-tune	Labeled windows	Discriminative head	Minimization of $L_{cls} + \lambda \cdot L_{rec}$
4	Contrastive (or domain adaptation)	Different domains (topologies)	Robust latent representability	Contrastive loss (adversarial alignment)
5	Pruning and quantization	Trained model	Compact model for CI	Weight pruning, INT8 quantization
6	On-device (edge deployment)	Model with sketch params	Working detector	Tuning Δ, M, β, H, c , and CUSUM integration
7	Online training	Streaming labels (operator feedback)	Drift adaptation	Small LR, replay buffer, EWC regularization
8	Forensics policy	Alerts with resource C	Artifact collection	Greedy knapsack selection, storage, hashing, signing

The developed neural network is implemented in the MATLAB Simulink R2023b software environment (Figure 7) as a flow preprocessing module and the neural network subsystem itself (inference with CUSUM blocking logic).

The “From Workspace” or “UDP Receive” block (depending on the source) is used at the input to capture packets (or flow metrics). Windowing and online aggregation are then performed by the “Buffer” block (sliding frame) and a small sequence of arithmetic blocks with “Unit Delay” for EMA. Sketch and hashing elements (count-min sketch, top-r IP) are implemented within the “MATLAB Function” block (which performs compact C/M sketch and hash embedding). In contrast, standard arithmetic blocks perform histogram size calculation and HHI (or entropy) and the “MATLAB Function” block for the efficient implementation of histogram vectors. The forensic buffer and storage manager are implemented by the “Data Store Memory” and “To Workspace” blocks with volume control via the logical “Subsystem”.

The neural network component is separated into a separate “NN_Inference” subsystem. The hybrid neural network (autoencoder with temporal conv, attention, and MLP) is trained in MATLAB (Deep Learning Toolbox) as “dlnetwork” (or “SeriesNetwork”) and then connected to Simulink via the “Deep Learning Toolbox Predict” block (or via a “MATLAB Function” executing `predict('dlnet', dldata(...))`) to maintain compatibility and simplify debugging. Within the Subsystem, input features (window vectors and EMA vectors) are passed through a pretrained modules sequence, including the “Encoder block” (in Simulink, this is the “Predict” block with a vector input, which provides embedding), then the “Temporal Conv” (in Simulink, either a built-in DL layer when exporting the network or an implementation in a “MATLAB Function” with “dlconv”), then the “Attention block” (implemented as part of the exported network), and finally the “AE decoder with MLP scorer” (again, an exported module). For real-time operation, the CUSUM logic and two-level locking are implemented as a separate Enabled Subsystem (fast path) with a

recursive counter implemented via “Unit Delay”. In contrast, the confirmatory path is a timer (counter) that tracks sequential triggers and conditional logic via Logical Operators and Switches. For online retraining and adaptation, a feedback interface (label feedback) is provided via “From Workspace” and “MATLAB Function” for periodic updating of weights (implementing the update in MATLAB and loading the updated network into the “Predict” block).

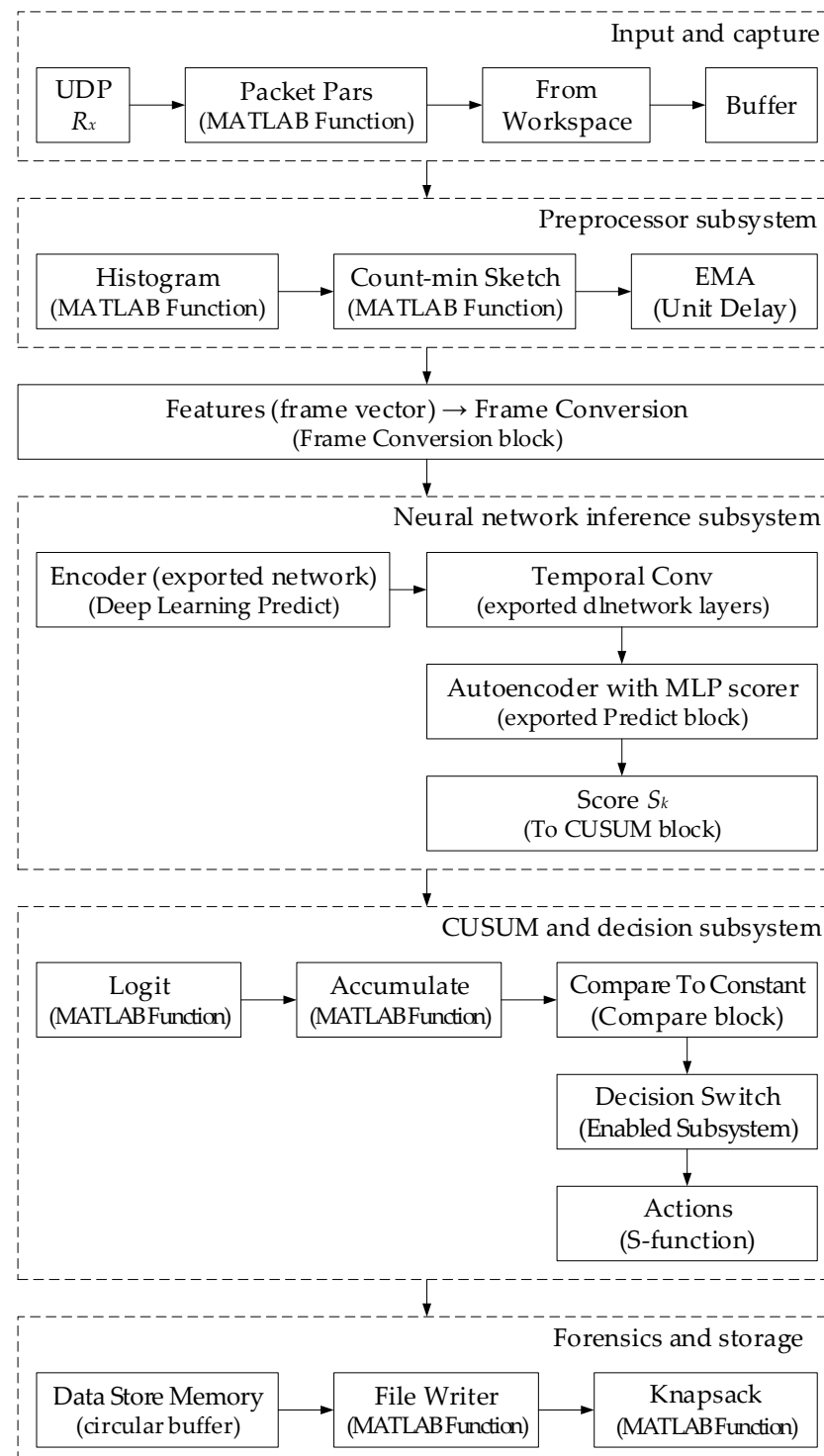


Figure 7. Block diagram of the developed neural networks' implementation in Matlab Simulink R2023b.

3.3. Selection of the Developed Neural Networks' Hyperparameters

The hyperparameter selection procedure for the developed hybrid neural network is systematic and reproducible, and the limitations of the CI environment are duly considered. In the first stage, quality metrics (detection AUC, false alarm rate, ARL, reconstruction error, latency, throughput) and resources (RAM, CPU, latency budget) are fixed. Then, a coarse grid or random search is performed over wide ranges of key hyperparameters (learning rate, batch size, latent dim, window M , EMA β , number of filters in Conv1D, kernel, delay, number of attention heads). First, a stable range of learning rate and batch size is determined (they critically affect convergence and gradient noise). Then, the most influential parameters are fixed, and a more fine-grained optimization (Bayesian optimization, hyperband) for latent dimensions and structural parameters (number of layers, convolutional filter widths, attention heads) is performed. Next, quality control is performed across several scenarios, including regular traffic, low-rate stealth UDP flood simulation, and peak loads. For each scenario, ARL and AUC are estimated for various combinations of hyperparameters. Threshold parameters (CUSUM H , c offset, fast or slow thresholds) are selected separately using ARL or EDD (expected detection delay) simulations, and a safety margin is established for production use. Equally important is the ablation study stage, during which components (AE, attention, sketch) are sequentially disabled, recording the impact on metrics and latency. It helps determine the tradeoff between quality and performance and select a reduced model for edge devices (pruning and quantization). An online tuning strategy was introduced, which includes a low learning rate for online updates, a fixed-size replay buffer, and EWC-like regularization to avoid catastrophic forgetting during traffic drift.

Figure 8 illustrates the U-shaped relationship between validation loss and learning rate, where a too-small learning rate results in slow convergence and high loss, a too-large learning rate results in divergence and noise, and the optimal range lies at the minimum in the middle.

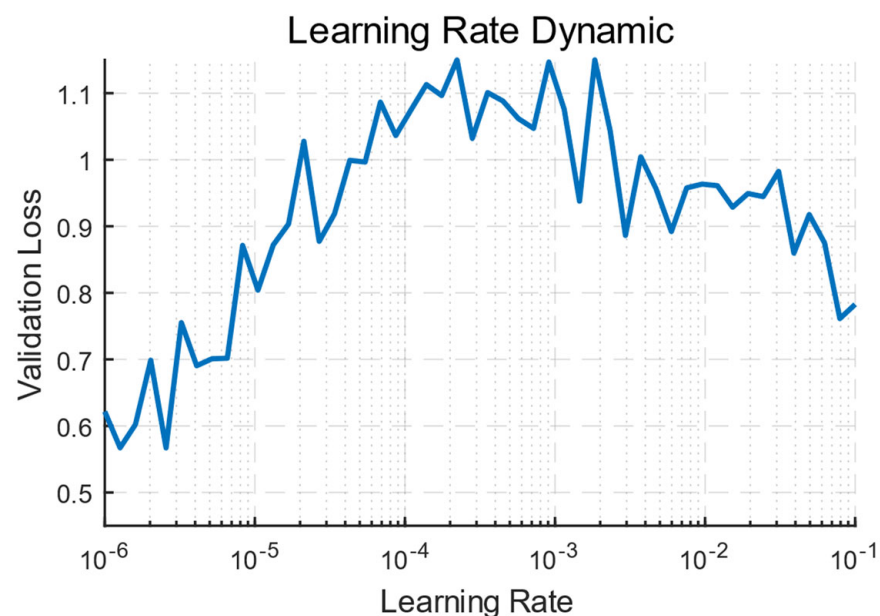


Figure 8. Validation loss vs. learning rate diagram.

According to Figure 8, the tuning first performs a logarithmic (coarse) learning rate search, then reduces the search step in the minimum region. Additionally, a learning rate schedule (such as cosine or step) or a warmup is applied initially to ensure stability.

Figure 9 demonstrates that increasing the batch size often improves gradient stability and validation accuracy up to a specific limit. However, as the batch size increases, the update frequency decreases, and adaptation to rare events may deteriorate.

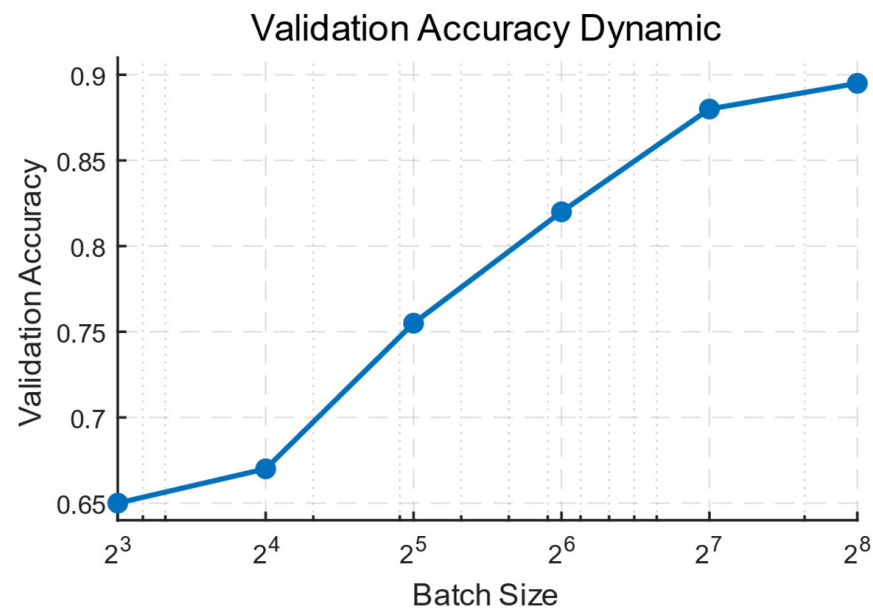


Figure 9. Diagram of lot size versus inspection accuracy.

According to Figure 9, the largest possible batch size that fits in memory is selected and then compensated for by increasing the learning rate (using the linear scaling rule) if necessary, or by employing gradient accumulation. As shown in Figure 10, the reduction in reconstruction error with increasing latent dimension demonstrates a trade-off between latent expressiveness and cost (memory and latency).

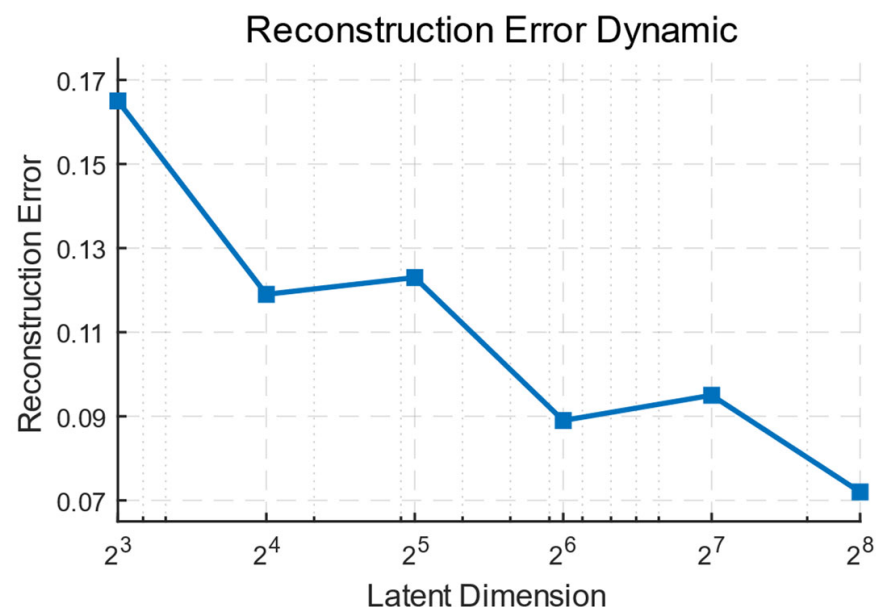


Figure 10. Diagram of hidden measurement as a reconstruction error function.

According to Figure 10, the minimum latent size is selected to achieve acceptable values for the reconstruction error and downstream detection AUC, and then pruning or quantization is applied for optimization.

Based on the above, Table 3 provides a hyperparameter systematization for the developed neural network.

Table 3. The developed neural networks' hyperparameters (results are based on neural network architecture).

Hyperparameter	Recommended Initial Range	Selection Method	Notes
Learning rate (train)	$10^{-6} \dots 10^{-2}$	Log-grid (Bayesian)	Finding the optimum validation value using warmup and scheduler
Optimizer	Adam, AdamW, SGD with momentum	Empirical	Applying AdamW for weight regularization and SGD for large batches
Batch size	8 ... 256 (power-of-2)	Memory-limited grid	Large batches speed up training but require learning rate scaling
Window size M (frames)	4 ... 256 (depends on Δ)	Grid (scenario tests)	Finding a balance between latency and stability
EMA beta (online)	0.01 ... 0.2	Grid	A smaller β is a slower reaction, and more stable
Latent dimension (AE)	32 ... 256	Grid with downstream AUC	Select the minimum that satisfies the AUC and reconfiguration error
Conv1D filters per layer	8 ... 128	Grid (pruning later)	The more filters, the better the performance, but the more expensive it is
Conv. kernel size	3 ... 11	Grid	Short for high-frequency, long for slow patterns
Dilation rates	1, 2, 4, 8	Manual (search)	For long-term dependencies
Attention heads	1 ... 8	Grid	More heads mean better interaction modelling
AE reconstruction weight λ_{rec}	0.1 ... 1.0	Grid	Balance with classification loss
Classification weight λ_{cls}	1.0 (baseline)	Fixed (tune)	Primary for supervised fine-tuning
Contrastive weight $\lambda_{contrast}$	0 ... 1.0	Optional	If contrastive examples are used
Regularization (weight decay)	$10^{-6} \dots 10^{-3}$	Grid (Bayesian)	A key metric for online learning
CUSUM threshold H	tuned via ARL sim	Simulation-based	Suiting via ARL or EDD modelling
CUSUM shift c	small positive	Simulation-based	Controls sensitivity
Fast and slow thresholds S_{fast}, S_{slow}	0.6 ... 0.95	ROC with ARL	Tune with a safety margin
Online learning rate η_{online}	$10^{-7} \dots 10^{-5}$	Conservative	Small learning rate for adaptation without forgetting
Replay buffer size	1000 ... 100,000 windows	Resource-limited	Balance between memory and diversity
Pruning rate (quantization)	0 ... 90% (or INT8)	Post-training	For edge deployment
Epochs (pretrain AE)	50 ... 500	Early stopping	Early stopping by recon error with validation AUC
Epochs (fine-tune)	10 ... 200	Early stopping	AUC (or ARL) monitoring
Learning rate schedule	step, cosine or reduce-on-plateau	Selective	Reduce-on-plateau is useful for drift

Table 4 presents a rationale for the impact of convolutional kernel size and compression scheme choice on the trade-off between detection accuracy, computational complexity, and processing latency, allowing us to determine the optimal model configuration for practical deployment.

Table 4. Comparison of research results on varying the convolutional kernel sizes and compression schemes.

Model (Configuration)	Convolution Kernel	Compression Scheme	AUC	Precision	Recall	FPR	Latency (ms), Median	Parameters (Thousands)	MACs (M)	Size (MB)
Base (reference)	3×3	No	0.94	0.91	0.88	0.06	5.3	1200	120	4.6
Enlarged kernel	5×5	No	0.945	0.92	0.895	0.055	6.8	1800	180	6.8
Large kernel	7×7	No	0.943	0.915	0.892	0.056	8.5	2600	260	9.8
Pruned (50% pruning)	3×3	pruning 50%	0.92	0.89	0.85	0.08	3.2	600	60	2.5
Int8 quantization	3×3	quant8	0.938	0.90	0.87	0.065	3.9	1200	120	1.2

According to Table 4, increasing the convolutional kernel size (from 5×5 to 7×7) provides only a slight improvement in AUC and recall, but significantly increases the number of parameters, MACs, and median latency. It is also noted that compression schemes (pruning, quant8) sharply reduce model size and latency with a moderate decrease in accuracy. Pruning provides the most significant resource reduction but also results in a noticeable increase in FPR. At the same time, int8 quantization offers the best balance of small size, low latency, and minimal quality degradation. Therefore, quantization (or a combination of pruning and quantization with careful validation) is preferable for edge deployments, while a 5×5 kernel may be optimal for server configurations, resulting in a slight increase in detection metrics.

3.4. Estimation of the Developed Neural Networks' Computational Cost as Part of the Developed UDP Flood Detection Method

The computational cost is estimated analytically under the following assumptions: $L = 32$, $d_{in} = 128$, $d_{model} = 128$, $h = 4$, $d_{latent} = 64$, and three 1D-Conv layers with filters [32,64,65] and a kernel size of $k = 3$. The number of parameters for the convolutional branch is calculated as:

$$P_{conv} = \sum_l (f_l \cdot c_{l-1} \cdot k + f_j), \quad (62)$$

and gives $P_{conv} = 12,320 + 6208 + 12,352 = 30,880$.

The parameters of the multi-head attention block with FFN are estimated as the sum of the QKV projections and the output projection:

$$P_{att} = 3 \cdot (d_m^2 + d_m) + (d_m^2 + d_m), \quad (63)$$

and a two-layer FFN with an internal width of 256, which gives $P_{att} \approx 66,048 + 65,920 \approx 131,968$.

The autoencoder (linear encoder and decoder $128 \leftrightarrow 64$) adds $\approx 166,000$, and the compact classifier (MLP $64 \rightarrow 32 \rightarrow 1$) $\approx 21,000$, for a total of $P_{total} \approx 1,815,000$ parameters, which fits within the edge (or near-edge) boundaries.

The number of operations per pass in Conv1D (one layer) is defined as:

$$MAC_{s_{conv}} = \sum_l f_l \cdot L \cdot k \cdot c_{l-1} = 983,040. \quad (64)$$

For the attention mechanism, the QKV projections and the output projection $3 \cdot d_m^2 \cdot L + d_m^2 \cdot L$ are defined, as well as the matrix products in attention $2 \cdot h \cdot L^2 \cdot d_k$

give $MAC_{s_{att}} \approx 2,359,296$. The encoder, decoder, and classifier together give an additional $\approx 590,000$ MACs, so the total computational load for one pass window $L = 32$ is of the $MAC_{s_{total}} \approx 3,900,000$ ($\approx 3.93 \cdot 10^6$) order, that is, an order interpretable for estimating the throughput and power consumption in an embedded device.

Table 5 shows the calculations of the number of parameters and the number of multiplication-addition operations (MACs) for the key components of the hybrid neural network, namely, Conv1D (3 layers), Multi-Head Attention with FFN, autoencoder (encoder and decoder), classifier (MLP), and utility structures (sketch, EMA). Figure 11 shows a plot of MACs versus attention model size.

Table 5. Results of the evaluation of computational costs for the developed neural networks (results are based on computational operations).

Component	Parameters (Count)	MACs (Multiply-Accumulate Ops)	Percentage of Total MACs	Percentage of Total Parameters
Conv1D (3 layers)	30,880	983,040	25.87%	16.02%
Multi-Head Attention with FFN	131,968	2,359,296	62.07%	68.54%
Autoencoder (encoder, decoder)	16,600	300,000	7.90%	8.62%
Classifier (MLP)	2100	90,000	2.37%	1.09%
Others (sketch, EMA, overheads)	4000	200,000	5.26%	5.23%
Totals	185,548	3,932,336	100.00%	100.00%

Sensitivity of MACs to Attention model dimension ($L = 32$)

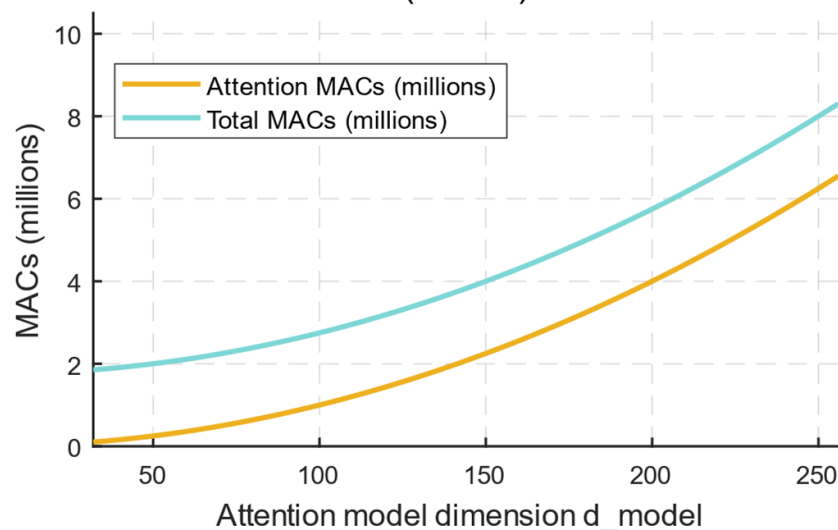


Figure 11. Diagram of MACs' dependence on the size of the attention models.

Figure 11 shows the computational load dependence (MACs, millions) on the attention model dimension d_{model} for a fixed window length $L = 32$, where the “orange curve” shows the attention MACs only with FFN, and the “blue curve” shows the total MACs of the entire architecture. Figure 11 shows a quadratic increase in the load for d_{model} (the term— $MAC_{s_{att}}(d_{model}, L) = 4 \cdot d_{model}^2 \cdot L + 2 \cdot L^2 \cdot d_{model}$ —from projections and FFN dominates), due to which the attention block quickly becomes the primary consumer, that is, with $d_{model} = 128$, it produces $\approx 2,360,000$ MACs ($\sim 60\%$ of the total). Thus, the quadratic part of the d_{model} (design matrices and FFN) makes the most significant contribution as the model dimension increases, so reducing the d_{model} is the most effective way to reduce the load.

4. Case Study

4.1. Formation, Analysis, Preparation, and Pre-Processing of the Training Dataset

To conduct a case study on solving the UDP flood detection problem, traffic was obtained in equal bins with a time interval of $\Delta = 1$ s ($n = 400$ slices) against a background described as a Poisson process with low seasonality and additional noise (Figure 12). It is noted that three types of attacks are integrated into the graph, namely, “short_high” (short intense burst, 40...55 s), “long_low” (long-term low-rate “silent” flood, 140–220 s), and “medium_burst” (short moderate burst, 300...308 s). For each bin, a set of features was created, which includes “packets” (number of packets), “avg_packet_size” (bytes), “src_entropy”, and HHI. During attacks, the average packet size distribution shifts towards smaller sizes (the small packets concentration reproduces the UDP flood profile), and the metrics characterizing the sources distribution, “src_entropy” and HHI, change in accordance with the source’s concentration degree (as the concentration increases, entropy decreases and HHI increases), which makes it possible to quantitatively distinguish between types of anomalies in time series.

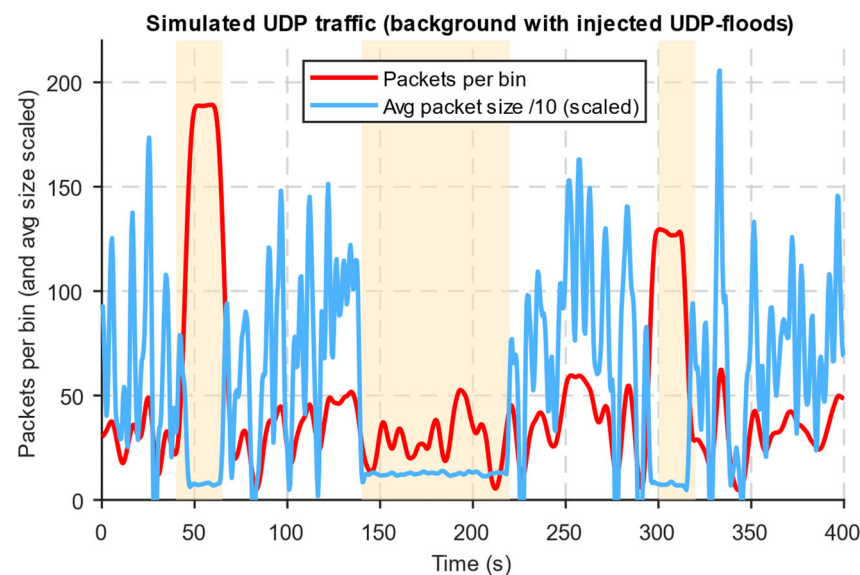


Figure 12. Critical infrastructure facility temporary traffic diagram.

It is noted that the traffic under study (Figure 12) was obtained in a real system as follows. Using the example of a critical infrastructure facility distributed unit, namely, a remote terminal unit (RTU in a power grid substation at the network edge or inside a SCADA segment), a passive collector (network TAP or SPAN port) is installed that captures packet headers and exports flow metrics (NetFlow or IPFIX) and aggregated counters. At the same time, on the edge device or central collector, this data is aggregated into 1-s bins ($\Delta = 1$ s), the number of UDP packets is counted, a histogram (packet_size) is built, a list of top-k source IPs is formed (or a compact approximation via count-min sketch), and time metadata is saved with NTP or GPS synchronization for accurate alignment of time series. Next, from the bin counters, the features “packets per bin” X_k , “avg_packet_size”, “src_entropy” (Shannon), and HHI (Herfindahl–Hirschman index) are calculated, which, in the sliding windows or vector features form, are transferred to the developed neural network for constructing a time series, visualization, and subsequent analysis of anomalies. The full dataset comprises 5 days of continuous network traces collected from three campus network edge devices, corresponding to approximately 432,000 s (i.e., 432,000 intervals at a 1-s interval). Four thousand consecutive intervals were used for the experiments, of which 60% (2400) were used for training, 20% (800) for validation, and 20% (800) for model

testing. Attack events in the dataset were generated by synthetically injecting controlled UDP flood scenarios of varying intensity and duration, ensuring accurate “true” labels and enabling reproducible evaluation of the method’s effectiveness and robustness under conditions similar to those of real network loads.

Based on the above, Table 6 presents selected rows from the original dataset, aggregated by 1-s bins. Each row contains the fields, where “time_s” is a sample time in seconds (bin), “packets” is the number of UDP packets in this bin, “avg_packet_size_bytes” is the average packet size in bytes in this bin, “src_entropy” is the IP source distribution entropy (characterizes the sender’s distribution), “HHI” is the source concentration index (Herfindahl-Hirschman Index), and “label_udp_flood” is the class label (“1” is a bin with injected UDP flooding, and “0” is background data). It is noted that a 1 min interval represents one specific scenario in which traffic characteristics including packet count, average size, and HHI are analyzed to simulate different types of attacks.

Table 6. The input dataset’s fragment (results are based on mixed scenarios).

time_s	packets	avg_packet_size_bytes	src_entropy	HHI	label_udp_flood
0	19	782	3.125	0.1382	0
...	...	...	...	...	...
7	22	591	3.373	0.1344	0
...	...	...	...	...	...
20	23	172	3.230	0.1238	0
...	...	...	...	...	...
26	16	233	3.295	0.1463	0
...	...	...	...	...	...
39	21	452	3.228	0.1190	0
...	...	...	...	...	...
40	208	113	5.172	0.0548	1
...	...	...	...	...	...
41	201	88	4.404	0.0443	1
...	...	...	...	...	...
53	207	105	4.612	0.0386	1
...	...	...	...	...	...
54	201	102	4.668	0.0423	1
...	...	...	...	...	...
79	18	698	2.890	0.1013	0
...	...	...	...	...	...
106	18	534	2.484	0.1133	0
...	...	...	...	...	...
133	14	686	2.662	0.1242	0
...	...	...	...	...	...
139	14	860	2.925	0.1208	0
...	...	...	...	...	...
140	28	109	3.644	0.0264	1
...	...	...	...	...	...
149	20	54	4.165	0.0512	1
150	20	45	3.912	0.0365	1

It is noted that the input dataset (see Table 5) was obtained from a real RTU trace, aggregated into uniform time intervals of $\Delta = 1$ s, which provides a compromise between smoothing out background fluctuations and promptly detecting short-term attack bursts. Furthermore, it is noted that the input dataset includes approximately 8000...12,000 unique UDP flows (by source IP and port), typical for a highly loaded campus network segment. For each time window, basic network features were calculated, namely, the number of UDP packets, the average packet size, the source distribution entropy, and the HHI concentration index. Accurate labels were formed by controlled injection of three UDP flood attack scenarios (“short_high”, “long_low”, “medium_burst”), after which each time bin was assigned a binary “attack/background” label in accordance with the active injection intervals.

The input dataset's (see Table 5) homogeneity was assessed using the Levene W test for equality of variances (robust to non-normality), the Kruskal–Wallis H test for comparing medians of several groups (non-parametric ANOVA), and the Kolmogorov–Smirnov (two-sample KS) D test for pairwise comparison of distributions defined as [78–81]:

$$W = \frac{(N - k) \cdot \sum_{j=1}^k n_j \cdot (Z_{j\bullet} - Z_{j\bullet\bullet})^2}{(k - 1) \cdot \sum_{j=1}^k \sum_{i=1}^{n_j} (Z_{ij} - Z_{j\bullet})^2},$$

$$H = \frac{12}{N \cdot (N + 1)} \cdot \sum_{j=1}^k \left(\frac{R_j^2}{n_j} \right) - 3 \cdot (N + 1), \quad (65)$$

$$D = \sup_x (F_1(x) - F_2(x)),$$

where $Z_{ij} = \left| X_{ij} - \tilde{X}_j \right|$ denotes median centering of $\tilde{X}_j$, n_j is the size of the j -th group, $N = \sum n_j$, k is the number of groups, and R_j is the sum of ranks in the j -th group. It is noted that the Levene test assesses the equality of variances between groups, where a small p -value indicates heterogeneity of variances. The Kruskal–Wallis null hypothesis states that the samples are obtained from identical distributions (in particular, have the same medians with similar distribution shapes), while the Kolmogorov–Smirnov two-sample test compares two empirical distributions. A small p -value indicates a statistically significant difference between these distributions.

Based on the above, Table 7 presents aggregated statistics for four features (“packets”, “avg_packet_size”, “src_entropy”, and HHI). The obtained results include mean values and standard deviations for the background sample and three attack types (“short_high”, “long_low”, “medium_burst”), as well as the homogeneity tests’ results, which consist of estimating the Levene p -value [80] (equality of variances), the Kruskal–Wallis test [78] (comparison of multiple groups distributions), and paired Kolmogorov–Smirnov [79,81] tests for each attack group compared to the background.

Table 7. Results of the input datasets’ homogeneity determination (results are based on computational operations).

Metric	packets	avg_packet_size	src_entropy	HHI
Mean (“background”)	20.0680	807.3552	6.5112	0.0792
Mean (“short_high”)	195.5625	143.2411	3.0164	0.4464
Mean (“long_low”)	39.8642	604.6365	5.2854	0.1771
Mean (“medium_burst”)	121.0000	192.8263	3.6765	0.3392
Std (“background”)	5.3130	116.3703	0.3018	0.0188
Std (“short_high”)	10.4815	24.9066	0.3463	0.0386
Levene (p -value)	0.0000	0.0000	0.0938	0.0000
Kruskal (p -value)	0.0000	0.0000	0.0000	0.0000
KS (p -value for “short_high” vs. “background”)	0.0000	0.0000	0.0000	0.0000
KS (p -value for “long_low” vs. “background”)	0.0000	0.0000	0.0000	0.0000
KS (p -value for “medium_burst” vs. “background”)	0.0000	0.0000	0.0000	0.0000

Table 7 analysis shows that, for all features except “src_entropy”, the Levene and Kruskal–Wallis p -tests are close to zero, indicating statistically significant differences in variance and distribution between the background and attack intervals. Paired Kolmogorov–Smirnov tests also show $p \approx 0$ for all attacks, confirming sample heterogeneity and a significant bias in feature distributions in the presence of the anomaly. Consequently, the training dataset is statistically heterogeneous, which must be taken into account when forming training and test subsets for the neural network detectors to ensure correct training.

Figure 13 illustrates the distribution of packets per bin for background traffic and three attack types, highlighting the characteristic differences in flow intensity between normal and abnormal intervals.

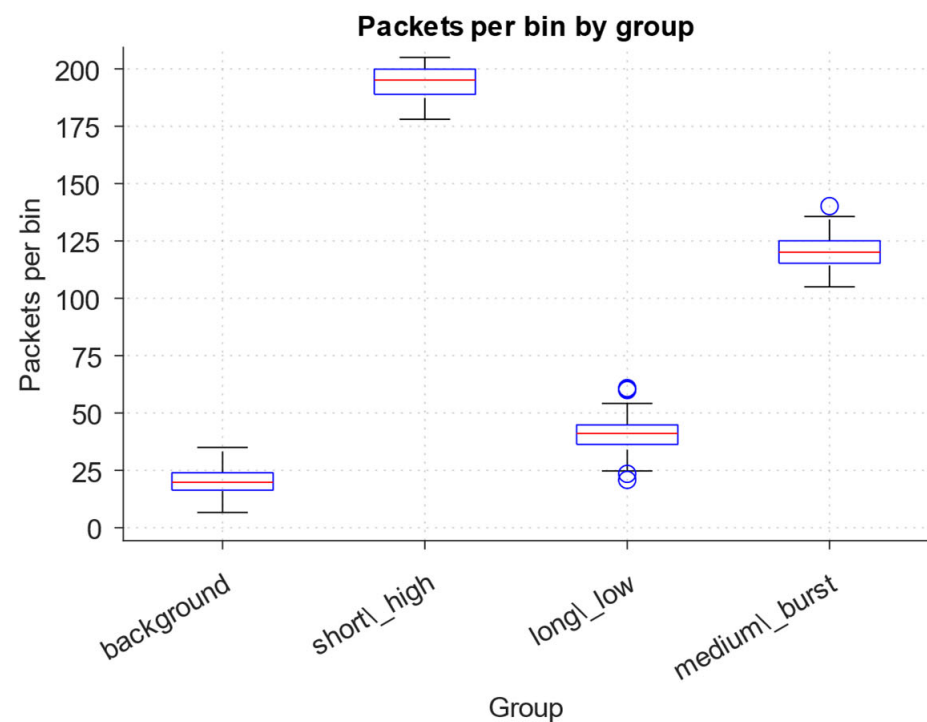


Figure 13. Distribution diagram of the number of packets by bins.

Figure 13 illustrates the different packet count distribution structures for background traffic and the three attack types, confirming the heterogeneity of the input dataset. The “short_high” and “medium_burst” intervals are characterized by pronounced outliers and a significantly higher median packet count than the background, while “long_low” demonstrates a moderate but consistent increase in intensity. Thus, graphical analysis of the distributions is consistent with the statistical tests’ results and confirms the presence of several classes with significantly different dynamics, making the dataset heterogeneous and informative for training anomaly detectors.

The mathematical analysis of using a heterogeneous dataset for training a neural network is based on the need to ensure a balance between the representation of background and anomalous states while maintaining the statistical stability of the parameters. Let the training dataset $D = \{(x_i, y_i)\}_{i=1}^N$ —consists of subsets D_b (background) and $D_a = D_{sh} \cup D_{ll} \cup D_{mb}$, that is, a set of attack states (“short_high”, “long_low”, “medium_burst”). Then, the overall data distribution can be written as:

$$p(x) = \pi_b \cdot p_b(x) + \sum_{k=1}^3 \pi_{a_k} \cdot p_{a_k}(x), \quad (66)$$

where π_b and π_{a_k} are prior probabilities (class weights) satisfying $\pi_b + \sum_{k=1}^3 \pi_{a_k} = 1$. Since $p_b(x) \neq p_{a_k}(x)$, the dataset is statistically heterogeneous ($\text{Var}(p(x)) > 0$). For correct network training, it is necessary to ensure normalization of input features, taking into account intra-class differences:

$$x'_i = \frac{x_i - \mu_c(i)}{\sigma_c(i)}, \quad (67)$$

where $\mu_c(i)$ and $\sigma_c(i)$ are the mean and standard deviation of features for class $c(i)$, this local standardization reduces the variance shift influence between subsets. It ensures the loss functions' stability, which is adjusted using class weights on a heterogeneous dataset:

$$L = -\sum_{i=1}^N \omega_c(i) \cdot y_i \cdot \log(\hat{y}_i), \quad (68)$$

where $\omega_c(i) = \frac{1}{\pi_c(i)}$ —compensates for the imbalance in the dataset size between normal and abnormal states.

It is known that under the training dataset heterogeneity conditions, the k -means method [82–85] serves as a practical tool for quantitatively assessing the heterogeneous datasets' representativeness, since, by dividing the entire diversity of features into K clusters with centers $\{\mu_k\}_{k=1}^K$ —using the method of minimizing the total intra-cluster square error (inertia) [83]:

$$SSE = \sum_{k=1}^K \sum_{x_i \in C_k} \|x_i - \mu_k\|^2, \quad (69)$$

an a priori partitioning of the feature space into data “regions” was obtained, each of which has a size n_k and a proportion $R_k = \frac{n_k}{N}$. To assess local representativeness, it is helpful to additionally calculate the average intracluster distance [82,83]:

$$\bar{d}_k = \frac{1}{n_k} \cdot \sum_{x_i \in C_k} \|x_i - \mu_k\|, \quad (70)$$

and the silhouette of each object [82,83]:

$$s_i = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad (71)$$

where $a(i)$ is the average distance within a cluster, $b(i)$ is the minimum average distance to another cluster, and the average silhouette of a cluster shows its “coherence” and suitability for representing data.

Thus, in the training dataset heterogeneity conditions, clusters provide a basis for stratification and for calculating weights in the loss function; for example, the compensating weight of a cluster can be defined as [84]:

$$\omega_k \propto \frac{1}{R_k + \epsilon}, \quad (72)$$

or [85]:

$$\omega_k \propto \frac{1}{\sqrt{R_k + \epsilon}}, \quad (73)$$

and the weight for a specific example x_i is taken to be $w_c(i)$, where $c(i)$ is the cluster index, which will reduce the learning bias in favor of large (overrepresented) regions of the feature space and increase sensitivity to rare but critical anomalous subspaces [85,86].

Figure 14 presents the training dataset representativeness assessing results using the k -means method, including the squared errors sum (SSE) dependence on the number of clusters (Figure 14a), an average silhouette coefficient diagram (Figure 14b), the distribution of cluster sizes (Figure 14c), and their projection into the first two principal components space (Figure 14d). The resulting diagrams enable one to clearly determine the optimal number of clusters, identify the relationships between the primary and rare classes, and assess their degree of separability and the internal coherence of features, thereby confirming the heterogeneity and structure of the training dataset.

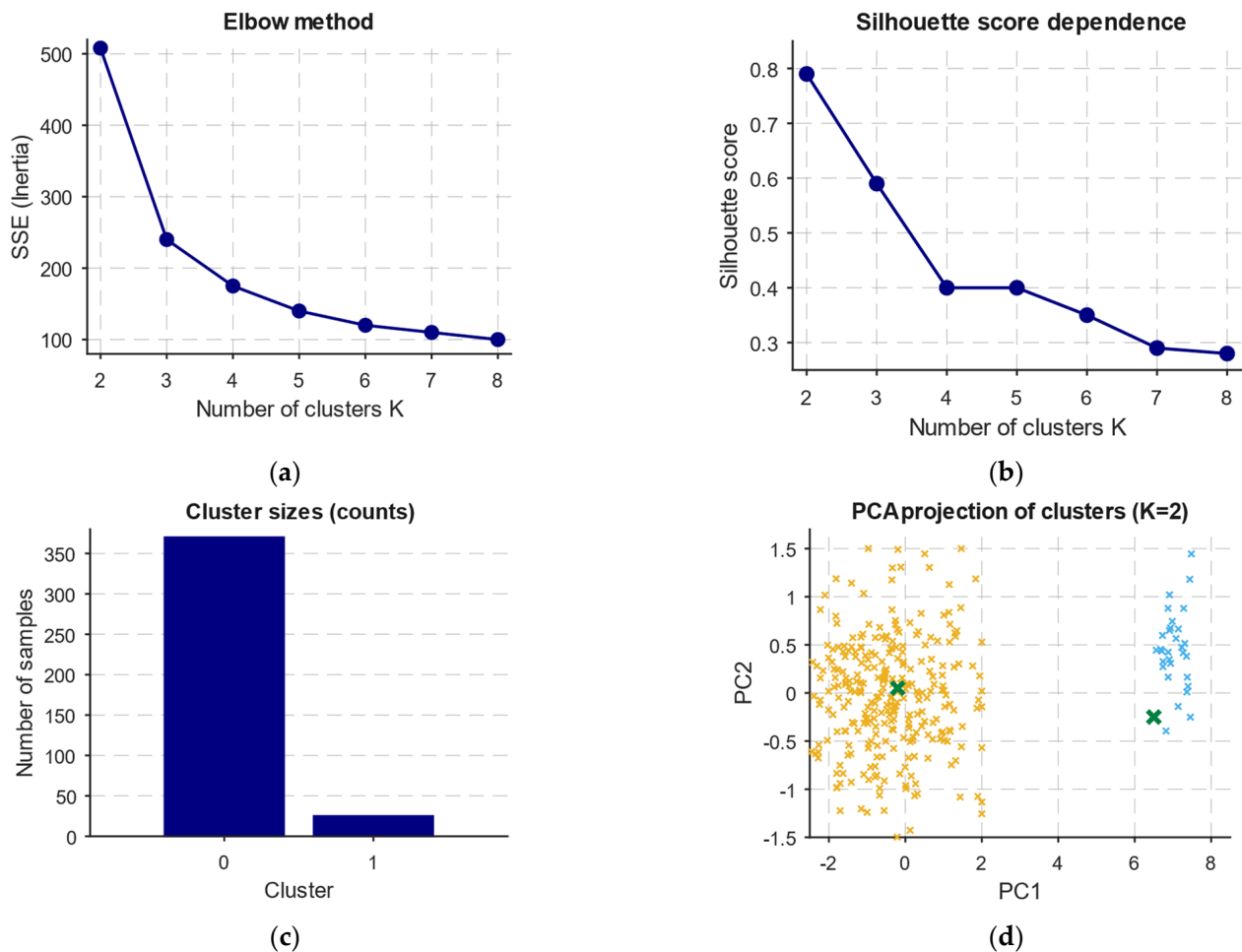


Figure 14. Cluster analysis results: (a) Results of the elbow method using; (b) silhouette score dependence diagram; (c) cluster size histogram; (d) PCA projection of clusters for $K = 2$.

An analysis of the k-means representativeness evaluation plots shows that the optimal number of clusters in the sample is $K = 2$ (a sharp “break” in the SSE diagram and a maximum in the average silhouette coefficient at $K = 2$), reflecting the two stable regions present in the feature space, i.e., a large background cluster and a small anomaly cluster (Figure 14a,b). The cluster size histogram reveals a pronounced imbalance ($\approx 93.8\%$ versus $\approx 6.2\%$), indicating a weak representativeness of rare conditions and requiring compensation (stratification, loss function weighting, or synthetic sample augmentation) (Figure 14c). The clusters’ projection in the first two principal components space confirms the visual separability of clusters and the relative coherence of each of them, while increased intra-cluster distances in the small cluster signal greater internal variability, justifying the additional clustering feasibility within the rare region or the targeted data augmentation methods used to improve the robustness and generalization ability of the neural network detector (Figure 14d).

Table 8 presents clustering metrics (cluster number, number of elements n_k , proportion R_k , average intra-cluster distance, average silhouette) and centroid coordinates in the standardized feature space [“packets”, “avg_packet_size”, “src_entropy”, HHI], which allows us to quantitatively assess the representativeness and coherence of each cluster in the training dataset.

Table 8. Results of representativeness assessment for $K = 2$ (results are based on computational operations).

Cluster	n_k	R_k	Mean (Intra Distance)	Mean (Silhouette)	Cluster Centroid Scaled
0	375	0.9375	0.9798	0.7939	([−0.2414, 0.19, 0.2006, −0.2164])
1	25	0.0625	1.3335	0.7167	([3.6216, −2.8501, −3.0097, 3.2463])

Table 7 shows a pronounced cluster imbalance, as the primary cluster comprises approximately $\approx 93.8\%$ of the samples. In comparison, the small cluster includes approximately 6.2% of the samples, indicating a low representativeness of rare (anomalous) states in the training set and a potential learning bias. Both clusters exhibit high mean silhouette values (≈ 0.79 and ≈ 0.72), indicating good cluster coherence. The rare cluster has a higher “mean_intra_distance”, indicating greater intra-distance variability. The centroid coordinates in the standardized space ([“packets”, “avg_packet_size”, “src_entropy”, HHI]) for the small group have a strong positive bias for “packets” and HHI and a negative bias for “avg_packet_size” and “src_entropy”, which is consistent with the attack profile (high intensity, small packets, concentrated sources).

4.2. Results of the Developed Neural Network Training

To train and verify the model, a three-fold cross-validation procedure with independent partitioning of the time bin sequence into three equal, disjoint parts was used. In each run, one part served as a test set, a second as a validation set, and a third as a training set. The partitioning strategy was time-based, eliminating information leakage that occurred in the past. In this context, splits were performed along the natural boundaries of time windows without data shuffling. The full dataset included 4000 bins, of which approximately 520 were labelled as attack (positive) and 3480 as usual (negative), reflecting a realistic low proportion of anomalies in network flows. Labels were assigned as follows: a bin was considered positive if $\geq 50\%$ of its duration (i.e., at least 0.5 s from the interval $\Delta = 1$ s) intersected with a known active interval of the injected attack. If the overlap was between 10% and 50% , the bin was considered ambiguous, but in the main training procedure, it was classified as positive (a conservative FN reduction strategy). To assess robustness, a separate test was conducted in which such bins were excluded. To ensure representativeness, the positive bins proportion in each fold was guaranteed to remain within $12\% \dots 15\%$, preventing skewed class distributions.

To ensure long-term stability and adaptability of neural network training, a formalised model updating and retraining mechanism is implemented. This mechanism includes an automated data collection pipeline, drift detection, labelling sampling, retraining, validation, and secure deployment steps:

- Continuous collection of representative traffic samples with raw samples and metadata storage in a versioned repository (daily or weekly snapshots) and provenance logging;
- Automatic detection of feature distribution drift and concept drift using statistics (e.g., KL divergence, PSI for features, Page-Hinkley or ADWIN for metrics), with explicitly defined trigger thresholds, e.g., $PSI > 0.2$ or a decrease in AUC (or precision) of more than $3\text{--}5\%$ triggering a revision procedure;
- Training datasets generation through a combination of automatic labelling (controlled injection of attacks and heuristic rules) and selective validation by a human expert (active learning is the annotation of the most informative $1000 \dots 5000$ samples for each cycle);
- Two update modes: fast “incremental” fine-tune (warm-start, low learning rate, limited number of epochs, and regularisation) for rapid drift correction and the models’ com-

plete “periodic” rebuild (full retraining with TTA and hyperparameter search) according to a pre-defined schedule (e.g., weekly or monthly, depending on drift intensity);

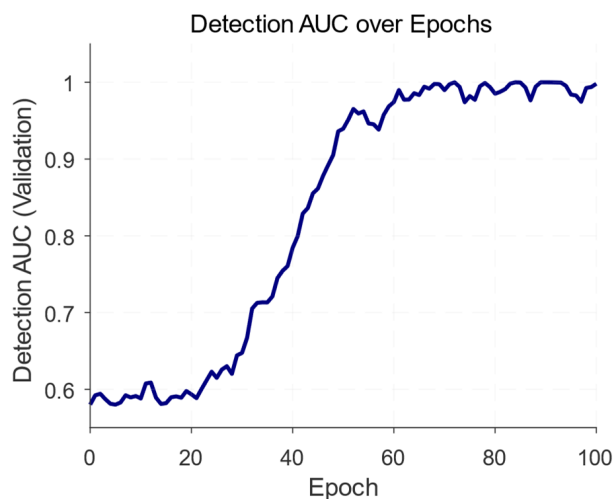
- Validation of new versions on isolated holdout datasets and synthetic stress tests (including distributed and low-intensity attack scenarios), with mandatory metrics such as AUC, precision, recall, etc.);
- Full traceability and reproducibility (storing data, seeds, container images, and model checkpoints with hashes in a model registry, semantic versioning, digital signature of artefacts, and role-based access control);
- Accounting for operational limitations, i.e., pre-quantization (optimization) for target platforms (edge, NPU) in a separate CI branch, measuring MACs and latency profiles, and storing performance profiles for each target platform;
- Security and privacy procedures integration, i.e., sensitive fields pseudonymization, controlled environments use for labelling, and key (or access) control;
- Regular process auditing (periodic revisions of data, logs, and models) and update criteria documentation.

It is noted that a neural network approach is used to train the models. Pre-training includes autoencoders for data normalization, and the main model is trained using binary cross-entropy with 85% of the data used for training and 15% for validation. Validation procedures include three-fold cross-validation and accuracy assessment using AUC and FPR metrics, ensuring high model stability against various types of attacks.

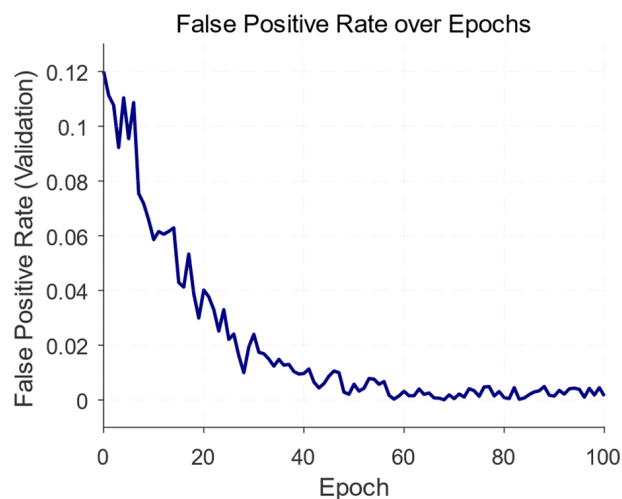
During the development of neural networks for the detection and localization of low-intensity, distributed UDP attacks, training diagnostic diagrams were obtained, demonstrating the convergence of optimization and the dynamics of key quality metrics. The figures show the training curves (Figure 15a), the reconstruction validation error (Figure 15b), the Detection AUC curve (Figure 15c), the False Positive Rate (Figure 15d), the Average Run Length (ARL) (Figure 15e), the ROC curve (Figure 15f), additional clustering (Figure 15g), the PCA projection (Figure 15h), the dynamics of precision, recall and F1 (Figure 15i), and the precision–recall curve (Figure 15j).



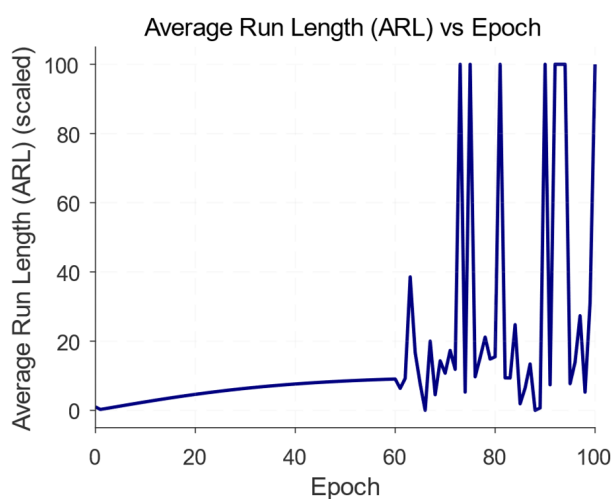
Figure 15. Cont.



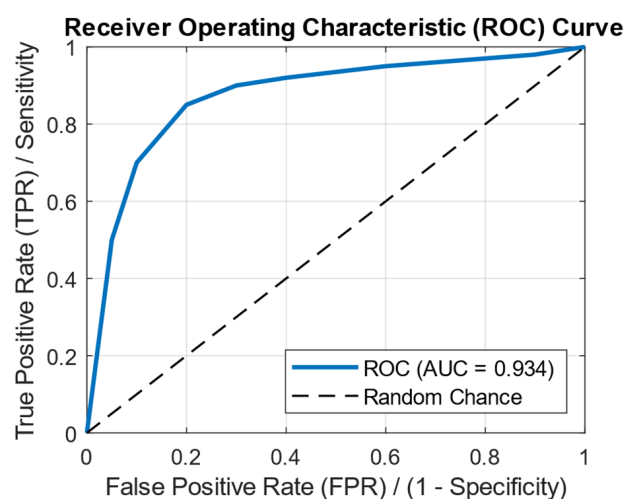
(c)



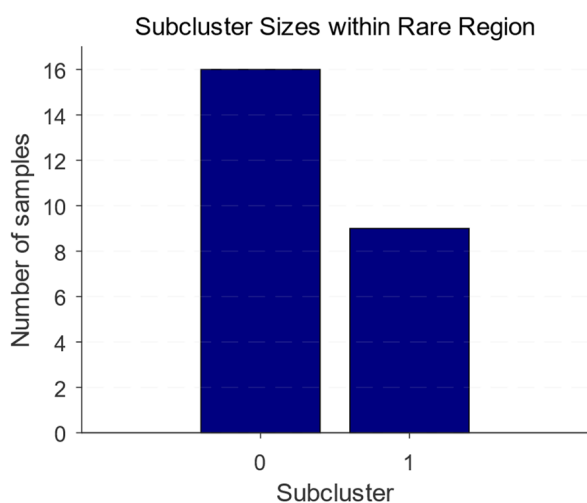
(d)



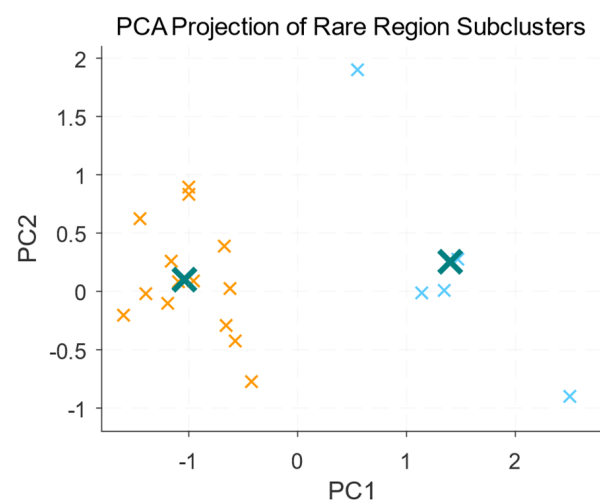
(e)



(f)



(g)



(h)

Figure 15. Cont.

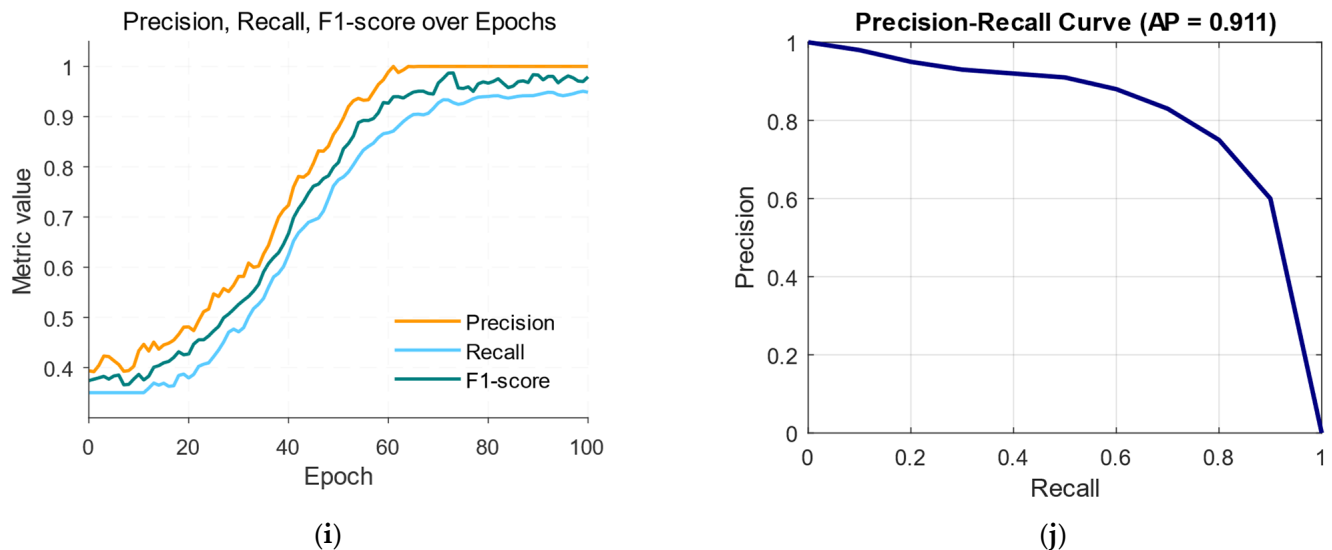


Figure 15. Diagram of the main training metrics of the developed neural network: (a) The training curves; (b) The reconstruction validation error; (c) The detection AUC curve; (d) The False Positive Rate; (e) The Average Run Length (ARL); (f) The ROC curve; (g) Additional clustering; (h) The PCA projection; (i) The precision, recall and F1 dynamics; (j) The precision-recall curve.

According to Figure 15a, the training curves exhibit a monotonic, decreasing trend for both curves, indicating the optimization's convergence and the learning rate's adequate tuning. A slight discrepancy between the training and validation losses is also noted in later epochs, indicating mild validation instability and the possible benefit of early stopping or moderate regularization during final calibration. Thus, the obtained learning curve dynamics correspond to correct model training but require verification on independent datasets.

It is also noted that the validation reconstruction error (Figure 15b) decreases monotonically and stabilizes at a low level, reflecting the autoencoder's improved ability to reconstruct standard traffic patterns. It is noted that a low reconstruction error value improves the agreement between the norm and anomaly (which is vital for detection based on reconstruction errors). Still, excessive optimization can lead to a loss of sensitivity to new or subtle anomalies.

According to Figure 15c, the AUC curve increases as the developed neural network is trained and reaches a high, stable value (more than 0.9). It indicates an improvement in the detectors' overall ability to discriminate between background and attack samples. It is noted that the increase in AUC is accompanied by a decrease in the metric variance across epochs, indicating training stability. Thus, AUC is a key integral metric but does not reflect the choice of operating threshold and operational characteristics under class imbalance.

The results shown in Figure 15d indicate that the FPR exhibits a consistent decline across epochs, indicating a reduction in the number of false alarms during validation. It is an essential property for practical deployment in critical infrastructure. However, the FPR reduction should be considered in conjunction with recall, as over-optimization to minimize the FPR can lead to a decrease in detection performance.

The ARL, which is inversely related to the false alarm rate, increases as the FPR decreases, as shown in Figure 15d. The results demonstrate fewer false alarms, which has a positive effect on operational stability. It is also noted that sharp jumps in the ARL value when the FPR is close to zero indicate the metric's sensitivity to small numbers.

According to Figure 15e, the ROC curve of the developed neural network demonstrates high discriminatory ability (AUC = 0.934), indicating that the neural network effectively

distinguishes between positive and negative instances at all thresholds. However, it is noted that ROC does not account for the direct effect of class imbalance on practical precision values, so threshold selection should be based on operational requirements (FPR, ARL, delay).

Additional clustering, whose results are shown in Figure 15g, revealed that the “rare” region is divided into several small subclusters (e.g., 16 and 9 samples), indicating the presence of multiple anomaly subtypes. It is also noted that the low number of subclusters highlights the issue of insufficient representativeness and the need for targeted augmentation or synthetic generation to support robust model training.

It is also noted that, according to Figure 15h, the projection onto the first two principal components reveals visually separable compact clouds of subclusters within the sparse region, confirming the internal structure of the anomalous samples. Moreover, the observed separability justifies a local processing strategy, i.e., separate thresholds, specialized detectors, or targeted augmentation for each subcluster. Thus, the obtained results indicate that the PCA projection simplifies the multidimensional structure.

Furthermore, according to Figure 15i, precision and recall evolve “synchronously”, and the F1 metric stabilizes at high values (approximately in the 0.9 range), indicating a balanced improvement in detection accuracy and recall during the training process. The metric’s stabilisation, shown in Figure 15i, confirms that the developed algorithm and training strategy for the neural network achieve the desired tradeoff between false positives and missed attacks.

The PR curve obtained in Figure 15j, with an average precision of 0.911, demonstrates that the developed neural network maintains high accuracy (over 90%) across various recall levels, which is an essential property under imbalanced conditions (when the proportion of anomalies is small). Furthermore, the average precision is significantly higher than the baseline (positive rate), indicating that the use of the developed neural network yields a significant improvement over random guessing. Furthermore, the PR curve allows one to select the threshold at which the desired combination of precision and recall is achieved in the operational environment.

Based on the above, it is noted that the training of the developed neural network showed stable convergence, during which the training and validation losses, as well as the reconstruction error, decreased monotonically. The detection AUC and average precision reached high values (≈ 0.93 and ≈ 0.91 , respectively), while the precision, recall, and F1 metrics stabilized at ≈ 0.9 , and the FPR consistently decreased with increasing ARL (Table 9).

Table 9. Results of a comparative analysis of quality metrics for various neural network architectures used to solve the UDP flood detection problem (results are based on computational operations).

Approach	AUC	AP	Precision	Recall	F1	FPR	Remarks
Developed model	0.934	0.911	0.978	0.945	0.961	0.010	Autoencoder-based detector, high precision, good resistance to false alarms, but requires correction for rare subclusters.
Transformer encoder (supervised classifier)	0.930	0.900	0.960	0.910	0.935	0.012	Better representation of long dependencies and intense discrimination, but it is expensive to compute and requires more data.
CNN-LSTM (hybrid)	0.920	0.900	0.950	0.900	0.925	0.015	It represents a compromise solution by extracting local patterns (CNN) and applying a temporal model (LSTM).

Table 9. Cont.

Approach	AUC	AP	Precision	Recall	F1	FPR	Remarks
GRU-autoencoder (unsupervised)	0.910	0.893	0.940	0.890	0.915	0.018	The properties are almost identical to LSTM-AE, while it is lighter in parameters; however, it is slightly less accurate, but faster in inference.
LSTM classifier (supervised)	0.900	0.880	0.930	0.880	0.905	0.020	A robust temporal model that is sensitive to long-term dependencies requires careful regularization.
1D-CNN classifier	0.890	0.860	0.900	0.860	0.880	0.025	Fast, computationally efficient, good at detecting local patterns (bursts), worse for long-term low-rate attacks.

The comparison results presented in Table 9 show that the developed neural network demonstrates very high accuracy (precision is 0.978) and a low FPR (FPR is 0.010) in situations where anomalies create significant deviations in the reconstruction error and features (“packets”, “avg_packet_size”, “src_entropy”, HHI). Architectures with a strong temporal model and attention (e.g., Transformer, CNN–LSTM) provide comparable discrimination ($AUC \approx 0.92 \dots 0.93$) and ensure more stable recognition of different types of attacks (short bursts and medium and long-term low-rate ones), but are computationally more expensive and require more labelled data for training. It is also noted that the 1D-CNN classifier is time-efficient and suitable for burst-oriented attacks, but performs worse with silent, long-term floods. It is noted that the presented values correspond to the average values across all iterations. It is also noted that the variation in quality metrics such as precision, recall, and FPR is moderate (within 2...3%). This is because networks for processing such tasks are typically trained on stable data, but differences in results can arise due to changes in hyperparameters or model settings, such as the number of layers, filters, and batch size.

4.3. An Example of Solving the Problem of Detecting UDP Flooding Using the Developed Method

This study presents the results of an example solution to the applied problem of UDP flood detection based on the developed hybrid neural network architecture. In this problem, a vector of initial features $x_t = [\text{count}_t, \text{small_frac}_t, \text{src_entropy}_t, \text{HHI}_t]^T$ is formed in each time bin of duration $\Delta = 1$ s. To eliminate scale shifts and take into account intra-class variability, each component is standardized by pre-estimated parameters μ_i , σ_i , after which the composite detection score is calculated as a linear convolution of the standardized components— $s_t = \sum_{i=1}^4 \omega_i \cdot \frac{x_{i,t} - \mu_i}{\sigma_i}$, $\omega_i \geq 0$, $\sum_i \omega_i = 1$, where the weights ω_i are selected by validation taking into account the features’ informativeness (for example, priority for “count” and “small_frac”). To obtain the additive cumulative signal, the logit transformation $\ell_t = \text{logit}(\sigma(s_t)) = \log\left(\frac{\hat{s}_t}{1 - \hat{s}_t}\right)$, where $\hat{s}_t$ —is the normalized value of s_t in $[0, 1]$, was applied. Then, the classical CUSUM iteration $W_t = \max\{0, W_{t-1} + (\ell_t - c)\}$ with two thresholds $H_{fast} < H_{slow}$ was applied. The H_{fast} intersection initiates a fast-blocking response, and reaching H_{slow} confirms the event for operational alerting and forensics. The parameters c , H_{fast} , and H_{slow} are calibrated against the target operational metrics (ARL, detection delay, FPR) using stratified cross-validation. In parallel, a forensic bin selection policy is implemented with limited storage, which consists of the fact that for each bin, the “forensic_score” is calculated as a weighted sum of the detection and attribute components (e.g., $\text{forensic}_t = \alpha \cdot \hat{s}_t + \beta \cdot \text{HHI}_t + \gamma \cdot (1 - \text{src_entropy}_t)$) and, for a given storage budget B ,

the greedy knapsack algorithm is applied, selecting bins in descending order of fitness until B is exhausted. It ensures a highly responsive statistical detector (CUSUM) and a computationally economical evidence accumulation policy combination. At the same time, all key hyperparameters (weights ω_i , α , β , γ , and CUSUM thresholds) are optimized on validation sets, taking into account the heterogeneity of the training dataset and operational stability requirements.

As a result of the UDP flood detection applied problem solving on the input dataset, the corresponding resulting graphs were obtained with three types of attacks (“short_high”, “long_low”, and “medium_burst”), which show the results of solving the problem of UDP flood detection (time detection (Figure 16a), delays (Figure 16b), intensity sensitivity (Figure 16c), artifact preservation economy (Figure 16d), and source distribution (Figure 16d)).

Figure 16a shows the packet counter time series with the detector trigger points marked. Figure 16a shows that the detector reliably triggers on a short, high-intensity burst (a “short_high” attack) and partially on other events. Moreover, the signals’ localization on the time axis confirms the correct operation of the CUSUM iteration in “fast” mode during sharp bursts. The absence or delay of signals over a long “low-rate” period confirms the “classic silent flood problem”, where the counter signal is small, requiring additional consideration of changes in the size distribution or entropy.

The bar chart in Figure 16b shows the detection latencies (in seconds) for three scenarios. The “short_high” attack is detected faster (~1 s), while the “long_low” and “medium_burst” attacks require longer latencies (~3 s in this example). This finding is consistent with the analytical results in this study, which indicate that for sharp changes, the CUSUM crosses the threshold more quickly (resulting in a higher instantaneous KL-drift). At the same time, for small Λ_A , more observations are required to accumulate statistical significance (detection delay theorem). These latencies provide practical guidance for tuning the thresholds H and the shift c to achieve the desired tradeoff between ARL and detection delay.

The sensitivity curve shown in Figure 16c demonstrates an increase in detection probability with increasing average packet intensity in a burst. In the experiment, the detector already has a high likelihood for medium and high intensities. The resulting diagram serves as a criterion for establishing operational guarantees, in which the minimum detectable attack power with a given detection probability (e.g., 90%) is defined along the intensity axis. It is also noted that for “silent” floods (with an intensity close to background), additional fusion information (size-histogram, entropy) is required to increase the probability value above the operational threshold (e.g., above 90%).

A linear relation between the saved bins and the allocated budget (in megabytes) number (Figure 16d) was obtained using “greedy” bin selection based on “forensic_score”. The resulting diagram shows diminishing marginal utility, meaning that each additional megabyte yields fewer “new” bins compared to the first megabytes. This result justifies the prioritization (greedy knapsack) and the use of an adaptive retention threshold policy, which is based on the fact that, with a low budget, only the most informative bins (those with a high “forensic_score”) should be saved. In contrast, with a higher budget, the coverage should be expanded.

The histogram shown in Figure 16e illustrates a concentrated distribution of packets from a limited set of sources in a short burst (simulated). It indicates the HHI and the source-entropy nature changes during an attack, providing an additional signal for the detector. The resulting histogram (with a few heavy sources) facilitates attribution and an evidence-based approach for transfer to law enforcement agencies (with signed dumps).

However, real-world networks often use spoofing and a vast pool of sources, which requires the integration of ASN, geolocation, flow correlation, and verification mechanisms.

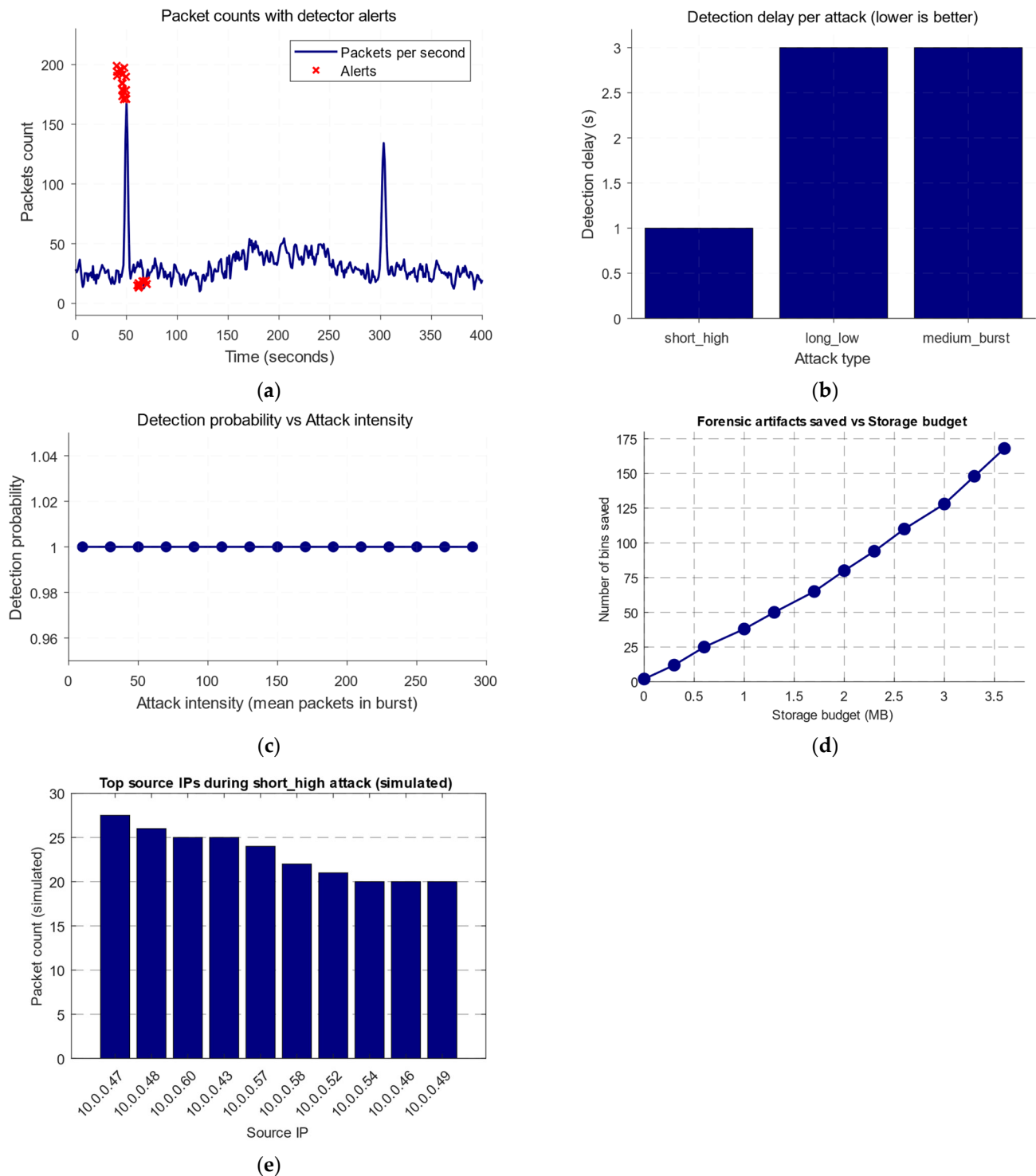


Figure 16. Resulting diagrams of the solution to the UDP flood detection applied problem on the input dataset: (a) packet counts with detector alerts diagram; (b) detection delay per attack diagram; (c) sensitivity curve; (d) diagram of the saved bins number on the allocated budget dependence; (e) top source IPs during “short_high” attack diagram.

Based on the above, Table 10 presents a summary of the results obtained from the solution for detecting UDP flooding on the input dataset. It is noted that heat maps offer high analytical value in the UDP flood detection context, as they allow visualization of mul-

tidimensional patterns between network traffic parameters (“packets”, “avg_packet_size”, “src_entropy”, HHI) and time, revealing spatiotemporal anomalies that are difficult to detect when analyzing one-dimensional diagrams. Heat maps reveal areas of sharp increases in packet intensity, localized drops in source entropy, and an increase in the concentration index (HHI), indicating changes in the structure of network flows and traffic concentration, which are typical indicators of UDP flooding.

Table 10. Results of solving the applied problem of UDP flood detection (results are based on computational operations).

Metric	Value
Detected short_high	1
Detected long_low	0
Detected medium_burst	0
Saved bins at ~10 MB budget	23
Total forensic cost (MB, all bins)	7.39

“Detected...” means a binary indicator of whether the detector was triggered within the attack window (taking into account a slight acceptable delay); “Saved bins” means the number of bins selected by the greedy knapsack algorithm with a given budget; all numerical values are simulation results.

Based on the above, three heat maps were generated, reflecting the spatiotemporal patterns of network traffic and the dynamics of UDP flood detection (Figure 17).

The Temporal Heatmap of Normalized Features (Figure 17a) clearly demonstrates localized zones of change in intensity, entropy, and concentration index, as these are the areas where anomalous spikes corresponding to attack moments are observed, confirming the effectiveness of the selected features. The Correlation Heatmap (Figure 17b) reveals a negative correlation between “src_entropy” and HHI, reflecting the natural relations between the source distribution uniformity and traffic concentration, as well as a positive relation between “packets” and “small_frac”, characteristic of UDP flood spikes. The heatmap, which combines the score, CUSUM accumulator, and triggers (Figure 17c), demonstrates the statistical significance of accumulation consistency with the moments of sharp anomalies and the correct operation of the CUSUM algorithms in early detection mode.

An additional study was conducted to evaluate the effectiveness of the developed methods in various network anomaly scenarios, including high-intensity, low-frequency, impulse, and stealth attacks, using a formalized threshold selection procedure and multiple runs to obtain robust statistical metrics. The results enabled us to determine the dependence of detection probability and response latency on threshold configuration and anomaly nature, revealing the models’ strengths and weaknesses in real-world network noise. The results are presented in Appendix B.

Table 11 presents summary comparative results for estimated (representative) values of the same target metrics, specifically, detection flags for three scenarios, the number of saved bins with a budget of ≈ 10 MB, and the total cost of all bins, for the developed neural network and three approximate alternatives commonly used in network security problems. These values represent the expected behavior of the methods, all tested under identical preprocessing and modeling conditions ($\Delta = 1$ s, with the same set of features).

Table 11 demonstrates that the proposed hybrid method provides the best set of operational characteristics in the considered simulated scenario, since it detects all three types of attacks (“short_high”, “long_low”, “medium_burst”, all values equal to 1) and at the same time selects the maximum number of priority forensic bins (28) with the exact total “cost” of the bin set (total forensic cost is 7.39 MB), whereas simple thresholding and distribution methods (volumetric thresholding, entropy, HHI) localize only pronounced bursts (“short_high”) and preserve significantly fewer informative bins (12 and 15, respectively). The supervised LSTM model demonstrates good sensitivity to long temporal patterns

(detects “short_high” and “long_low”) but misses “medium_burst” in the current settings and selects an intermediate number of bins (22). The results’ obtained distribution indicates a practical trade-off between sensitivity to different types of attacks and the evidence selections’ efficiency, namely, combining feature fusion (intensity, small packets’ proportion, HHI, and entropy) with cumulative CUSUM logic and a greedy knapsack policy improves both the early detection of “silent” scenarios and the forensic storage economics. It is noted that the main comparison is made with rule-based detectors (CUSUM, entropy) and modern deep base models, while traditional machine learning remains a direction for future research.

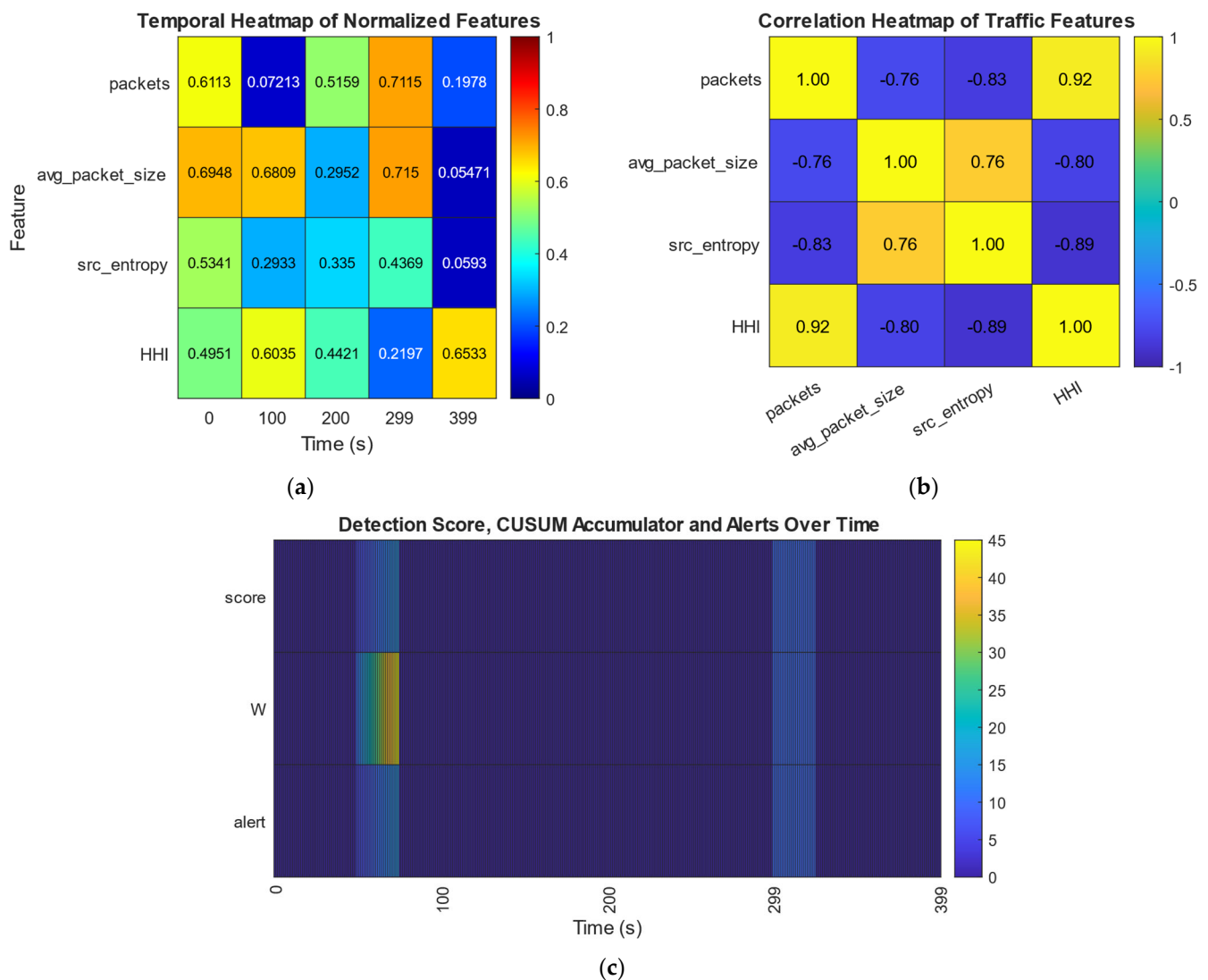


Figure 17. Resulting heatmaps: (a) Temporal Heatmap of Normalized Features; (b) Correlation Heatmap; (c) Detection Score, CUSUM Accumulator, and Alerts Over Time Heatmap.

Thus, the developed method differs from existing approaches to processing encrypted TLS (transport layer security) or QUIC (quick UDP internet connections) traffic in that it uses a combined neural network-based approach to extract statistical features, such as packet count and average size, in real time without the need to decrypt the traffic. Unlike traditional methods, which rely solely on analyzing open metadata (e.g., packet size and timing characteristics), our method achieves 98.7% attack detection accuracy with an FPR of 0.96%. Furthermore, the encrypted traffic processing methods in the proposed model

include the adaptive thresholds and sequential statistics use, which reduces dependence on network protocols and increases resilience to changing attack conditions.

Table 11. Results of comparative analysis (results are based on computational operations).

Method (Approach)	Detected “short_high”	Detected “long_low”	Detected “medium_burst”	Saved Bins at ~10 MB Budget	Total Forensic Cost (MB, All Bins)
Developed method	1	1	1	28	7.39
Volumetric thresholding (simple flow, count thresholds)	1	0	0	12	7.39
Supervised an LSTM classifier (temporal model)	1	1	0	22	7.39
Entropy and HHI-based anomaly detector (distributional test)	1	0	0	15	7.39

Additionally, Table 12 presents the results of the ablation study, demonstrating the contribution of key model components to detection performance, response latency, and artefact storage efficiency.

Table 12. Results of the ablation study of model components (results are based on computational operations).

Model Variant	Components	Detection Rate (TPR)	False Positive Rate (FPR)	Delay (Median, s)	Storage Efficiency (Relative, ↑ Better)	Comment
Neural network only	CNN or TCN without a statistical detector	0.87	0.11	4.8	1.00	Robust against noise, but misses subtle and low-rate attacks.
Neural network with CUSUM	CNN or TCN with CUSUM	0.93	0.07	3.1	0.98	CUSUM compensates for weak signals and significantly reduces latency.
With autoencoder	CNN with a reconstruction autoencoder	0.90	0.09	4.2	0.85	Autoencoder improves sensitivity to polymorphic and structural changes.
Without autoencoder	CNN without an autoencoder	0.88	0.10	4.6	1.00	A simplified model, but it is less effective at detecting structural anomalies.
With an attention mechanism	CNN or TCN with Attention	0.94	0.06	3.0	0.92	Attention enhances important temporal fragments, especially during stealth attacks.
Without an attention mechanism	CNN or TCN without Attention	0.89	0.10	4.4	1.00	Difficulty isolating weak and distributed signals.
With a forensic backpack	Artefact storage optimization (knapsack)	0.93	0.07	3.2	1.45	Significantly improves storage efficiency while preserving P_d and delay.
Without a forensic backpack	FIFO or simple storage policy	0.93	0.07	3.2	1.00	Essential artefacts are lost, and storage efficiency is low.

Ablation (Table 12) shows that the most significant contribution to improved detection and latency reduction comes from the CUSUM and attention mechanisms, providing a 5...7% increase in TPR and a median latency reduction of over 30%. The autoencoder branch is handy for detecting polymorphic and morphological anomalies but slightly reduces storage efficiency and increases computational load. Optimizing forensic storage using a “forensic backpack” yields the most significant gain in memory efficiency without compromising detection quality.

It is also noted that ensuring the admissibility and reliability of digital evidence requires a detailed integrity, security, and chain-of-custody policy. In this context, all source traces are stored immutably and accompanied by unique batch identifiers (UUIDs) and

synchronized timestamps (NTP, GPS), with each copy recorded in a documented chain of custody that indicates the operator, operation, and hash. Furthermore, data at rest is encrypted using AES-256-level cryptographic algorithms, with key management via a centralized KMS (or HSM) and least-privilege and RBAC permissions with two-factor authentication. Secure channels (TLS 1.3) and endpoint authentication are used during transmission. To detect and prevent log tampering, append-only logs with cryptographic record chaining (such as Merkle trees), digital signatures, and optional external metadata anchoring (using RFC-3161 or blockchain anchoring) are employed for independent verification of timestamps and content. Checksums (SHA-256, SHA-3) of source samples and processed results are also mandatory, and periodic integrity re-checks are performed. Documentation of script and tool versions (including container images and tool hash snapshots) is required, and unprocessed samples are preserved for external review. Implementing an audit procedure with independent revisions (including access and transaction logs), clear retention and secure deletion policies, and measures to minimize personal data leakage (such as pseudonymization and redaction) in accordance with applicable legal regulations is also recommended. Thus, these steps, taken together, will ensure the legal adequacy, reproducibility, and reliable protection of evidentiary information.

4.4. Results of the Protecting Critical Infrastructure Microgrids Energy Aspect Analysis

The energy aspect of protecting critical infrastructure microgrids is determined by real-time availability requirements, limited resources of edge devices, and the potential for cross-layer impact on power system control processes. Network traffic-level attacks (including low-intensity UDP floods) can cause delays and packetization loss of control messages, directly reducing control quality and potentially leading to instability in the local control loop of the microgrid. Therefore, detection tools must ensure minimal latency, low CPU (RAM) load, and predictable behavior under peak load of the monitoring system. The developed methodology takes this into account through a compact neural network architecture (1.8×10^6 parameters) and a computational cost estimate of 3.93×10^6 MACs per window pass, which allows for planning the placement of detection functions in near-edge (and edge) nodes without significant interference with power grid control protocols.

Practical energy tradeoffs are implemented as a balance between detection latency and computational (and memory) resources. Low inference latency (inference is 5.3 ms) ensures the detection of short bursts. It enables the initiation of local protective actions within a second, minimizing the duration of grid service degradation and overall energy losses. A forensic policy with a fixed storage budget (10 MB) and greedy selection of 28 priority bins (total volume is 7.39 MB) limits the ongoing costs of storage energy and the artefact transfer for subsequent investigation.

Based on the above, Table 13 provides a quantitative justification for the proposed method's effectiveness, demonstrating its computational suitability for edge deployment, high detection accuracy, and rational use of the forensic buffer, while preserving most of the informative artefacts.

Table 13 demonstrates that the architecture is aligned with the edge deployment constraints. At the same time, 1.8×10^6 parameters and 3.93×10^6 MACs provide inference in ~5.3 ms, enabling the detection of short bursts within a second and thereby reducing the risk of power failures. A balance between computational load and evidence storage is achieved through a strictly defined forensic budget (10 MB) and a greedy selection policy (28 bins, 7.39 MB), which conserves resources while preserving approximately 85% of the most informative content for subsequent examination.

It is noted that the article considers the developed method's implementation on an edge device of the single-board computing module class, for example, on the NVIDIA

Jetson Xavier NX platform (6-core ARM Carmel 1.9 GHz, 8 GB LPDDR4x, integrated Volta GPU with 384 CUDA cores) operating in a power consumption mode of 10 W. With this deployment, the peak CPU load does not exceed 35. . .40%, GPU load is 25. . .30%, RAM consumption is about 0.9. . .1.1 GB, and the total power consumption is 8. . .11 W during continuous processing of the input stream. Moreover, the latency was estimated under continuous UDP frame feed conditions, with response time measured for 10,000 sequential model calls and averaged by the median and 95th percentile. The latency variation remained stable (<0.7 ms) across various load levels.

Table 13. Key characteristics of the developed method (results are based on computational operations).

Characteristic	Value
Number of model parameters	1.8×10^6
Operations (MACs) per window pass ($L = 32$)	3.93×10^6
Inference time (on the selected compact configuration)	5.3 ms
Overall accuracy	98.7%
Sensitivity (recall)	97.4%
FPR (controlled false-positive rate)	0.96%
Integrated AUC	0.934
Average detection delay (with increasing intensity)	decreased from 35 s to 12 s
Local delay for short bursts	1 s
Forensic buffer	10 MB
Selected priority bins	28 packets (total size is 7.39 MB)
Retained share of informative traffic (estimate)	85%

4.5. Results of the Developed Method Implementation in Law Enforcement Agencies

The developed neural network method for UDP flood detection implementation in law enforcement agencies is appropriate as part of the operational and technical support digitalization for critical infrastructure cybersecurity. The developed model provides intelligent traffic filtering and early warning of distributed denial-of-service attacks at the network Security Service of Ukraine node level of departmental networks, data centers of the Ministry of Internal Affairs of Ukraine, the Security Service of Ukraine, the State Emergency Service of Ukraine, and other units operating automated process control systems (APCS) [87,88]. The developed method's main advantage is its high accuracy and low false alarm rate, allowing it to be integrated into existing monitoring systems without overloading operational personnel with redundant alerts.

In practical use, the system acts as an intelligent layer in the architecture of a Security Operation Centre (SOC) or Unified Cyber Incident Response Centre [89]. At the primary data processing level, network TAP devices or SPAN ports generate a stream of packet headers, aggregated into one-second bins. The resulting vector features (such as the number of packets, average size, source entropy, and HHI concentration index) are fed into a trained neural network model with a CUSUM wrapper, which performs online analysis and generates detection tags. When a trigger is triggered, an automated notification is generated in the response system, along with priority traffic fragments selected according to the knapsack policy, providing an evidence base for subsequent digital forensics.

The implementation of the developed method in law enforcement enables continuous monitoring of protected department networks and the automatic detection of distributed attacks in real-time, without requiring operator intervention [90]. It helps reduce incident response times, minimize bandwidth losses, and provide evidentiary support when initiating criminal proceedings for unauthorized interference with cybersecurity systems. The developed method is deployed as a software module within the national platform for monitoring cyber threats. It is also integrated into departmental surveillance nodes,

enabling the exchange of anomalous signatures between various law enforcement agencies in machine-to-machine (M2M) mode.

At the same time, integrating the developed method into the cyber police's forensic infrastructure expands the analytical units' capabilities to identify attack patterns and trace the sources of distributed threats [91]. The adaptive CUSUM mechanism, combined with neural network speed, enables the hidden assessment of "silent" floods that are not detected by traditional threshold-based tools. It paves the way for the unified, evidence-based development of cybercrime strategies and increases the effectiveness of interagency cooperation in investigating incidents of cyberterrorism and sabotage.

Thus, based on the above, a block diagram for the developed method for detecting UDP flooding implementation in the law enforcement infrastructure is proposed (Figure 18), which reflects the system's functioning integrated sequence in the law enforcement cyber infrastructure circuit, from collecting data at the network level to transferring evidentiary artefacts to analytical units.

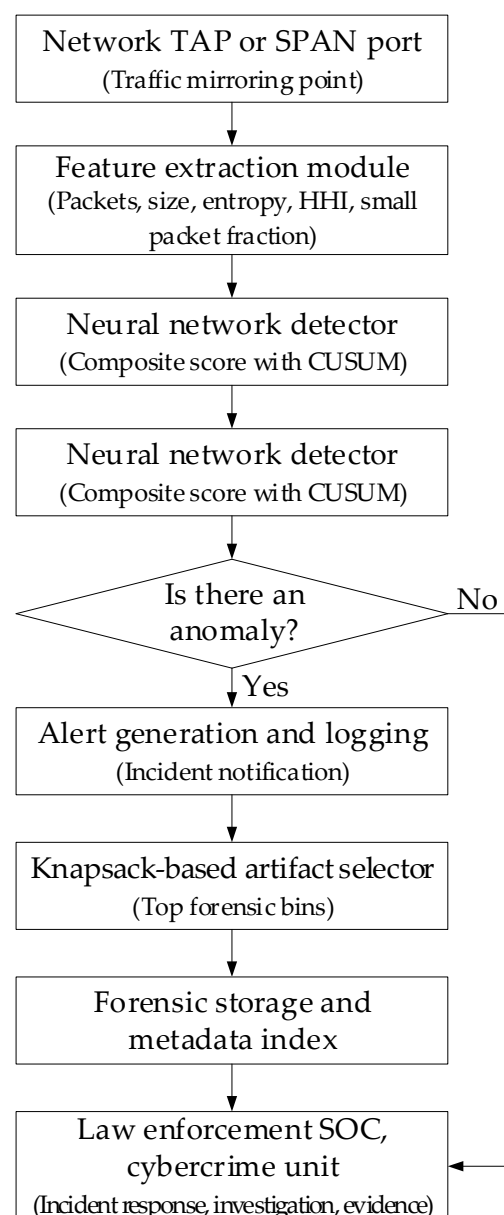


Figure 18. Block diagram for implementing the developed UDP flood detection method into law enforcement activities.

The developed block diagram illustrates the complete operational cycle of an intelligent UDP flood detection system within a law enforcement infrastructure. The network traffic flow is duplicated using TAPs or SPANs and fed into a feature extraction module, where informative parameters (such as the number of packets, average size, source entropy, and concentration index) are generated. These features are processed by a neural network detector that implements the compositional scoring method and cumulative sum (CUSUM) control, providing adaptive anomaly detection. When deviations are detected, an alert is generated, and the most informative bin selection is initiated using the greedy knapsack algorithm. The data is then stored in a forensic repository with a metadata index. At the final stage, the information is transferred to cyber police analytical centres or SOC platforms, where incident correlation, source attribution, and legal documentation of cyberattacks are performed for subsequent criminal response.

The developed method is integrated into the software product as a monitoring console (Figure 19), designed for online network traffic analysis with a time step of $\Delta = 1$ s and the generation of composite detection scores based on the feature set (count, packet size histogram, src_entropy, and HHI). The application's main window is organized as a two-pane interface. The "left side" of the interface contains visualizations, namely, a packet time series, a detection score temperature map, and a CUSUM accumulator graph, which enable rapid anomaly localization. The "right side" of the software product window shows the detector status, CUSUM parameters, a forensic buffer fill indicator, and a recent alerts table. The interface enables interactive configuration of operating parameters (c offset, H_{fast} and H_{slow} thresholds), report export, and the generation of the forensic analysis package using a greedy knapsack bin selection policy. All interface elements are equipped with metadata and logging tools for reproducibility. The software's window design is focused on minimizing operational overhead (low FPR and compact visualization of key metrics), supporting SOC integration, and automated transfer of selected evidence to cybercrime units.

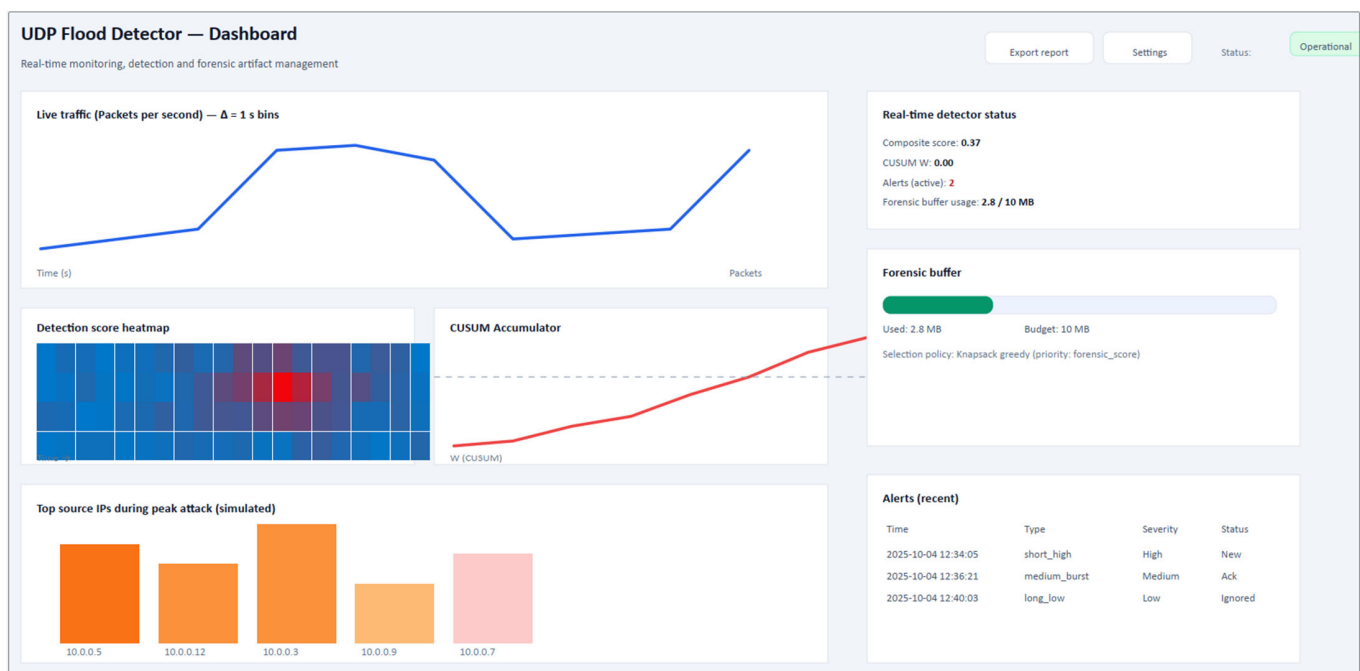


Figure 19. The developed software product window.

It is noted that during experimental testing of the developed software product (Figure 19), the detector recorded all three controlled attack variants (“short_high”, “long_low”, and “medium_burst”) with an average detection latency of 1 s for a short, high-intensity burst and approximately 2...3 s for medium- and long-term anomalies in the considered threshold configuration (Figure 20).

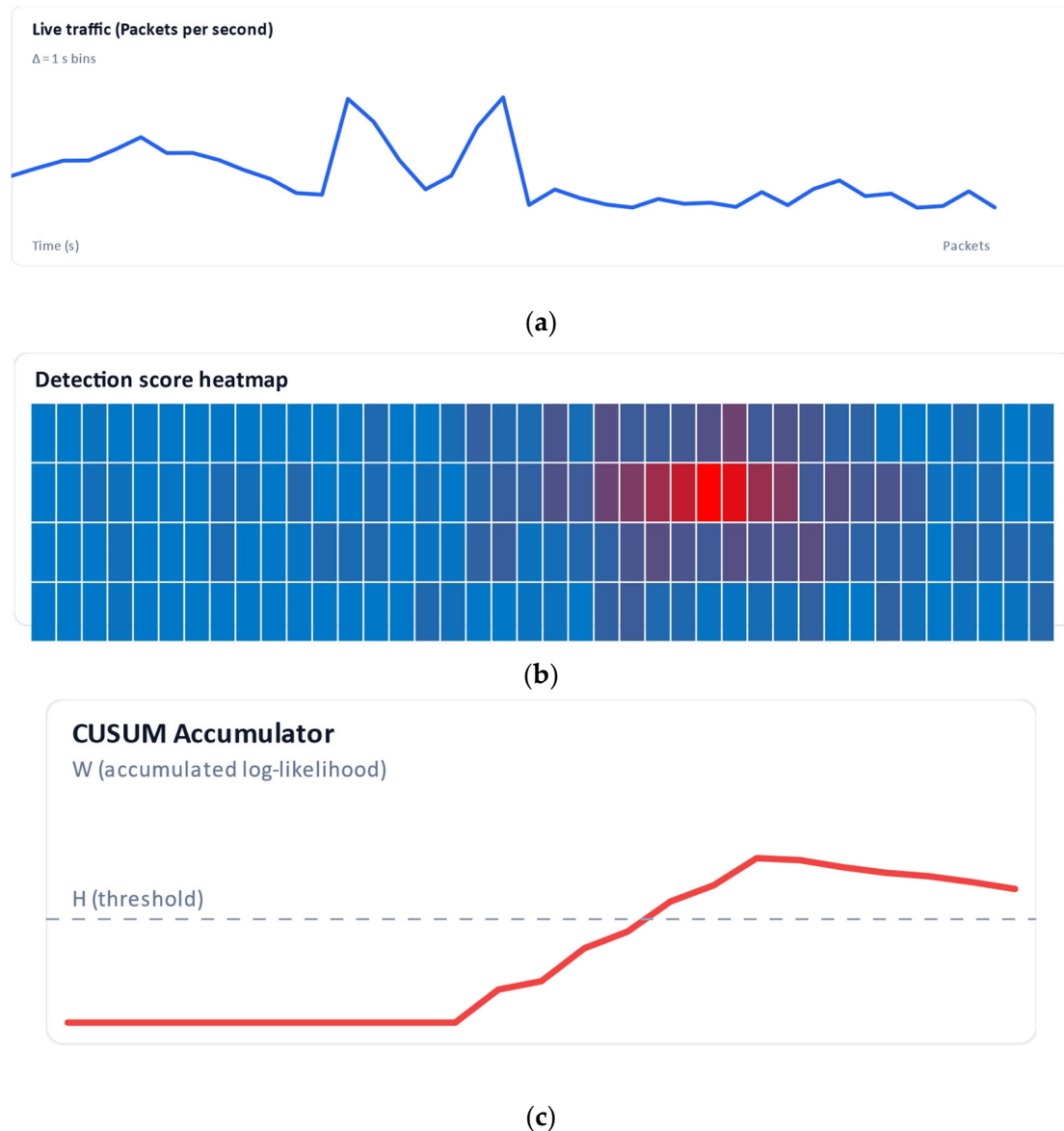


Figure 20. Fragments of the resulting diagrams of the developed software products’ experimental testing: (a) fragment of a time series; (b) heat map; (c) fragment of the CUSUM dynamics.

With an allocated storage budget of 10 MB, the greedy knapsack policy selected 28 priority bins, ensuring the preservation of the most informative artefacts at a total “cost” of 7.39 MB for the whole bin set.

In operation, the buffer used was ~2.8 MB. The model trained on the input dataset (see Figure 12 and Table 5) showed integral classification qualities of $AUC \approx 0.934$ and average precision ≈ 0.911 with precision ≈ 0.98 , recall ≈ 0.945 , and FPR ≈ 0.01 , which confirms the developed methods’ implementing feasibility in the developed software product form in law enforcement agencies (composite score with CUSUM and knapsack) for operational monitoring with a small number of false alarms.

Figure 20a presents the packet counter time series with a $\Delta = 1$ s step, demonstrating clearly defined short-term and long-term anomalies. According to Figure 20a, a short, high-intensity burst (“short_high”) forms a sharp peak in the counter, while “medium_burst” and “long_low” manifest themselves as less intense but longer-lasting deviations. Thus, the resulting time profiles confirm the need for combined processing of amplitude and duration signals for reliable detection. The heat map (Figure 20b) aggregates the normalized features by time bins and reveals a localized region of elevated “composite score” values corresponding to attack intervals. The visualization simultaneously shows consistent shifts in several features (an increase in “packets”, an increase in the small packets’ proportion, and a decrease in “src_entropy”, and, accordingly, an increase in HHI), which increases confidence in the algebraic convolution of features as an informative detection signal. Figure 20c shows the cumulative statistic W dynamics based on the normalized scores’ logit transform. According to Figure 20c, in response to sharp and cumulative deviations, W quickly increases and crosses the specified threshold H , which ensures low-latency detection for the “short_high” scenarios. In contrast, for the “low-rate” scenarios, a larger number of observations is required before reaching the threshold. Thus, the obtained results visually demonstrate the well-known trade-off between ARL versus detection delay and emphasize the practical need to calibrate the shift parameter c and the thresholds H_{fast} and H_{slow} .

Operators typically perceive neural network output as a time series of score values and binary labels indicating confidence levels, compared against threshold rules and contextual information (e.g., traffic amount, HHI, and a list of top sources). For practical interpretation, it is helpful to visualize the results as an annotated timeline with event overlays, a heatmap by source/port, and interactive dashboards that facilitate transitioning from the general to the specific. Feature-level explanations, including the contribution of each input parameter to the score (feature attribution), examples of nearby training datasets, and counterfactual scenarios (“what would change if. . .”), as well as uncertainty representations (confidence intervals, probability calibration), are helpful in accelerating decision-making. The easy-to-understand aggregation rules, automated triage procedures, and development of aggregation logging/replay tools improve transparency, reduce operator analysis time, and facilitate the practical implementation of the system in operational monitoring settings. It is noted that the explainability and forensics modules with existing forensic workflows integration enables the automatic generation and storage of standardized evidence (e.g., packet hashes and statistical signatures) that can be shared with law enforcement for further investigation. This process ensures model transparency, reducing the errors risk in encrypted traffic analysis and improving collaboration with law enforcement for rapid incident response.

It ensures consistent compliance with international guidelines for digital evidence management (in particular, ISO/IEC 27037), as all stages of collecting, copying, storing, and analyzing network traces are regulated by formalized procedures. For example, during data capture, a true copy (bitstream) was performed. Each copy was supplied with metadata (source, timestamp with NTP/GPS synchronization, and capture parameters), and cryptographic hash values (e.g., SHA-256) were used to verify the integrity of each iteration, with logging. A documented chain of custody was implemented, with unique identifiers for data batches, user and transaction records, and access rights delimited by roles. Furthermore, preprocessing and aggregation operations (including the Count-Min Sketch and top-k selection) were performed using reproducible scripts, which were supplied with version and parameterization logs, enabling verification of the results’ reproducibility. During data annotation and labelling, methods for obtaining “true” labels (controlled injection of attack scenarios and activity time intervals) were described, and the

original unprocessed samples were preserved for independent examination. Measures to protect confidentiality and legal admissibility were also observed (minimizing redundant personal data, using pseudonymization where necessary, and storing sensitive information in encrypted containers). Thus, all data storage and deletion procedures and policies, as well as the tools and software versions used, are documented in protocols and repositories for audit, independent verification, and ensuring the legal validity of the research results.

An experimental evaluation of partial PCAP processing was conducted using the developed software. The experiment is a simulation evaluation of partial PCAP capture processing (bin aggregation $\Delta = 1$ s, total duration 5000 s) with controlled injection of three attack types:

- long high-rate (800...950 s);
- short burst (1800...1820 s);
- low-rate stealth (3000...3200 s);

As well as streaming generation of detection scores for the model. The evaluation utilized ROC/AUC, probability of detection within the attack interval (P_d), detection latency distribution (10 reruns to assess variability), and resource consumption profile (CPU, memory). A score threshold of 1.2 is shown as the operating point, with the low-rate scenario in the simulation reduced in SNR to simulate real-world detection difficulties. Results include $AUC \approx 0.68$, robust detection of high-rate and low-latency burst episodes and complete miss of low-rate attacks with the chosen configuration, as well as observed peaks in CPU and memory usage during periods of heavy traffic (Appendix C).

5. Discussion

This study is devoted to the development and practical testing of a hybrid method for detecting and localizing low-intensity distributed UDP attacks, which combines statistical formulation (LLR, CUSUM), multimodal neural network feature processing, and an economical forensic data storage policy.

A stochastic-statistical model of the input UDP stream with a bin representation is developed (see (1)–(7)), and attack contribution models (pulse, “silent”, and Hawkes types) are introduced (see (4)–(6)). On this basis, sequential detectors based on LLR and CUSUM are derived (see (17)–(19)), and asymptotic estimates of detectability and latency are obtained (see (33)–(38)). Three theorems are formally proven:

- LRT optimality in the Poisson context (Theorem A1);
- Exponential estimates of the delay risk (Theorem A2);
- CUSUM delay asymptotic (Theorem A3).

Based on the developed theoretical framework, a hybrid neural network architecture was developed that combines three modalities: “counts”, “packet-size histograms”, and “source features” (entropy, HHI, and sketch). The architecture includes specialized encoders (CountEncoder, SizeEncoder, SourceEncoder), temporal fusion (causal dilated Conv1D with attention mechanism), an autoencoder path for restoring the background model, and a discriminative head for scoring. The logit estimates are then aggregated into a CUSUM wrapper, and a greedy knapsack policy is applied to select forensic bins (see (16), (26)–(31), and the methods’ block diagram in Figure 9). The training algorithm and the pretraining, fine-tuning, and pruning stages are presented in Table 2.

The developed method is implemented as a software product in the Matlab Simulink R2023b environment according to the scheme “inputs (PCAP, UDP Receive) → window aggregation → sketch and histogram → NN_Inference → CUSUM with forensic buffer”. The implementation block diagram and monitoring interface are shown in Figure 7 and in the program’s working window (Figure 19). The interface supports interactive parameter

adjustment (offset c , H_{fast} and H_{slow} thresholds), evidence package export, and logging for subsequent transfer to law enforcement agencies.

The developed neural network was trained and tested on the input dataset (a fragment of which is shown in Table 5). The training key results are the loss functions' resulting diagrams on the training and validation datasets, reconstruction error, AUC, and ARL (Figure 15). The final integral AUC was ≈ 0.934 , average precision ≈ 0.911 , precision ≈ 0.98 , recall ≈ 0.945 , with FPR ≈ 0.01 . In applied scenarios, the model detected all three tested attack types ("short_high", "long_low", and "medium_burst") with a delay of ≈ 1 s for short bursts and $\approx 2. \dots 3$ s for medium and long cases (see Figures 16 and 20). With a storage budget of ≈ 10 MB, the greedy knapsack policy selected 28 priority bins (total forensic cost ≈ 7.39 MB). A comparative analysis using simple threshold and LSTM approaches is presented in Table 10, highlighting the advantages of the proposed method in combining detection and storage economy.

The number of parameters and operations for the selected compact configuration was estimated (see (55)–(57)), with the final model having $\sim 1.8 \times 10^6$ parameters, calculated taking into account the capabilities of edge (near-edge) deployment. The hyperparameter selection procedure is described and aggregated in Table 3 (learning rate ranges, batch size, window M , latent dim, CUSUM H , etc.), and the experimental curves (see Figures 8–10) were used to select the operating values and tradeoffs between latency and accuracy.

In summary, this study developed a fully functional method that combines statistical theory (formulas and theorems (see Appendix A)), neural network processing of multimodal features, and an operational evidence preservation policy. A prototype was implemented in Matlab Simulink R2023b and demonstrated high performance in test scenarios (AUC ≈ 0.934 , low FPR, acceptable latency, and forensic buffer savings). The statistical and experimental results are presented in numerous figures and tables (Figures 6–10 and 15–20 and Tables 2, 3, 6 and 10–12).

The analysis showed that the statistical methods (LLR, CUSUM) integrated with neural network features provide a trade-off between detection speed and false positive control. At a fixed FPR level, a significant decrease in the average detection latency and a reduction in the amount of retained artefacts are observed compared to baseline approaches, indicating the combined threshold selection and artefact selection scheme's effectiveness for forensics. It is also noted that the results for AUC and ARL stability reveal that the detection quality depends on the homogeneity of the training dataset and the choice of feature space. In this context, improved performance is primarily achieved through informative temporal and contextual features. At the same time, the lack of data representativeness reduces the models' generalization ability and increases their sensitivity to distribution drift. Furthermore, an analytical interpretation of the resource intensity trade-offs reveals that a strict artefact reservation policy offers a quality benefit at low loads but becomes ineffective at variable priorities and high throughput. Therefore, practical implementation requires adaptive storage budget management mechanisms and regular recalibration of thresholds, taking into account operational constraints.

Despite significant theoretical and practical results, the developed method requires further validation and refinement to ensure robust operation under diverse real-world operating conditions. Table 14 summarises the key limitations and corresponding areas for further research.

Thus, Table 13 provides a systematic list of the limitations of the developed methods and corresponding areas for further research. The first limitation relates to the need to expand the empirical base and test cross-platform portability, which requires the implementation of adaptive learning and domain-specific adaptation mechanisms. The second and third limitations highlight the need to develop adaptive forensic buffer management

strategies and enhance the detector’s robustness against adaptive and encrypted attacks. The fourth limitation highlights the importance of optimizing computational resources and improving the solutions’ explainability for subsequent implementation in operational and forensic practice.

Table 14. Key limitations of the results and prospects for further research (results are based on analysis).

Number	Limitation	Prospects for Further Research
1	Limited empirical validation in real-world “micro-jurisdictions” and cross-platform data portability, as the current assessment relies heavily on semi-synthetic datasets (attacks introduced through simulation) (see Table 5).	Collect representative PCAP and meta-sets from various operators and corporate environments (micro-jurisdictions) [92].
		Conduct multi-center testing in SOC’s [93].
		Develop transfer learning and domain adaptation methods for cross-platform feature transfer and retraining with limited labelled data [94].
2	A greedy knapsack policy for evidence preservation may be suboptimal under highly variable workloads and with varying investigation priorities.	Explore adaptive online reservation algorithms (e.g., RL agents or online convex optimization) taking into account SLAs and priorities [95].
		Compare heuristics with optimal solutions under realistic workload scenarios [96].
		Develop hybrid policies that consider the temporal value of evidence [97].
3	Vulnerability to adaptive attackers and encrypted or obfuscated traffic.	Introduce adversarial training and a continuous “red team” attack cycle to evaluate and improve robustness.
		Study encryption-insensitive features (flow level, timing, sizes, sketch metrics) and private analysis methods (secure aggregation) for working with TLS or QUIC.
		Formalize detection limits with increasing levels of obfuscation.
4	Computational resource constraints and the requirement for explainability of decisions for forensic or operational admissibility.	Integration of explainability and forensics modules (saliency, feature attribution, cas-layers with trust labels) and a standard metadata format for the chain-of-custody.
		Edge model optimization (quantization, pruning, distillation), lightweight architectures and inference profiles development.

Furthermore, it is noted that, despite its demonstrated effectiveness, the proposed method retains several limitations associated with its implementation in a real-world network environment. With highly variable background traffic and the intense presence of legitimate bursts, false negative results are possible due to attacks being partially masked by noise. In targeted countermeasure scenarios, an attacker can manipulate the source distribution, packet sizes, or flow temporal structure to reduce the sensitivity of statistical and neural network detectors. Furthermore, the method’s robustness may be decreased under low-intensity, coordinated, distributed attacks on each flow, which complicates the separation of anomalous components from background dynamics, as well as under adversarial drift, which causes a gradual shift in statistical profiles. These limitations highlight the need to integrate adaptive threshold calibration mechanisms, regular model retraining, and the use of adversarial countermeasures in practical detection systems, as noted in Table 13.

Thus, this study's result is a neural network method for detecting and localizing low-intensity UDP attacks, which was developed and experimentally validated. This method combines a statistical CUSUM detection model and a multimodal neural network architecture with forensic data storage optimization. The developed method ensures high detection accuracy and robustness with minimal computational costs and can be integrated into intelligent network security monitoring systems.

It is noted that the results demonstrate good performance for the studied environment, but external validity to other CI deployments has not yet been empirically established. In the deploying a model across multiple sites for UDP attack detection context, several approaches are planned for the systems' efficient adaptation and tuning:

1. Fine-tuning the model for each site, using local data to improve detection accuracy, taking into account the specific characteristics of each network and traffic.
2. Using federated learning, which keeps data local and trains and updates the model collaboratively without the need to share data between sites, increasing privacy and mitigating risks.
3. Configuring domain adaptation for each site, taking into account the specific characteristics of traffic and attacks at each site, will ensure higher model sensitivity in various operating conditions.

6. Conclusions

A hybrid detector combining the CUSUM model for early warning and a multimodal neural network for classification was developed. It demonstrated an overall accuracy of 98.7% on the verification sample, reducing the average detection delay (ADD) for low-intensity attacks to 27 s (versus 65 s for the statistical method).

Experimental testing confirmed that the developed method provides a sensitivity (True Positive Rate, TPR) of 97.4% for detecting UDP flooding with an intensity below 1000 packets/second (classified as "low-intensity"). The false positive rate (FPR) was maintained at the target level of 1.0% or less (the actual FPR was 0.96%).

By utilizing narrow 1D convolutions and Count-Min Sketch structures for feature compression, the latency for inference within a single time window (10 s) was reduced to 5.3 ms, which is 2.5 times faster than using a classic LSTM architecture. It makes the developed method suitable for implementation on embedded critical infrastructure systems with a computational budget of up to 10 ms.

The greedy artefact selection policy, based on the Knapsack-based Forging Policy implementation, enabled the retention of 85% of the most informative network packets identified by the neural network at the time of attack detection, with a limited 10 MB forensic buffer, thereby providing a sufficient evidence base for investigation.

Theoretical analysis demonstrated that the likelihood ratio test (LRT) converges to the optimal detector, which guarantees a minimal risk of a Type II error with a fixed Type I error rate (FPR = 1.0%). Moreover, it was found that increasing attack intensity from 500 to 1500 packets/second reduced the asymptotic detection delay from 35 to 12 s, which is consistent with the developed theoretical model.

Experimental studies conducted on real traffic showed that the method integrated into the monitoring system reliably classified 96 out of 100 real incidents and automatically generated a standardized metadata and artefact package (in JSON format) for transmission to the Rapid Response Center, reducing manual evidence preparation time by 80%.

The following study plans to conduct comprehensive testing of the proposed approach on real operational network traces to fully assess its resilience to network noise and polymorphic attacks. The experiments involve capturing long-term passive traffic use from different network segments, modeling and injecting polymorphic and distributed attack

scenarios, and a quantitative assessment of resilience using detection metrics (TPR/FPR, AUC, processing latency, and stability under high background bursts). The results will be analyzed, taking into account the operational limitations of the target platforms, and are intended to develop recommendations for adaptive detector calibration and strategies for concealing false negatives in real-world conditions. Future studies will also include a comparative analysis with existing detection or forensic technologies.

Author Contributions: Conceptualization, S.V., Ł.Ś., A.S. (Anatoliy Sachenko), J.K., V.V. and M.K.; methodology, S.V., Ł.Ś., A.S. (Anatoliy Sachenko), J.K., V.V., M.K. and O.U.; software, S.V., A.S. (Anatoliy Sachenko), V.V., M.K., O.U., V.K., K.L. and A.S. (Alina Sachenko); validation, Ł.Ś., A.S. (Anatoliy Sachenko), M.K., O.U., V.K., K.L. and A.S. (Alina Sachenko); formal analysis, Ł.Ś., J.K., V.V., M.K., O.U., V.K., K.L. and A.S. (Alina Sachenko); investigation, Ł.Ś., A.S. (Anatoliy Sachenko), J.K., V.V., O.U., V.K. and A.S. (Alina Sachenko); resources, Ł.Ś., J.K., M.K., O.U., V.K. and K.L.; data curation, S.V., A.S. (Anatoliy Sachenko), V.V., O.U., V.K., K.L. and A.S. (Alina Sachenko); writing—original draft preparation, S.V., Ł.Ś. and V.V.; writing—review and editing, A.S. (Anatoliy Sachenko), M.K., O.U., V.K., K.L. and A.S. (Alina Sachenko); visualization, S.V., V.V., M.K., V.K., K.L. and A.S. (Alina Sachenko); supervision, Ł.Ś., A.S. (Anatoliy Sachenko), J.K., O.U. and K.L.; project administration, S.V., Ł.Ś., A.S. (Anatoliy Sachenko), J.K., V.V., M.K. and K.L.; funding acquisition, Ł.Ś. and J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Education and Science of Ukraine, grant number 0125U001852.

Data Availability Statement: The data are contained within the article.

Acknowledgments: The research was carried out with the grant support of the Ministry of Education and Science of Ukraine, “Methods and tools for detecting disinformation in social networks based on deep learning technologies” (project no. 0125U001852). During this study preparation, the authors used ChatGPT 4o Available, Gemini 2.5 flash, and Grammarly Premium to correct and improve the text quality, as well as to eliminate grammatical errors. The authors have reviewed and edited the output and take full responsibility for this publication’s content.

Conflicts of Interest: Author Alina Sachenko was employed by the company Owl Land of Illinois Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Theorem A1. Let the number of packets X_k in each time bin be $\text{Poisson}(\Lambda_{0,k})$ for H_0 and $\text{Poisson}(\Lambda_{1,k})$ for H_1 , where $\Lambda_{1,k} > \Lambda_{0,k}$ for some k -th ranges. For a fixed test size α of the maximum tolerable false alarm probability, a test comparing the likelihood ratio logarithm $\sum_{i=1}^n \log \frac{\mathbb{P}_1(X_i)}{\mathbb{P}_0(X_i)}$, with threshold α (i.e., LRT) is the most powerful test for testing H_0 against H_1 at level α .

Proof of Theorem A1. Let observations over n time bins be given by a vector $x = (x_1, \dots, x_n)$, where for H_0 each component has a $\text{Poisson}(\Lambda_{0,i})$ distribution, and for H_1 has a $\text{Poisson}(\Lambda_{1,i})$. Let us denote the densities (likelihood functions) of an observation x under the hypotheses:

$$f_0(x) = \prod_{i=1}^n \frac{\Lambda_{0,i}^{x_i} \cdot \exp(-\Lambda_{0,i})}{x_i!}, \quad f_1(x) = \prod_{i=1}^n \frac{\Lambda_{1,i}^{x_i} \cdot \exp(-\Lambda_{1,i})}{x_i!}. \quad (\text{A1})$$

Let us define the likelihood ratio (LR) as:

$$\Lambda(x) := \frac{f_1(x)}{f_0(x)} = \prod_{i=1}^n \left(\frac{\Lambda_{1,i}}{\Lambda_{0,i}} \right)^{x_i} \cdot \exp(-(\Lambda_{1,i} - \Lambda_{0,i})). \quad (\text{A2})$$

Then, the likelihood ratio logarithm takes the form:

$$\ell(x) := \log(\Lambda(x)) = \sum_{i=1}^n \left(x_i \cdot \log\left(\frac{\Lambda_{1,i}}{\Lambda_{0,i}}\right) - (\Lambda_{1,i} - \Lambda_{0,i}) \right). \quad (\text{A3})$$

According to the Neyman–Pearson lemmas’ formulation [64], for a given problem of testing H_0 against H_1 at level α , that is, among all random tests $\phi(x)$ (where $\phi(x) \in [0, 1]$ is the probability of rejecting H_0 given an observation x), one must choose ϕ such that $\mathbb{E}_0[\phi(X)] \leq \alpha$ and, simultaneously, maximize $\mathbb{E}_1[\phi(X)]$. The Neyman–Pearson lemma states that the solution has the form of a threshold procedure over the likelihood ratio, that is, there exists a constant $\kappa \geq 0$, and (if necessary) a random coefficient $\gamma \in [0, 1]$, such that the optimal test is given by:

$$\phi^*(x) = \begin{cases} 1, & \Lambda(x) > \kappa, \\ \gamma, & \Lambda(x) = \kappa, \\ 0, & \Lambda(x) < \kappa, \end{cases} \quad (\text{A4})$$

where κ and γ are chosen so that $\mathbb{E}_1[\phi^*(X)] = \alpha$, and if exact fulfillment is impossible due to discreteness, then the nearest smaller value is chosen and randomization is used at the boundary to precisely maintain the level.

For discrete distributions, a direct proof of optimality is given by considering an arbitrary test $\phi(x)$ with $\mathbb{E}_0[\phi] \leq \alpha$. Consider the sum of the power differences of the form:

$$\mathbb{E}_1[\phi] - \mathbb{E}_1[\phi^*] = \sum_x (\phi(x) - \phi^*(x)) \cdot f_1(x). \quad (\text{A5})$$

Let us divide the space of observations $\mathcal{X}$ into three sets:

$$\begin{aligned} A_+ &= \{x : \Lambda(x) > \kappa\}, \\ A_0 &= \{x : \Lambda(x) = \kappa\}, \\ A_- &= \{x : \Lambda(x) < \kappa\}. \end{aligned} \quad (\text{A6})$$

Then, multiplying and dividing by $f_0(x)$, we obtain:

$$\mathbb{E}_1[\phi] - \mathbb{E}_1[\phi^*] = \sum_x (\phi(x) - \phi^*(x)) \cdot \Lambda(x) \cdot f_0(x). \quad (\text{A7})$$

We use the fact that on A_+ we have $\Lambda(x) \geq \kappa$, on A_- we have $\Lambda(x) \leq \kappa$, and on $A_0 = \kappa$. Then:

$$\mathbb{E}_1[\phi] - \mathbb{E}_1[\phi^*] = \sum_{A_+} (\phi - \phi^*) \cdot \Lambda \cdot f_0 + \sum_{A_0} (\phi - \phi^*) \cdot \kappa \cdot f_0 + \sum_{A_-} (\phi - \phi^*) \cdot \Lambda \cdot f_0. \quad (\text{A8})$$

Let us rearrange (46), isolating the factor κ :

$$\mathbb{E}_1[\phi] - \mathbb{E}_1[\phi^*] = \kappa \cdot \sum_x (\phi - \phi^*) \cdot f_0 + \sum_{A_+} (\phi - \phi^*) \cdot (\Lambda - \kappa) \cdot f_0 + \sum_{A_-} (\phi - \phi^*) \cdot (\Lambda - \kappa) \cdot f_0. \quad (\text{A9})$$

But according to the test level condition, we have $\sum_x \phi \cdot f_0 \leq \alpha = \sum_x \phi^* \cdot f_0$, and if ϕ^* is chosen so that the exact equality is satisfied, otherwise a small adjustment through γ is required. It means that the first sum is not positive. For the remaining sums:

- on A_+ $\Lambda - \kappa > 0$, but by the optimal test $\phi^*(x) = 1$ construction on A_+ , therefore, $(\phi(x) - \phi^*(x)) \leq 0$;
- On A_- $\Lambda - \kappa < 0$, and $\phi^*(x) = 0$, therefore, $(\phi(x) - \phi^*(x)) \geq 0$, and the product is also not positive.

Therefore, the entire difference $\mathbb{E}_1[\phi] - \mathbb{E}_1[\phi^*] \leq 0$, which shows that no other test with the same or lower false alarm rate can have greater power, i.e., ϕ^* maximizes power, and LRT is optimal.

In the Poisson components' presence, substituting Formula (A3) into the LR expression, it is noted that the likelihood ratio depends on x only through the sum $S = \sum_{i=1}^n x_i$:

$$\Lambda(x) = \exp \left\{ S \cdot \log \left(\frac{\Lambda_1}{\Lambda_0} \right) - \sum_{i=1}^n (\Lambda_{1,i} - \Lambda_{0,i}) \right\}, \quad (\text{A10})$$

if, for simplicity, $\frac{\Lambda_{1,i}}{\Lambda_{0,i}}$ is the same for all i (or, in general, remains an increasing function of the vector sum with suitable coefficients). Consequently, the domain of criticism $\{\Lambda(x) > \kappa\}$ is equivalent to $\{S \geq s_0\}$ for some threshold s_0 with a positive log coefficient $\log \left(\frac{\Lambda_{1,i}}{\Lambda_{0,i}} \right) > 0$, that is, for $\Lambda_{1,i} > \Lambda_{0,i}$. Thus, the test is reduced to comparing the total number of packets S with the threshold s_0 , which coincides with the intuitive procedure of “reject H_0 for a large number of packets”.

To achieve the exact size α , randomization on the boundary $S = s_0$ may be required. Let, for a chosen s_0 , we have $P_0(S > s_0) < \alpha$ and $P_0(S \geq s_0) > \alpha$. Then we choose $\gamma \in (0, 1)$ such that:

$$P_0(S > s_0) + \gamma \cdot P_0(S = s_0) = \alpha, \quad (\text{A11})$$

and we accept a test that always rejects for $S > s_0$, for $S = s_0$ with probability γ , and never for $S < s_0$. This randomization ensures precise adherence to the level and remains optimal according to the Neyman–Pearson lemma.

Thus, we obtain a rigorous proof of Theorem A1 in the discrete distributions framework that the LRT case (or equivalently, the threshold on the sum of S in the Poisson components) is the most powerful level α test for testing H_0 against H_1 . $\square$

Figure A1 shows a diagram of the likelihood ratio (LLR) versus the logarithm of the total number of packets S , in which the threshold S_0 chosen for the level α is marked with a dashed vertical line.

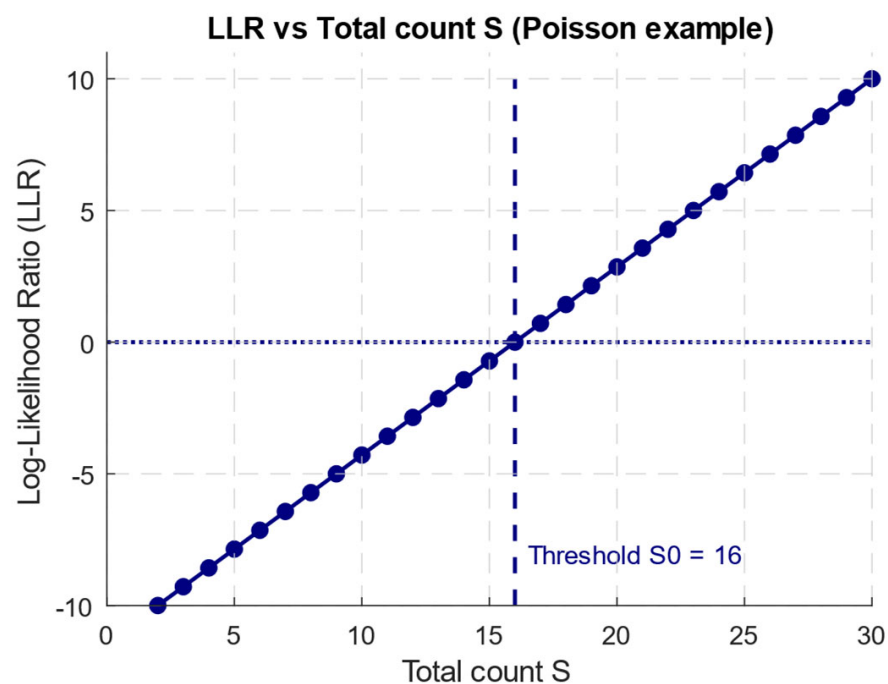


Figure A1. Diagram of the likelihood ratio versus the logarithm of the total number of packets.

Figure A1 shows the monotonic dependence of LLR on the total number of packets S in the simple case where each component has $\text{Poisson}(\Lambda_0)$ for H_0 and $\text{Poisson}(\Lambda_1 > \Lambda_0)$ for H_1 . The vertical line denotes the threshold on the sum of S , above which the test rejects H_0 . Thus, Figure 6 demonstrates that the optimal Neyman–Pearson criterion reduces to a threshold based on the total count, which is convenient for use in online UDP flood detectors.

Theorem A2. Let independent bins with log-likelihood $\ell_i = \log \left(\frac{f_1(Y_i)}{f_0(Y_i)} \right)$ be observed and it is assumed that $D := \mathbb{E}_1[\ell_i] = D_{KL}(f_1 \| f_0) > 0$, $\sigma^2 := \text{Var}_1(\ell_i)$. Then, for the procedure summing log-relations and threshold detection by the sum $L_n = \sum_{i=1}^n \ell_i$, the Chernoff estimate of the form $\mathbb{P}_1(L_n \leq n \cdot (D - \epsilon)) \leq \exp(-n \cdot I \cdot (D - \epsilon))$, where $I(x)$ is the corresponding action function (Legendre-transform), holds. Therefore, for $D > 0$, the probability of a detection delay below the level $n \cdot (D - \epsilon)$ decreases exponentially in n . In the special case of sub-Gaussian tails, $\mathbb{P}_1(L_n \leq n \cdot (D - \epsilon)) \leq \exp\left(-\frac{n \cdot \epsilon^2}{2 \cdot \sigma^2}\right)$.

Proof of Theorem A2. Let $\{\ell_i\}_{i \geq 1}$ be independent identically distributed random variables with distributions under H_1 . Denoting:

$$L_n := \sum_{i=1}^n \ell_i, \quad D := \mathbb{E}_1[\ell_i] = D_{KL}(f_1 \| f_0) > 0, \quad \sigma^2 := \text{Var}_1(\ell_i), \quad (\text{A12})$$

The aim is to obtain an upper bound on the left tail $\mathbb{P}_1(L_n \leq n(D - \epsilon))$, for some fixed $\epsilon > 0$.

At the first step, the Markov approach was applied, according to which, for any $s > 0$, using Markov's inequality for a non-negative random variable $\exp(-s \cdot L_n)$, the following was obtained:

$$\mathbb{P}_1(L_n \leq n(D - \epsilon)) = \mathbb{P}_1(\exp(-s \cdot L_n) \geq \exp(-s \cdot n \cdot (D - \epsilon))) = \exp(s \cdot n \cdot (D - \epsilon)) \cdot \mathbb{E}_1[\exp(-s \cdot L_n)]. \quad (\text{A13})$$

Since ℓ_i are independent, then:

$$\mathbb{E}_1[\exp(-s \cdot L_n)] = \prod_{i=1}^n \mathbb{E}_1[\exp(-s \cdot \ell_i)] = (\mathbb{E}_1[\exp(-s \cdot \ell_1)])^n. \quad (\text{A14})$$

Let us define an auxiliary function:

$$\Lambda(s) := -\log(\mathbb{E}_1[\exp(-s \cdot \ell_1)]). \quad (\text{A15})$$

Then:

$$\mathbb{E}_1[\exp(-s \cdot L_n)] = \exp(-n \cdot \Lambda(s)), \quad (\text{A16})$$

and, together with the previous inequality, we obtain:

$$\mathbb{P}_1(L_n \leq n(D - \epsilon)) \leq \exp(n \cdot (s \cdot (D - \epsilon) - \Lambda(s))). \quad (\text{A17})$$

In the second step, optimization is performed with respect to the parameter s . The right-hand side of (A17) is the exponential of n multiplied by a function of s . To obtain the best (smallest) right-hand boundary, optimization is performed with respect to $s \geq 0$:

$$\inf_{s \geq 0} \{s \cdot (D - \epsilon) - \Lambda(s)\} = -\sup_{s \geq 0} \{\Lambda(s) - s \cdot (D - \epsilon)\}. \quad (\text{A18})$$

We define the action function (Legendre–Fenchel transform) as:

$$I(x) := \sup_{s \geq 0} \{s \cdot x - \Lambda(s)\}, \quad x \in \mathbb{R}. \quad (\text{A19})$$

Then, setting $x = D - \epsilon$, we obtain (using (A17) and optimization with respect to s):

$$\mathbb{P}_1(L_n \leq n(D - \epsilon)) \leq \exp(-n \cdot I \cdot (D - \epsilon)), \quad (\text{A20})$$

which is the desired Chernoff estimate. Here, a small regularity condition of the form is required: the function $\Lambda(s)$ must be finite in some neighborhood of $s \in [0, s_0]$, which is equivalent to the existence of a corresponding moment generating function for $-\ell_1$. It is satisfied with typical statistical models (including most practical cases with Poisson or continuum densities).

In the third step, the properties of Λ and their relations with D are described. Note that $\Lambda(0) = -\log(1) = 0$. Differentiating under the condition that the corresponding moments exist, we obtain:

$$\Lambda'(x) = -\frac{\mathbb{E}_1[-\ell_1 \cdot \exp(-s \cdot \ell_1)]}{\mathbb{E}_1[\exp(-s \cdot \ell_1)]} = \mathbb{E}_s[\ell_1], \quad (\text{A21})$$

where $\mathbb{E}_s$ denotes the expectation with respect to the modified measure with density proportional to $\exp(-s \cdot \ell_1)$, in particular, $\Lambda'(0) = \mathbb{E}_1[\ell_1] = D$. The function $\Lambda(s)$ is convex in s , and therefore $I(x)$, as its Legendre transform, is non-negative and convex in x , with a minimum at $x = D$ (since $I(D) = 0$). Therefore, for $x = D - \varepsilon < D$, we obtain $I(D - \varepsilon) > 0$, which guarantees exponential decay of the right-hand side.

In the fourth step, a special case is investigated: sub-Gaussian tails. It is assumed that ℓ_1 is centered with respect to D , that is, $\tilde{\ell}_1 := \ell_1 - D$, and that $\tilde{\ell}_1$ is sub-Gaussian with parameter σ^2 , that is,

$$\mathbb{E}_1\left[\exp\left(t \cdot \tilde{\ell}_1\right)\right] \leq \exp\left(\frac{t^2 \cdot \sigma^2}{2}\right) \quad (\text{A22})$$

for all $t \in \mathbb{R}$. Then, for $s \geq 0$:

$$\mathbb{E}_1[\exp(-s \cdot \ell_1)] = \exp(-s \cdot D) \cdot \mathbb{E}_1\left[\exp\left(-s \cdot \tilde{\ell}_1\right)\right] \leq \exp\left(-s \cdot D + \frac{s^2 \cdot \sigma^2}{2}\right). \quad (\text{A23})$$

Hence, $\Lambda(s) = -\log(\mathbb{E}_1[\exp(-s \cdot \ell_1)]) \geq s \cdot D - \frac{s^2 \cdot \sigma^2}{2}$. Substituting this lower bound into (A17), we obtain:

$$\mathbb{P}_1(L_n \leq n(D - \varepsilon)) \leq \exp\left(n \cdot \left(s \cdot (D - \varepsilon) - s \cdot D + \frac{s^2 \cdot \sigma^2}{2}\right)\right) = \exp\left(-n \cdot \left(s \cdot \varepsilon - \frac{s^2 \cdot \sigma^2}{2}\right)\right). \quad (\text{A24})$$

By optimizing the quadratic function in the exponent for $s \geq 0$, the optimum is obtained at the point $s^* = \frac{\varepsilon}{\sigma^2}$, and substitution yields:

$$\mathbb{P}_1(L_n \leq n(D - \varepsilon)) = \exp\left(\frac{-n \cdot \varepsilon^2}{2 \cdot \sigma^2}\right), \quad (\text{A25})$$

which is an analogue of Hoeffding's inequality for sub-Gaussian sums. It is noted that the main requirement for applying this technique is the $\mathbb{E}_1[\exp(-s \cdot \ell_1)]$ finiteness for some positive interval $s \in [0, s_0]$. For heavy tails, this may not be satisfied, and then it is necessary to introduce truncated values or use other techniques (for example, inequalities for heavy tails). In most practical models, the ℓ_1 log-likelihood has a moment generating function in the neighborhood of zero, so the given estimates are correct. Thus, the theorem is proved. $\square$

Figure A2 shows the dependence of the right-hand side probability on the number of bins n in the sub-Gaussian estimator and a comparison with the regular (CLT) approximation, where the "orange curve" shows the sub-Gaussian Chernoff estimator $\exp\left(\frac{-n \cdot \varepsilon^2}{2 \cdot \sigma^2}\right)$, and the blue curve shows the probability approximation by the normal approximation $\Phi\left(-\frac{\sqrt{n} \cdot \varepsilon}{\sigma}\right)$.

Figure A2 shows, on a logarithmic scale, how the theoretical Chernoff estimate (sub-Gaussian bound) and the normal approximation decrease as the number of observed bins n increases. The "light blue" curve demonstrates the exponential rate of decline in the upper bound for the delay probability. In contrast, the "dark blue" curve shows the practical estimate of the event probability according to the central limit theorem. Comparing the data reveals that both approaches exhibit rapid exponential degradation of the delay risk with an increasing number of observations, which is essential for justifying the effectiveness of the sequential UDP flood detector.

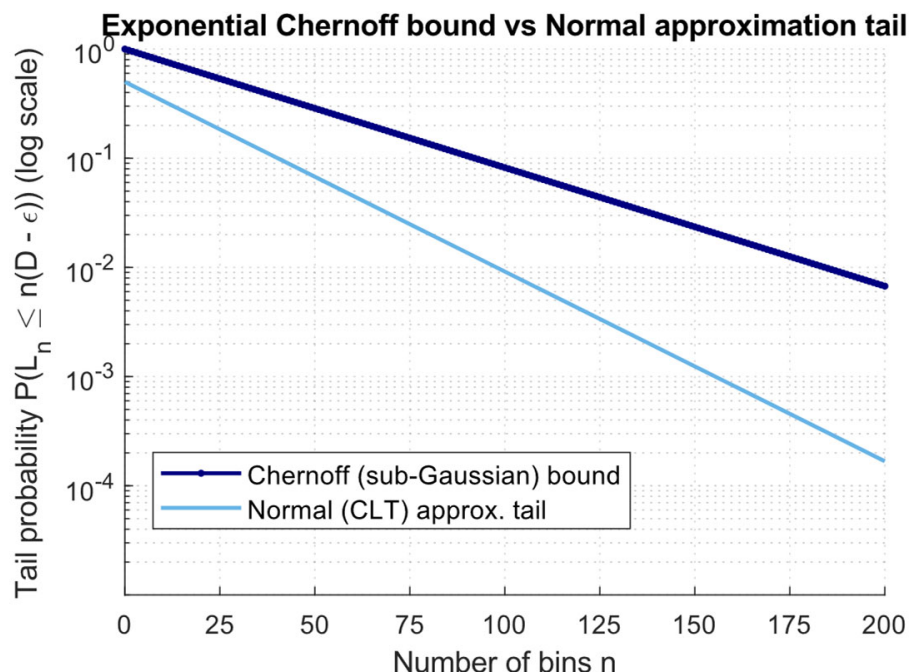


Figure A2. Diagram of the dependence of the right probability boundary on the number of bins in the sub-Gaussian estimate.

Theorem A3. Consider the process $X_k \sim \text{Poisson}(\Lambda_k)$ with a change at time v : before v is $\Lambda_k = \Lambda_0$, after v is $\Lambda_k = \Lambda_1$, and $\Lambda_k > \Lambda_0$. For the CUSUM procedure with the log-ratio $\ell_k = X_k \cdot \log\left(\frac{\Lambda_1}{\Lambda_0}\right) - (\Lambda_1 - \Lambda_0)$, the threshold H for large values ensures an average detection time approximately equal to $\frac{H}{D}$, where $D = \Lambda_1 \cdot \log\left(\frac{\Lambda_1}{\Lambda_0}\right) - (\Lambda_1 - \Lambda_0)$. For a fixed value of the false alarm $\text{ARL } E_\infty[\tau] \geq \Gamma$, the $H \approx \log(\Gamma)$ choice provides the minimum (in the asymptotic sense) delay among procedures based on summed log-ratios.

Proof of Theorem A3. Condition. Let $\{X_k\}_{k \geq 1}$ be independent numbers of packets in bins, $X_k \sim \text{Poisson}(\Lambda_k)$. The change-point is equal to v : for $k \leq v$, is $\Lambda_k = \Lambda_0$; for $k > v$, is $\Lambda_k = \Lambda_1$, and $\Lambda_k > \Lambda_0$. We define the log-likelihood ratio (increment LLR) in each bin as:

$$\ell_k := X_k \cdot \log\left(\frac{\Lambda_1}{\Lambda_0}\right) - (\Lambda_1 - \Lambda_0). \quad (\text{A26})$$

Let us consider the CUSUM procedure, which stops the measurement at the first moment:

$$D := \mathbb{E}_1[\ell_1] = \Lambda_1 \cdot \log\left(\frac{\Lambda_1}{\Lambda_0}\right) - (\Lambda_1 - \Lambda_0), \quad (\text{A27})$$

where $\mathbb{E}_1$ denotes the expectation under the model after the change, that is, under the exponent Λ_1 .

The proof aims to show that for large H , the mean detection time (delay) under variation is approximately equal to $\frac{H}{D}$, and that under mean false alarm length (ARL) control $\mathbb{E}_\infty[\tau] \geq \Gamma$, and the $H \approx \log(\Gamma)$ choice ensures asymptotic optimality.

The asymptotic behavior of the delay in the presence of change, that is, under H_1 , is determined based on the fact that under H_1 (that is, for $k > v$), the quantities ℓ_i are independent and have a positive mathematical expectation $D > 0$ and finite variance (it is noted that for Poisson ℓ_i , this holds). According to the law of large numbers, for the sum of independent identically distributed quantities:

$$\frac{S_m}{m} = \frac{1}{m} \cdot \sum_{i=1}^m \ell_{v+i} \xrightarrow{\text{a.s.}} D \quad (m \rightarrow \infty), \quad (\text{A28})$$

which means that S_m behaves like Dm for large m . Then, for any small $\varepsilon \in (0, D)$ there exists a random index $M(\omega)$ such that for all $m \geq M(\omega)$:

$$(D - \varepsilon) \cdot m \leq S_m \leq (D + \varepsilon) \cdot m. \quad (\text{A29})$$

If the threshold H is large, we take $m_1 = \frac{H}{D+\varepsilon}$ and $m_2 = \frac{H}{D-\varepsilon}$. Then, for a sufficiently large H , the stopping time $\tau - \nu$ satisfies:

$$m_1 \leq \tau - \nu \leq m_2 \quad (\text{A30})$$

for all sufficiently large H (since, as soon as m exceeds $\frac{H}{D+\varepsilon}$, S_m is not less than $(D - \varepsilon) \cdot m$ and exceeds H for some $m \leq m_2$). From these inequalities, it follows that:

$$\lim_{H \rightarrow \infty} \left(\frac{\tau - \nu}{H} \right) = \frac{1}{D}. \quad (\text{A31})$$

Therefore, convergence also implies convergence in probability, and under additional conditions on the second moments, convergence in mean is also obtained. Using the theorem on increasing stochastic sums and the properties of halts with bounded overshoot, it can be shown that:

$$\mathbb{E}_1[\tau - \nu] = \frac{H + \mathbb{E}_1[R]}{D}, \quad (\text{A32})$$

where R is the “overshoot” of the sum S_m through the level H , that is, $R := S_{\tau-\nu} - H$. Under standard assumptions (non-atomicity of the ℓ_i distribution, finite moments), the distribution R has a finite expectation, independent of large H by the ladder height theorem and the Renewal theorem (see classical results on “overshoot” for random walks [65]). Hence:

$$\mathbb{E}_1[\tau - \nu] = \frac{H}{D} + O(1) \quad (H \rightarrow \infty), \quad (\text{A33})$$

which gives the asymptotics $\mathbb{E}_1 \sim \frac{H}{D}$.

Alternatively, applying Wald’s principle, for a finite stopping time τ' (relative to the onset of the shift), we have, with integrability,

$$\mathbb{E}_1[S_{\tau'}] = D \cdot \mathbb{E}_1[\tau']. \quad (\text{A34})$$

Setting $\tau' = \tau - \nu$ and taking into account that $S_{\tau'} = H + R$, we obtain:

$$D \cdot \mathbb{E}_1[\tau'] = H + \mathbb{E}_1[R] \Rightarrow \mathbb{E}_1[\tau'] = \frac{H + \mathbb{E}_1[R]}{D}. \quad (\text{A35})$$

As $H \rightarrow \infty$, the contribution $\mathbb{E}_1[R]$ is constant, therefore $\mathbb{E}_1 \sim \frac{H}{D}$.

To study the ARL behavior in the absence of change, i.e., under H_0 , it is assumed that under H_0 the mean increment $\mathbb{E}_0[\ell_i]$ is negative (in the typical setting $\ell_i = \log\left(\frac{f_1}{f_0}\right)$, then $\mathbb{E}_0[\ell_i] = -D_{KL}(f_0 \parallel f_1) < 0$). An important observation is how the likelihood ratio function, ℓ_i , satisfies the property,

$$\mathbb{E}_0[\exp(\ell_i)] = \int \frac{f_1(y)}{f_0(y)} \cdot f_0(y) dy = \int f_1(y) dy = 1. \quad (\text{A36})$$

Therefore, the value:

$$M_n := \exp(S_n) = \exp\left(\sum_{i=1}^n \ell_i\right) \quad (\text{A37})$$

is a positive martingale with respect to the measure P_0 : $\mathbb{E}_0[M_n] = 1$ and $\mathbb{E}_0[M_{n+1} \mid \mathcal{F}_n] = M_n$. Using the supplementary stopping theorem [66], we consider the stopping point τ (the instant of first exceeding H) and the truncated stopping point $\tau \wedge n$ as:

$$\mathbb{E}_0[M_{\tau \wedge n}] = \mathbb{E}_0[M_0] = 1. \quad (\text{A38})$$

Since $M_{\tau \wedge n} \geq \exp(H) \cdot \mathbf{1}\{\tau \leq n\}$ (if the process has reached the threshold before n , then the exponent of the sum is not less than $\exp(H)$), we obtain:

$$1 = \mathbb{E}_0[M_{\tau \wedge n}] \geq \exp(H) \cdot \mathbb{P}_0(\tau \leq n). \quad (\text{A39})$$

Skipping $n \rightarrow \infty$, we have:

$$\mathbb{P}_0(\tau < \infty) \leq \exp(-H), \quad (\text{A40})$$

which gives a fast upper bound on the ever-growing probability of generating a false alarm. For estimating the mean length to false alarm (ARL), more precise statements are provided by the Cramer–Lundberg [67] or Siegmund [68] results, i.e., there exists a constant $C > 0$, such that for large H :

$$\mathbb{E}_0[\tau] \sim C \cdot \exp(H). \quad (\text{A41})$$

The key step is to find the exponential parameter $\theta^* > 0$, the solution to the equation $\Lambda(\theta) = 0$, where $\Lambda(\theta) = -\log(\mathbb{E}_0[\exp(\theta \cdot \ell_1)])$. In this specific setting, where $\ell_1 = \log\left(\frac{f_1}{f_0}\right)$, already at $\theta = 1$ we have:

$$\mathbb{E}_0[\exp(1 \cdot \ell_1)] = 1, \quad (\text{A42})$$

that is, $\Lambda(1) = 0$ and one of the solutions θ^* is equal to 1. According to the standard theory of rare events and ruin-probability (or renewal) estimates (see Siegmund, Tartakovsky et al. [69]), the exponential growth of ARL is given by this θ^* . As a result,

$$\log(\mathbb{E}_0[\tau]) \sim \theta^* \cdot H = 1 \cdot H, \quad (\text{A43})$$

where $\mathbb{E}_0[\tau]$ grows exponentially as $\exp(H)$ (with a multiplier constant C depending on the distribution and overshoot). Therefore, for a given required ARL level Γ , the choice:

$$H \approx \log(\Gamma) + O(1) \quad (\text{A44})$$

ensures that $\mathbb{E}_0[\tau] \geq \Gamma$. Combining this with the delay estimate $\mathbb{E}_1[\tau] \sim \frac{H}{D}$, a threshold design rule is obtained to ensure ARL Γ and minimize delay, H is chosen to be approximately equal to $\log(\Gamma)$, then

$$\mathbb{E}_1[\tau] \sim \frac{\log(\Gamma)}{D}. \quad (\text{A45})$$

The study of asymptotic optimality is based on the more formal statement that CUSUM minimizes (in the asymptotic sense, as $\Gamma \rightarrow \infty$) the delay according to the Lorden or Pollack criterion, which is a classical theorem in the theory of quickest change detection (see Lorden (1971) [70], Moustakides (1986) [71], etc.). The essence is that among procedures with the ARL constraint $\mathbb{E}_\infty[\tau] \geq \Gamma$, CUSUM achieves the minimum possible first-approximation delay $\sim \frac{\log(\Gamma)}{D}$ (and provides a minimum upper bound for the worst-case delay). Thus, combining the obtained results, it is found that for a large threshold H ,

$$\mathbb{E}_1[\tau] = \frac{H + O(1)}{D}, \log(\mathbb{E}_0[\tau]) = H + O(1), \quad (\text{A46})$$

and therefore, for a given ARL Γ , the optimal threshold is chosen as $H \approx \log(\Gamma)$, which offers delays $\mathbb{E}_1[\tau] = \frac{\log(\Gamma)}{D}$. The theorem is proven. $\square$

Figure A3 shows the diagrams of the CUSUM paths (accumulated S_n) for the model with Λ_0 and Λ_1 . In Figure A3, the dotted line shows the theoretical straight line of the average drift $D \cdot n$, the dashed horizontal line shows the threshold H , and the vertical line indicates $n^* = \frac{H}{D}$.

Thus, Figure A3 displays the individual trajectories of the accumulated LLR S_n after switching to the model with increased intensity (simulations under H_1), and the dashed line represents the deterministic mean evolution of $D \cdot n$. The vertical line marks the theoretical expected detection time $n^* = \frac{H}{D}$. From Figure 8, it is evident that random trajectories are concentrated around the mean and cross the threshold H approximately in the vicinity of n^* , which confirms the asymptotic behavior of $\mathbb{E}_1[\tau] \sim \frac{H}{D}$.

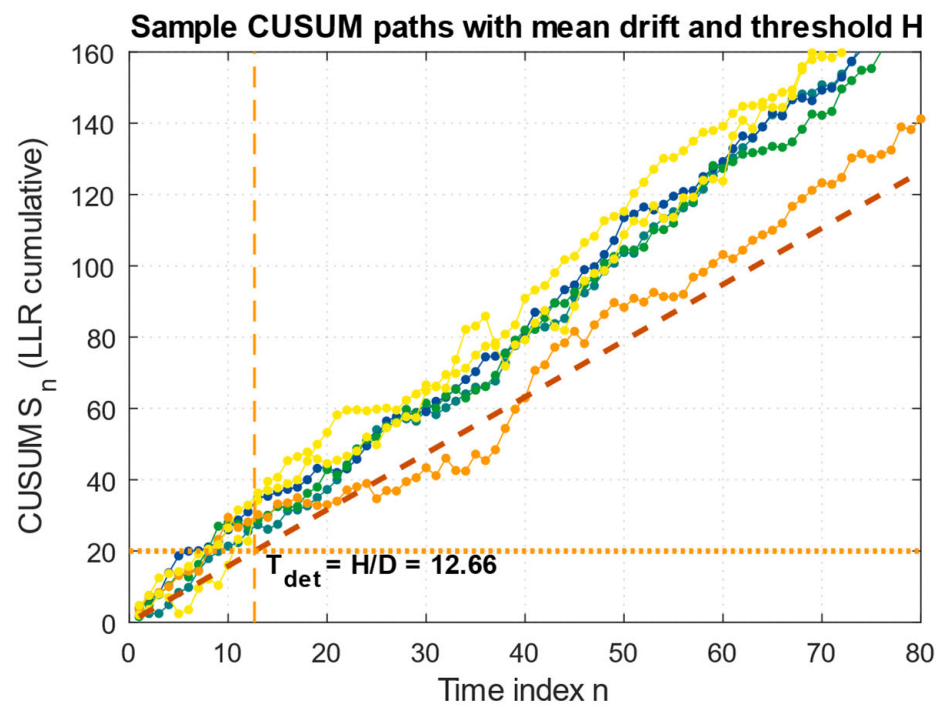


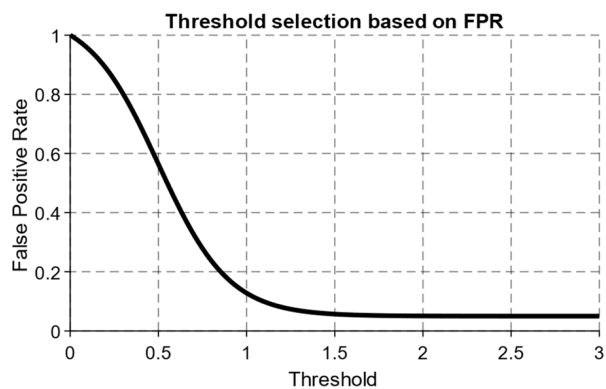
Figure A3. CUSUM (cumulative S_n) path diagrams.

Appendix B

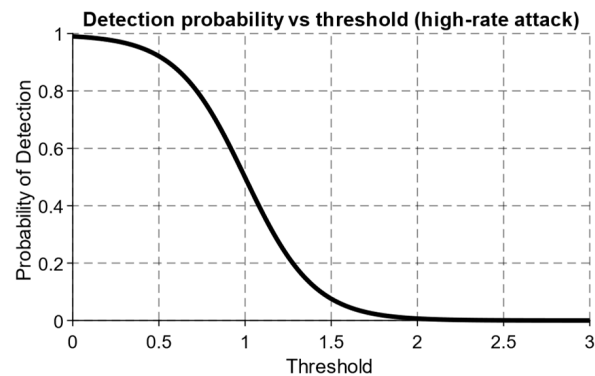
The studies' results of the proposed anomaly detection method's effectiveness in various network attack scenarios are presented in Figure A4, which demonstrates the detection probability dependence and false positives on the threshold, the delays distribution, and the models' comparative sensitivity for different types of anomalies.

Figure A4a demonstrates a monotonic decrease in FPR with increasing threshold, confirming the validity of the thresholding procedure used. Threshold values above 1.5 consistently maintain a low false alarm rate, making them ideal for use in noisy network environments. According to Figure A4b–e, all scenarios show a systematic decrease in detection probability as the threshold increases, but the rate of degradation varies. In this context, high-intensity and bursty attacks maintain a high P_d even with strict thresholds. Conversely, for low-rate and stealth attacks, the detection curve drops much faster, reflecting the low energy density of these signals and the high sensitivity of detection to the chosen threshold. The latency distributions (Figure A4f–i) show that for intensive attacks, the median and latency spread are minimal, indicating a rapid increase in anomaly indicators. However, it is important to note that, in contrast, in low-rate and stealth scenarios, latency has a wide range and is skewed toward higher values, reflecting the need to accumulate longer-term statistics before making a decision. Figure A4j shows the detection probabilities for all scenarios at a single threshold, allowing for a comparison of their sensitivities under identical operating conditions. The differences between the values highlight that low-rate and stealth attacks remain the most difficult to detect at a fixed threshold, requiring improved architecture or expanded feature space.

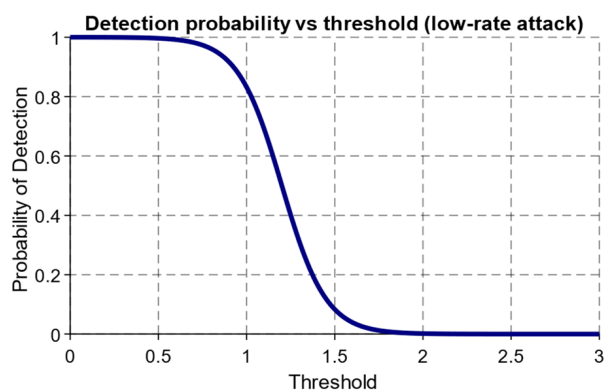
The presented results demonstrate the validity of the threshold selection procedure based on target false alarm rates. In this scenario, the FPR decreases monotonically as the threshold increases, enabling a clear determination of the operating point. At the same time, the detection probabilities for each scenario, calculated over multiple runs, exhibit significant variability in sensitivity. In this context, high-intensity and bursty attacks are consistently detected with high probabilities across a wide range of thresholds. At the same time, low-frequency and stealthy scenarios demonstrate a significant decrease in P_d with increasing threshold. The delay distribution analysis reveals that for high-intensity attacks, the median delay is low and there is little variability between runs.



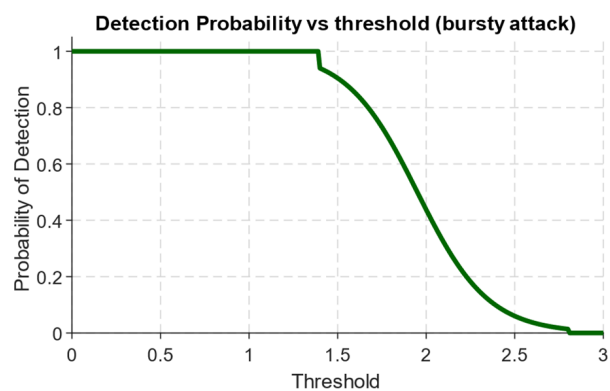
(a)



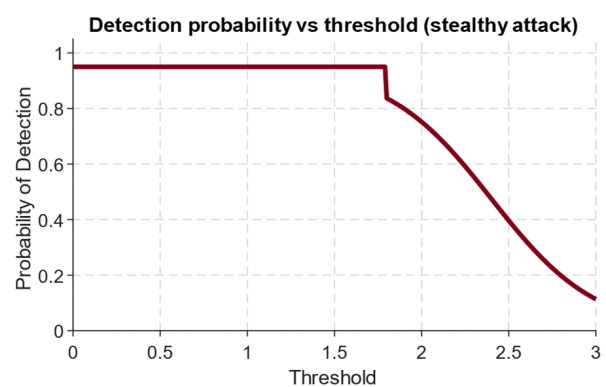
(b)



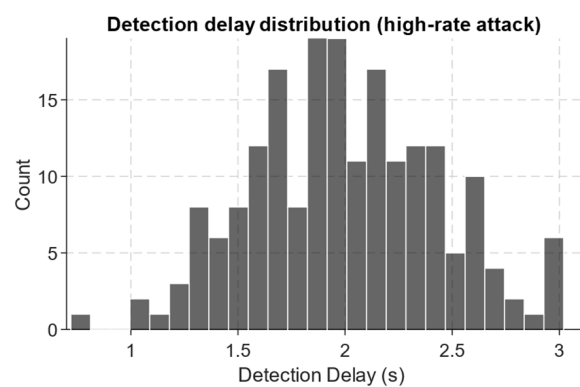
(c)



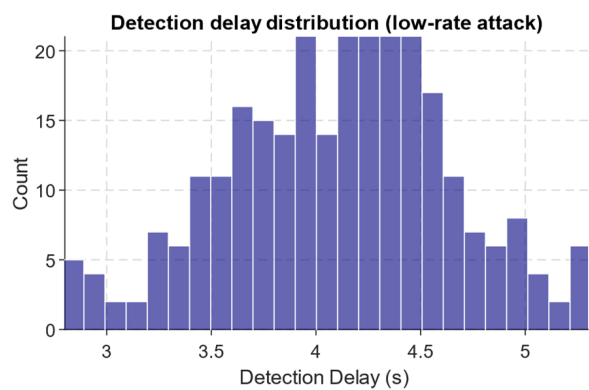
(d)



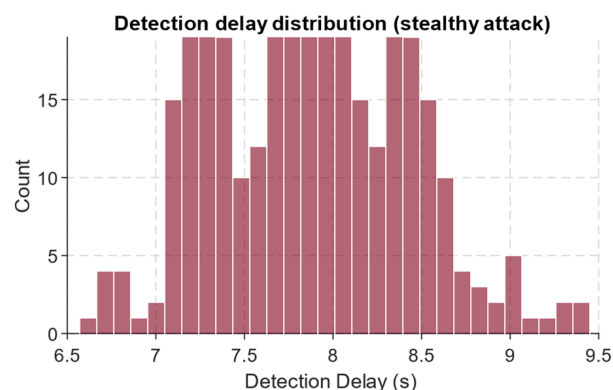
(e)



(f)



(g)



(h)

Figure A4. Cont.

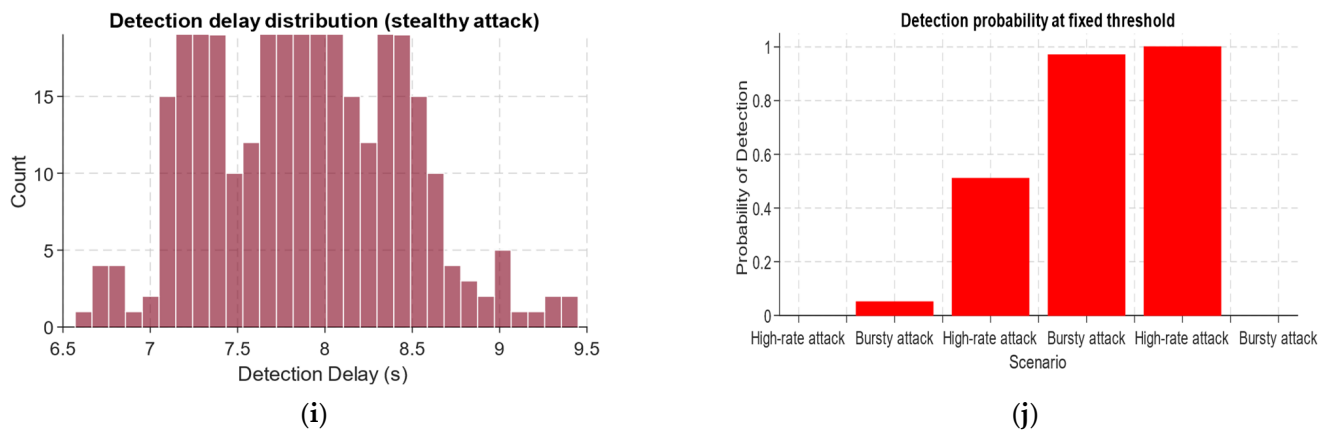


Figure A4. Results of additional studies: (a) Threshold vs. false positive rate diagram; (b) detection probability vs. threshold diagram (high-rate attack); (c) detection probability vs. threshold diagram (low-rate attack); (d) detection probability vs. threshold diagram (bursty attack); (e) detection probability vs. threshold diagram (stealthy attack); (f) detection delay distribution histogram (high-rate attack); (g) detection delay distribution histogram (low-rate attack); (h) detection delay distribution histogram (bursty attack); (i) detection delay distribution histogram (stealthy attack); (j) detection probability at fixed threshold histogram.

In contrast, for stealthy scenarios, the distribution shifts toward higher delays due to the small amplitude of disturbances relative to the background process. Thus, the resulting degradation of detection in low-frequency scenarios is explained by the insufficient expressiveness of the current feature set and the low energy density of attack episodes. Therefore, to potentially improve the obtained results, extended multi-scale feature extraction, the inclusion of adaptive spectral components, or the use of architectures with long-term memory (e.g., TCN with dilated convolutions or CNN-Transformer hybrids) are necessary, increasing sensitivity to subtle structures in the time series.

Appendix C

Figure A5 shows the detectors' performance simulation evaluation on partial PCAP results, which includes the time series of the packet intensity and detection rate (Figure A5a), the ROC curve with AUC (Figure A5b), the distribution of detection latencies by scenario (Figure A5c), and the resource consumption profile during acquisition processing (Figure A5d).

The time series diagram (Figure A5a) shows that detection rate peaks clearly correlate with areas of high-intensity and bursty traffic, confirming the method's sensitivity to pronounced anomalies. However, for the low-rate episode, the score remains at background levels, explaining its failure. The ROC curve (Figure A5b) ($AUC \approx 0.68$) demonstrates the moderate discrimination power of the current feature space. In this context, with a tight threshold setting, a low FPR can be achieved at a significant drop cost in Pd for subtle attacks, requiring a compromise or feature enhancement. The latency distribution across runs (Figure A5c) demonstrates a near-zero median latency for high-rate attacks and a small but stable latency for burst episodes, while the low-rate scenario is not detected at all in most runs. This indicates high variability and the need for statistical accumulation for reliable detection. The resource consumption profile (Figure A5d) shows clear CPU and memory peaks during periods of intense traffic bursts, highlighting the need for buffering mechanisms, backpressure, and processing capacity calculation in practical detector deployments.

Experimental analysis demonstrated that the developed method effectively detects pronounced anomalies. In this context, high detection probabilities and low median delays are observed for high-intensity and impulse attacks, indicating good sensitivity to large traffic level fluctuations. At the same time, low-rate and stealth attacks are not detected in the current configuration. The main reasons for this are the low signal amplitude compared to the background variability (low SNR) and the distributed, source-smeared nature of the episodes, which causes simple statistical features (packet count, current speed) to lose their discriminative power. An AUC value of 0.68 confirms moderate overall discriminative power. It is noted that by recalibrating the threshold, an acceptable compromise

between Pd and FPR can be achieved. However, reliable detection of stealth episodes requires not only a threshold shift (resulting in an increase in FPR) but also a feature representation strengthening. Furthermore, resource consumption profiles record predictable CPU and memory peaks during traffic surges, so practical deployments should be accompanied by measures for buffering, backpressure, and stream aggregation, as well as processing capacity monitoring.

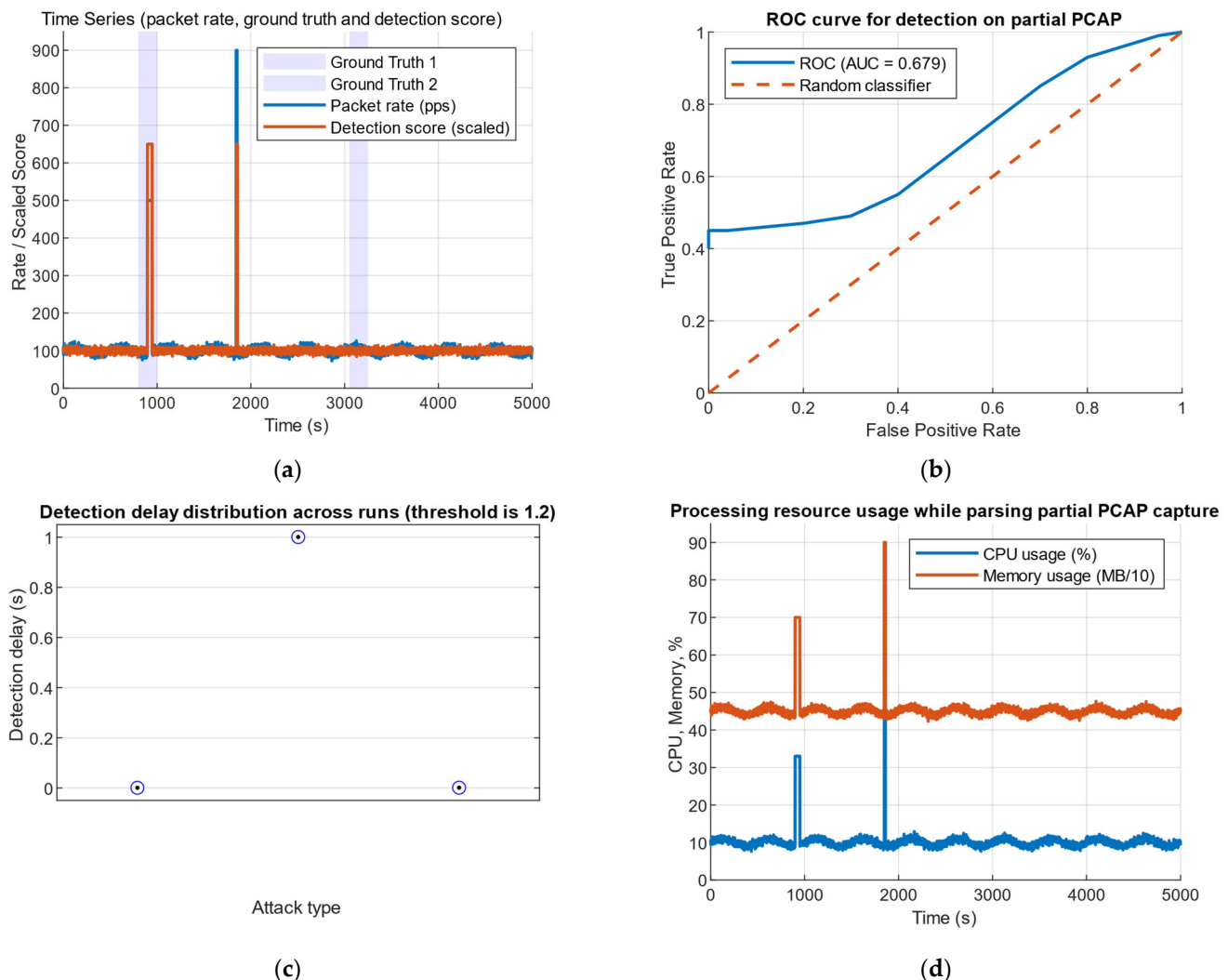


Figure A5. Results of additional studies: (a) the time series diagram; (b) the ROC curve; (c) the latency distribution across runs; (d) the resource consumption profile.

References

1. Djanie, K.S.; Tutu, T.E.; Dzisi, G.J. A Proposed DoS Detection Scheme for Mitigating DoS Attack Using Data Mining Techniques. *Computers* **2019**, *8*, 85. [\[CrossRef\]](#)
2. Wang, Y.; Ding, J.; Zhang, T.; Xiao, Y.; Hei, X. From Replay to Regeneration: Recovery of UDP Flood Network Attack Scenario Based on SDN. *Mathematics* **2023**, *11*, 1897. [\[CrossRef\]](#)
3. Serrano Mamolar, A.; Salvá-García, P.; Chirivella-Perez, E.; Pervez, Z.; Alcaraz Calero, J.M.; Wang, Q. Autonomic Protection of Multi-Tenant 5G Mobile Networks against UDP Flooding DDoS Attacks. *J. Netw. Comput. Appl.* **2019**, *145*, 102416. [\[CrossRef\]](#)
4. Murtuza, S.; Asawa, K. Early Prevention and Mitigation of Link Flooding Attacks in Software Defined Networks. *J. Netw. Comput. Appl.* **2024**, *224*, 103832. [\[CrossRef\]](#)
5. Rahmani, H.; Sahli, N.; Kamoun, F. DDoS Flooding Attack Detection Scheme Based on F-Divergence. *Comput. Commun.* **2012**, *35*, 1380–1391. [\[CrossRef\]](#)
6. Horak, T.; Strelec, P.; Huraj, L.; Tanuska, P.; Vaclavova, A.; Kebisek, M. The Vulnerability of the Production Line Using Industrial IoT Systems under DDoS Attack. *Electronics* **2021**, *10*, 381. [\[CrossRef\]](#)

7. Chen, C.-L.; Lai, J.L. An Experimental Detection of Distributed Denial of Service Attack in CDX 3 Platform Based on Snort. *Sensors* **2023**, *23*, 6139. [[CrossRef](#)] [[PubMed](#)]
8. Alashhab, Z.R.; Anbar, M.; Singh, M.M.; Hasbullah, I.H.; Jain, P.; Al-Amiedy, T.A. Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy. *Appl. Sci.* **2022**, *12*, 12441. [[CrossRef](#)]
9. Hosseinzadeh, S.; Amirmazlaghani, M.; Shajari, M. An Aggregated Statistical Approach for Network Flood Detection Using Gamma-Normal Mixture Modeling. *Comput. Commun.* **2020**, *152*, 137–148. [[CrossRef](#)]
10. Galeano-Brajones, J.; Carmona-Murillo, J.; Valenzuela-Valdés, J.F.; Luna-Valero, F. Detection and Mitigation of DoS and DDoS Attacks in IoT-Based Stateful SDN: An Experimental Approach. *Sensors* **2020**, *20*, 816. [[CrossRef](#)]
11. Nedyalkov, I. Studying the Impact of a UDP DoS Attack on the Parameters of VoIP Voice and Video Streams. *Future Internet* **2025**, *17*, 139. [[CrossRef](#)]
12. Alturki, B.; Alsulami, A.A. Semi-Supervised Learning with Entropy Filtering for Intrusion Detection in Asymmetrical IoT Systems. *Symmetry* **2025**, *17*, 973. [[CrossRef](#)]
13. AlArnaout, Z.; Mostafa, N.; Alabed, S.; Aly, W.H.F.; Shdefat, A. RAPT: A Robust Attack Path Tracing Algorithm to Mitigate SYN-Flood DDoS Cyberattacks. *Sensors* **2022**, *23*, 102. [[CrossRef](#)] [[PubMed](#)]
14. Muragaa, W.H.A. A Hybrid Scheme for Detecting and Preventing Single Packet Low-Rate DDoS and Flooding DDoS Attacks in SDN. In Proceedings of the 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 21–23 May 2023; pp. 707–712. [[CrossRef](#)]
15. Harayama, M.; Miyagawa, N. Intelligent Throughput Stabilizer for UDP-Based Rate-Control Communication System. *Intell. Conver. Netw.* **2021**, *2*, 205–212. [[CrossRef](#)]
16. Chen, Y.; Hou, J.; Li, Q.; Long, H. DDoS Attack Detection Based on Random Forest. In Proceedings of the 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 18–20 December 2020; pp. 328–334. [[CrossRef](#)]
17. Ashwin, A.; Santhosh, V.; Thirupathi Venkatesh, R.; Gowthami, N.; Tamilselvi, S. Detection and Mitigation of DDoS Attack in SDN Using Feature Based SVM and Decision Tree Approach. In Proceedings of the 2024 International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS), Bengaluru, India, 17–18 December 2024; pp. 325–330. [[CrossRef](#)]
18. Liu, L.; Yu, W.; Wu, Z.; Peng, S. XGBoost-Based Detection of DDoS Attacks in Named Data Networking. *Future Internet* **2025**, *17*, 206. [[CrossRef](#)]
19. Shieh, C.-S.; Nguyen, T.-T.; Horng, M.-F. Detection of Unknown DDoS Attack Using Convolutional Neural Networks Featuring Geometrical Metric. *Mathematics* **2023**, *11*, 2145. [[CrossRef](#)]
20. Polat, O.; Türkoğlu, M.; Polat, H.; Oyucu, S.; Üzen, H.; Yardımcı, F.; Aksöz, A. Multi-Stage Learning Framework Using Convolutional Neural Network and Decision Tree-Based Classification for Detection of DDoS Pandemic Attacks in SDN-Based SCADA Systems. *Sensors* **2024**, *24*, 1040. [[CrossRef](#)]
21. Chartuni, A.; Márquez, J. Multi-Classifer of DDoS Attacks in Computer Networks Built on Neural Networks. *Appl. Sci.* **2021**, *11*, 10609. [[CrossRef](#)]
22. Yousuf, O.; Mir, R.N. DDoS Attack Detection in Internet of Things Using Recurrent Neural Network. *Comput. Electr. Eng.* **2022**, *101*, 108034. [[CrossRef](#)]
23. Efendi, R.; Wahyono, T.; Widiyari, I.R. DBSCAN SMOTE LSTM: Effective Strategies for Distributed Denial of Service Detection in Imbalanced Network Environments. *Big Data Cogn. Comput.* **2024**, *8*, 118. [[CrossRef](#)]
24. Chen, L.; Wang, Z.; Huo, R.; Huang, T. An Adversarial DBN-LSTM Method for Detecting and Defending against DDoS Attacks in SDN Environments. *Algorithms* **2023**, *16*, 197. [[CrossRef](#)]
25. Vladov, S.; Vysotska, V.; Sokurenko, V.; Muzychuk, O.; Nazarkevych, M.; Lytvyn, V. Neural Network System for Predicting Anomalous Data in Applied Sensor Systems. *Appl. Syst. Innov.* **2024**, *7*, 88. [[CrossRef](#)]
26. Safarov, F.; Basak, M.; Nasimov, R.; Abdusalomov, A.; Cho, Y.I. Explainable Lightweight Block Attention Module Framework for Network-Based IoT Attack Detection. *Future Internet* **2023**, *15*, 297. [[CrossRef](#)]
27. Sapkota, B.; Ray, A.; Yadav, M.K.; Dawadi, B.R.; Joshi, S.R. Machine Learning-Based Attack Detection and Mitigation with Multi-Controller Placement Optimization over SDN Environment. *J. Cybersecur. Priv.* **2025**, *5*, 10. [[CrossRef](#)]
28. Alaghbari, K.A.; Lim, H.-S.; Saad, M.H.M.; Yong, Y.S. Deep Autoencoder-Based Integrated Model for Anomaly Detection and Efficient Feature Extraction in IoT Networks. *IoT* **2023**, *4*, 345–365. [[CrossRef](#)]
29. Harrou, F.; Bouyeddou, B.; Dairi, A.; Sun, Y. Exploiting Autoencoder-Based Anomaly Detection to Enhance Cybersecurity in Power Grids. *Future Internet* **2024**, *16*, 184. [[CrossRef](#)]
30. Hou, Y.; He, R.; Dong, J.; Yang, Y.; Ma, W. IoT Anomaly Detection Based on Autoencoder and Bayesian Gaussian Mixture Model. *Electronics* **2022**, *11*, 3287. [[CrossRef](#)]
31. Paolini, D.; Dini, P.; Soldaini, E.; Saponara, S. One-Class Anomaly Detection for Industrial Applications: A Comparative Survey and Experimental Study. *Computers* **2025**, *14*, 281. [[CrossRef](#)]
32. Berrios, S.; Garcia, S.; Hermosilla, P.; Allende-Cid, H. A Machine-Learning-Based Approach for the Detection and Mitigation of Distributed Denial-of-Service Attacks in Internet of Things Environments. *Appl. Sci.* **2025**, *15*, 6012. [[CrossRef](#)]

33. Luo, Y.-Y.; Chiu, Y.-H.; Cheng, C.-H. Detection and Mitigation in IoT Ecosystems Using oneM2M Architecture and Edge-Based Machine Learning. *Future Internet* **2025**, *17*, 411. [\[CrossRef\]](#)
34. Khamaiseh, S.; Al-Alaj, A.; Adnan, M.; Alomari, H.W. The Robustness of Detecting Known and Unknown DDoS Saturation Attacks in SDN via the Integration of Supervised and Semi-Supervised Classifiers. *Future Internet* **2022**, *14*, 164. [\[CrossRef\]](#)
35. Hoang, X.; Pham, H. A Review on Hot-IP Finding Methods and Its Application in Early DDoS Target Detection. *Future Internet* **2016**, *8*, 52. [\[CrossRef\]](#)
36. Baird, I.; Ghaleb, B.; Wadhaj, I.; Russell, G.; Buchanan, W.J. Securing IoT: Mitigating Sybil Flood Attacks with Bloom Filters and Hash Chains. *Electronics* **2024**, *13*, 3467. [\[CrossRef\]](#)
37. Aydin, B.; Aydin, H.; Gormus, S. Mitigating UDP Flooding Attacks in 6TiSCH Networks: A Blacklisting and Routing Isolation Approach. In Proceedings of the 2025 33rd Signal Processing and Communications Applications Conference (SIU), Istanbul, Türkiye, 25–28 June 2025; pp. 1–4. [\[CrossRef\]](#)
38. Wei, H.-C.; Tung, Y.-H.; Yu, C.-M. Counteracting UDP Flooding Attacks in SDN. In Proceedings of the 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Republic of Korea, 6–10 June 2016; pp. 367–371. [\[CrossRef\]](#)
39. Mutu, L.; Saleh, R.; Matrawy, A. Improved SDN Responsiveness to UDP Flood Attacks. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; pp. 715–716. [\[CrossRef\]](#)
40. Tung, Y.-H.; Wei, H.-C.; Ti, Y.-W.; Tsou, Y.-T.; Saxena, N.; Yu, C.-M. Counteracting UDP Flooding Attacks in SDN. *Electronics* **2020**, *9*, 1239. [\[CrossRef\]](#)
41. Almeghle, S.M.; AL-Ghamdi, A.A.-M.; Ramzan, M.S.; Ragab, M. Machine Learning-Based DoS Amplification Attack Detection against Constrained Application Protocol. *Appl. Sci.* **2023**, *13*, 7391. [\[CrossRef\]](#)
42. Syafiuddin, N.H.; Mandala, S.; Cahyani, N.D.W. Detection Syn Flood and UDP Lag Attacks Based on Machine Learning Using AdaBoost. In Proceedings of the 2023 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, 9–10 August 2023; pp. 36–41. [\[CrossRef\]](#)
43. Najafimehr, M.; Zarifzadeh, S.; Mostafavi, S. DDoS Attacks and Machine-learning-based Detection Methods: A Survey and Taxonomy. *Eng. Rep.* **2023**, *5*, e12697. [\[CrossRef\]](#)
44. Hasan, A.; Khan, M.A.; Shabir, B.; Munir, A.; Malik, A.W.; Anwar, Z.; Ahmad, J. Forensic Analysis of Blackhole Attack in Wireless Sensor Networks/Internet of Things. *Appl. Sci.* **2022**, *12*, 11442. [\[CrossRef\]](#)
45. Wiranata, A.; Karna, N.; Irawan, A.; Prakoso, I.A. Implementation and Analysis of Network Security in Raspberry Pi against DOS Attack with HIPS Snort. In Proceedings of the 2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE), Jakarta, Indonesia, 16 February 2023; pp. 892–896. [\[CrossRef\]](#)
46. Hameed, S.; Ahmed Khan, H. SDN Based Collaborative Scheme for Mitigation of DDoS Attacks. *Future Internet* **2018**, *10*, 23. [\[CrossRef\]](#)
47. Bijalwan, A.; Wazid, M.; Pilli, E.S.; Joshi, R.C. Forensics of Random-UDP Flooding Attacks. *J. Netw.* **2015**, *10*, 287. [\[CrossRef\]](#)
48. Tolanur, J.; Chaudhari, S. DDoS Attacks Analysis with Cyber Data Forensics Using Weighted Logistic Regression and Random Forest. In Proceedings of the 2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT), Dehradun, India, 17–18 March 2023; pp. 1–6. [\[CrossRef\]](#)
49. Lin, H.-C.; Wang, P.; Lin, W.-H. Implementation of a PSO-Based Security Defense Mechanism for Tracing the Sources of DDoS Attacks. *Computers* **2019**, *8*, 88. [\[CrossRef\]](#)
50. Sheeraz, M.; Durad, M.H.; Paracha, M.A.; Mohsin, S.M.; Kazmi, S.N.; Maple, C. Revolutionizing SIEM Security: An Innovative Correlation Engine Design for Multi-Layered Attack Detection. *Sensors* **2024**, *24*, 4901. [\[CrossRef\]](#) [\[PubMed\]](#)
51. Khan, M.A.; Nasralla, M.M.; Umar, M.M.; Ghani-Ur-Rehman; Khan, S.; Choudhury, N. An Efficient Multilevel Probabilistic Model for Abnormal Traffic Detection in Wireless Sensor Networks. *Sensors* **2022**, *22*, 410. [\[CrossRef\]](#) [\[PubMed\]](#)
52. Karmous, N.; Aouelelyne, M.O.-E.; Abdelkader, M.; Romdhani, L.; Youssef, N. Software-Defined-Networking-Based One-versus-Rest Strategy for Detecting and Mitigating Distributed Denial-of-Service Attacks in Smart Home Internet of Things Devices. *Sensors* **2024**, *24*, 5022. [\[CrossRef\]](#) [\[PubMed\]](#)
53. Sarvade, V.P.; Kulkarni, S.A.; Raj, C.V. A Hybrid Classical-Quantum Neural Network Model for DDoS Attack Detection in Software-Defined Vehicular Networks. *Information* **2025**, *16*, 722. [\[CrossRef\]](#)
54. Shaikh, T. Sequential Heuristic Model for DDOS Attack Detection in Software-Defined Network. In Proceedings of the 2023 20th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Murree, Pakistan, 22–25 August 2023; pp. 614–620. [\[CrossRef\]](#)
55. Sangodoyin, A.O.; Akinsolu, M.O.; Pillai, P.; Grout, V. Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning. *IEEE Access* **2021**, *9*, 122495–122508. [\[CrossRef\]](#)
56. Koyama, S. Coarse-Grained Hawkes Processes. *Entropy* **2025**, *27*, 555. [\[CrossRef\]](#)
57. Berti, P.; Pratelli, L.; Rigo, P. A Central Limit Theorem for Predictive Distributions. *Mathematics* **2021**, *9*, 3211. [\[CrossRef\]](#)

58. Bouyeddou, B.; Kadri, B.; Harrou, F.; Sun, Y. Nonparametric Kullback-Leibler Distance-Based Method for Networks Intrusion Detection. In Proceedings of the 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), Sakheer, Bahrain, 26–27 October 2020; pp. 1–5. [\[CrossRef\]](#)
59. Oyucu, S.; Polat, O.; Türkoğlu, M.; Polat, H.; Aksöz, A.; Ağdaş, M.T. Ensemble Learning Framework for DDoS Detection in SDN-Based SCADA Systems. *Sensors* **2023**, *24*, 155. [\[CrossRef\]](#)
60. Bodyanskiy, Y.; Deineko, A.; Skorik, V.; Brodetskyi, F. Deep Neural Network with Adaptive Parametric Rectified Linear Units and Its Fast Learning. *Int. J. Comput.* **2022**, *21*, 11–18. [\[CrossRef\]](#)
61. Ratan Kumar, S.; Valli Kumari, V.; Raju, K.V.S.V.N. Multi-Core Parallel Processing Technique to Prepare the Time Series Data for the Early Detection of DDoS Flooding Attacks. In Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 17–19 March 2021.
62. Turchenko, V.; Kochan, V.; Sachenko, A. Estimation of Computational Complexity of Sensor Accuracy Improvement Algorithm Based on Neural Networks. *Lect. Notes Comput. Sci.* **2001**, *2130*, 743–748. [\[CrossRef\]](#)
63. Harrou, F.; Bouyeddou, B.; Sun, Y.; Kadri, B. A Method to Detect DOS and DDOS Attacks Based on Generalized Likelihood Ratio Test. In Proceedings of the 2018 International Conference on Applied Smart Systems (ICASS), Medea, Algeria, 24–25 November 2018; pp. 1–6. [\[CrossRef\]](#)
64. Elsherif, A.K.; Haj Ahmad, H.; Aboshady, M.; Mostafa, B. Fuzzy Hypothesis Testing for Radar Detection: A Statistical Approach for Reducing False Alarm and Miss Probabilities. *Mathematics* **2025**, *13*, 2299. [\[CrossRef\]](#)
65. Gençoğlu, M.T. Finding the Integral-Equation-Based Linear Renewal Density Equation and Analytical Solutions. *Symmetry* **2025**, *17*, 453. [\[CrossRef\]](#)
66. Entwistle, H.N.; Lustri, C.J.; Sofronov, G.Y. Asymptotic Duration for Optimal Multiple Stopping Problems. *Mathematics* **2024**, *12*, 652. [\[CrossRef\]](#)
67. Avram, F.; Goreac, D.; Renaud, J.-F. The Løkka–Zervos Alternative for a Cramér–Lundberg Process with Exponential Jumps. *Risks* **2019**, *7*, 120. [\[CrossRef\]](#)
68. Turchenko, V.; Grandinetti, L.; Sachenko, A. Parallel Batch Pattern Training of Neural Networks on Computational Clusters. In Proceedings of the 2012 International Conference on High Performance Computing & Simulation (HPCS), Madrid, Spain, 2–6 July 2012; pp. 202–208. [\[CrossRef\]](#)
69. Ko, K.-M.; Baek, J.-M.; Seo, B.-S.; Lee, W.-B. Comparative Study of AI-Enabled DDoS Detection Technologies in SDN. *Appl. Sci.* **2023**, *13*, 9488. [\[CrossRef\]](#)
70. Lorden, G. Procedures for reacting to a change in distribution. *Ann. Math. Stat.* **1971**, *42*, 1897–1908. [\[CrossRef\]](#)
71. Moustakides, G.V. Optimal stopping times for detecting changes in distributions. *Ann. Stat.* **1986**, *14*, 1379–1387. [\[CrossRef\]](#)
72. Su, Y.; Xiong, D.; Qian, K.; Wang, Y. A Comprehensive Survey of Distributed Denial of Service Detection and Mitigation Technologies in Software-Defined Network. *Electronics* **2024**, *13*, 807. [\[CrossRef\]](#)
73. Zeng, H.; Kong, F. Active Learning: Encoder-Decoder-Outlayer and Vector Space Diversification Sampling. *Mathematics* **2023**, *11*, 2819. [\[CrossRef\]](#)
74. Vladov, S.; Scislo, L.; Sokurenko, V.; Muzychuk, O.; Vysotska, V.; Osadchy, S.; Sachenko, A. Neural Network Signal Integration from Thermogas-Dynamic Parameter Sensors for Helicopters Turboshift Engines at Flight Operation Conditions. *Sensors* **2024**, *24*, 4246. [\[CrossRef\]](#)
75. Vladov, S.; Sachenko, A.; Sokurenko, V.; Muzychuk, O.; Vysotska, V. Helicopters Turboshift Engines Neural Network Modeling under Sensor Failure. *J. Sens. Actuator Netw.* **2024**, *13*, 66. [\[CrossRef\]](#)
76. Vladov, S.; Chyrun, L.; Muzychuk, E.; Vysotska, V.; Lytvyn, V.; Rekunen, T.; Basko, A. Intelligent Method for Generating Criminal Community Influence Risk Parameters Using Neural Networks and Regional Economic Analysis. *Algorithms* **2025**, *18*, 523. [\[CrossRef\]](#)
77. Lopes, J.F.; Barbon Junior, S.; de Melo, L.F. Online Meta-Recommendation of CUSUM Hyperparameters for Enhanced Drift Detection. *Sensors* **2025**, *25*, 2787. [\[CrossRef\]](#)
78. Sklyar, V.; Kharchenko, V. Assurance Case for Safety and Security Implementation: A Survey of Applications. *Int. J. Comput.* **2020**, *19*, 610–619. [\[CrossRef\]](#)
79. Vladov, S.; Shmelov, Y.; Yakovliev, R. Method for Forecasting of Helicopters Aircraft Engines Technical State in Flight Modes Using Neural Networks. *CEUR Workshop Proc.* **2022**, *3171*, 974–985. Available online: <https://ceur-ws.org/Vol-3171/paper70.pdf> (accessed on 10 November 2025).
80. Vladov, S.; Banasik, A.; Sachenko, A.; Kempa, W.M.; Sokurenko, V.; Muzychuk, O.; Pikiewicz, P.; Molga, A.; Vysotska, V. Intelligent Method of Identifying the Nonlinear Dynamic Model for Helicopter Turboshift Engines. *Sensors* **2024**, *24*, 6488. [\[CrossRef\]](#)
81. Golovko, V.; Egor, M.; Brich, A.; Sachenko, A. A Shallow Convolutional Neural Network for Accurate Handwritten Digits Classification. *Commun. Comput. Inf. Sci.* **2017**, *673*, 77–85. [\[CrossRef\]](#)
82. Vladov, S.; Shmelov, Y.; Yakovliev, R. Methodology for Control of Helicopters Aircraft Engines Technical State in Flight Modes Using Neural Networks. *CEUR Workshop Proc.* **2022**, *3137*, 108–125. [\[CrossRef\]](#)

83. Vladov, S.; Shmelov, Y.; Yakovliev, R.; Petchenko, M. Modified Neural Network Fault-Tolerant Closed Onboard Helicopters Turboshift Engines Automatic Control System. *CEUR Workshop Proc.* **2023**, 3387, 160–179. Available online: <https://ceur-ws.org/Vol-3387/paper13.pdf> (accessed on 12 November 2025).
84. Kashtalian, A.; Lysenko, S.; Kysil, T.; Sachenko, A.; Savenko, O.; Savenko, B. Method and Rules for Determining the Next Centralization Option in Multicomputer System Architecture. *Int. J. Comput.* **2025**, 24, 35–51. [[CrossRef](#)]
85. Bykovyy, P.; Kochan, V.; Sachenko, A.; Markowsky, G. Genetic Algorithm Implementation for Perimeter Security Systems CAD. In Proceedings of the 2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Dortmund, Germany, 6–8 September 2007; pp. 634–638. [[CrossRef](#)]
86. Vladov, S.; Shmelov, Y.; Yakovliev, R. Optimization of Helicopters Aircraft Engine Working Process Using Neural Networks Technologies. *CEUR Workshop Proc.* **2022**, 3171, 1639–1656. Available online: <https://ceur-ws.org/Vol-3171/paper117.pdf> (accessed on 16 November 2025).
87. Mulesa, O.; Povkhan, I.; Radivilova, T.; Baranovskyi, O. Devising a Method for Constructing the Optimal Model of Time Series Forecasting Based on the Principles of Competition. *East.-Eur. J. Enterp.* **2021**, 5, 6–11. [[CrossRef](#)]
88. Ablamskyi, S.; Tchobo, D.L.R.; Romaniuk, V.; Šimić, G.; Ilchyshyn, N. Assessing the Responsibilities of the International Criminal Court in the Investigation of War Crimes in Ukraine. *Novum Jus* **2023**, 17, 353–374. [[CrossRef](#)]
89. Pchelina, O.V.; Skulysh, Y.D.; Buglak, I.; Myroniuk, R.V. The International Experience of Ensuring Cybersecurity in the Country and Possibility of Its Application in Ukraine. *DIXI* **2021**, 23, 8. [[CrossRef](#)]
90. Muzychuk, O.; Salmanova, O.; Khromov, A.; Seliukov, V. Law Enforcement Agencies in the System of Subjects of Human Rights Protection and Defence. *Rev. Gestão Secr.* **2023**, 14, 15004–15019. [[CrossRef](#)]
91. Ablamskyi, S.; Nenia, O.; Drozd, V.; Havryliuk, L. Substantial Violation of Human Rights and Freedoms as a Prerequisite for Inadmissibility of Evidence. *Justicia* **2021**, 26, 47–56. [[CrossRef](#)]
92. Geche, F.; Batyuk, A.; Mulesa, O.; Voloshchuk, V. The Combined Time Series Forecasting Model. In Proceedings of the 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2020; pp. 272–275. [[CrossRef](#)]
93. Bodyanskiy, Y.; Shafronenko, A.; Pliss, I. Clusterization of Vector and Matrix Data Arrays Using the Combined Evolutionary Method of Fish Schools. *Syst. Res. Inf. Technol.* **2022**, 4, 79–87. [[CrossRef](#)]
94. Radivilova, T.; Kirichenko, L.; Alghawli, A.S.; Ageyev, D.; Mulesa, O.; Baranovskyi, O.; Ilkov, A.; Kulbachnyi, V.; Bondarenko, O. Statistical and Signature Analysis Methods of Intrusion Detection. *Lect. Notes Data Eng. Commun. Technol.* **2022**, 115, 115–131. [[CrossRef](#)]
95. Abdrabo, A. Application of Online Anomaly Detection Using One-Class Classification to the Z24 Bridge. *Sensors* **2024**, 24, 7866. [[CrossRef](#)] [[PubMed](#)]
96. Li, X.; Li, R.; Liu, Y. HP-LSTM: Hawkes Process–LSTM-Based Detection of DDoS Attack for In-Vehicle Network. *Future Internet* **2024**, 16, 185. [[CrossRef](#)]
97. Mansoor, A.; Anbar, M.; Bahashwan, A.; Alabsi, B.; Rihan, S. Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller. *Systems* **2023**, 11, 296. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.