Applied Soft Computing Journal 202 (2026) 115796

Contents lists available at ScienceDirect

Applied Soft Computing

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

Detecting the invisible without knowledge: Unsupervised packet-based real-time detection of slow DoS attacks

Antonino Rullo [a,*], Enrico Cambiaso [b], Massimo Guarascio [a]

[a] Institute for High Performance Computing and Networking (CNR-ICAR), Rende, 87036, Italy
[b] Institute of Electronics, Computer and Telecommunication Engineering (CNR-IEIIT), Genoa, 16152, Italy

HIGHLIGHTS

- Intrusion detection based on unsupervised learning and packet-level traffic analysis.
- Packet sequences transformed into images and processed by a convolutional autoencoder.
- Custom preprocessing and postprocessing modules improve classification.
- Packet-based approach avoids the computational overhead of flow reconstruction.
- Performance comparable to supervised approaches.

ARTICLE INFO

Keywords:
Intrusion detection system
Slow DoS
Denial of service
Deep learning
Unsupervised learning

ABSTRACT

Slow DoS attacks are stealthy threats that operate at a much lower packet rate than flooding-based DoS attacks, allowing them to blend seamlessly with normal traffic and evade detection. While several solutions have been proposed in the literature, most rely on labeled attack data, which is often unavailable, or depend on flow-level features, which are resource-intensive and may not enable timely detection. To overcome these limitations, we define a machine learning-based intrusion detection system that requires no prior knowledge of malicious traffic and operates at the packet level, enabling real-time detection. Unlike flow-level approaches, packet-based analysis does not require maintaining long-lived per-flow state or reconstructing high-volume traffic aggregates, which can be computationally expensive in high-throughput environments. Our solution is based on a three-phase approach: First, the preprocessing phase normalizes incoming network traffic using a risk-aware strategy designed to highlight deviations that may indicate malicious behavior. Next, the anomaly-detection phase employs an unsupervised neural model to distinguish normal traffic from anomalous patterns. Finally, the postprocessing phase refines the model's output and, whenever an attack is detected, identifies the attacker's IP address. We evaluate the proposed IDS against three real-world datasets under various slow DoS attacks. Experimental results demonstrate detection performance comparable to that of supervised methods.

1. Introduction

Denial of Service (DoS) attacks are malicious attempts to prevent legitimate users from accessing a service by overwhelming it with excessive traffic or resource-consuming requests. DoS attacks vary in complexity, ranging from simple flooding (volumetric attacks) to sophisticated protocol exploits (application-layer attacks). Volumetric attacks focus on exhausting the target's bandwidth or network resources by generating massive amounts of traffic, leading to service degradation or complete downtime. In particular, traditional volumetric DoS attacks typically exploit Network or Transport layer mechanisms. ICMP-based flooding operates at the Network layer, while UDP and TCP-based attacks (including SYN flooding) operate at the Transport layer and aim to exhaust server connection backlogs or intermediate resources rather than bandwidth alone. Application-layer attacks target web servers and applications by depleting their processing power in a slow, stealthy manner rather than overwhelming network bandwidth. These attacks gradually exhaust server resources by keeping connections open for extended periods or forcing servers to perform computationally expensive

* Corresponding author.
Email addresses: antonino.rullo@icar.cnr.it (A. Rullo), enrico.cambiaso@cnr.it (E. Cambiaso), massimo.guarascio@icar.cnr.it (M. Guarascio).

https://doi.org/10.1016/j.asoc.2026.115796
Received 15 December 2025; Received in revised form 12 May 2026; Accepted 18 June 2026
Available online 19 June 2026

operations. Due to their low-rate nature, they are often referred to as *slow DoS* attacks.

Unlike volumetric attacks, which generate noticeable traffic spikes that can be detected through rate-based monitoring, slow DoS attacks mimic normal behavior, making them much harder to identify. Their stealthy nature necessitates more advanced detection strategies, such as behavioral analysis or connection tracking. This complexity makes developing effective detection techniques particularly challenging [60].

Motivations. Most existing state-of-the-art (SOTA) intrusion detection systems (IDS) for slow DoS attacks are signature-based, relying on statistical techniques or supervised machine learning algorithms [5,9,10,12,43,55]. However, these approaches present several drawbacks. While legitimate traffic can be easily collected in large volumes, acquiring sufficient malicious traffic is far more difficult. Even when real attack data is available, it is typically much smaller in quantity than legitimate traffic, leading to class imbalance issues that can negatively impact model performance. Additionally, network traffic patterns vary significantly across different environments, making it impractical to apply labeled data from one organization to another. To overcome these limitations, some studies use simulation tools to generate synthetic attack traffic in sufficient quantities to balance the dataset [14,35,56,57,59]. However, attackers continuously modify their strategies, making it difficult for models trained on simulated data to generalize across all possible attack variations. Similarly, while other studies focus on the early detection of ongoing slow DoS attacks [1,29], they rely on the definition of static thresholds and therefore lack the flexibility to adapt to variations in local network conditions or attack behaviors.

A further limitation of state-of-the-art (SOTA) techniques for slow DoS detection is that most of them operate at the *flow level* [14,17,21,22,26,28]. In flow-based analysis, individual packets are aggregated into flows—unidirectional or bidirectional sequences of packets sharing the same source and destination IP addresses, ports, and transport protocol. Detection approaches based on this paradigm rely on aggregated statistics such as packet counts, byte volumes, or flow duration. As network traffic increases, maintaining and processing these aggregates becomes computationally expensive. Moreover, the effectiveness of such methods strongly depends on the flow sampling rate: sampling too infrequently may cause ongoing attacks to remain undetected, whereas overly frequent sampling can generate an excessive number of flow instances to process. This trade-off is particularly challenging in high-speed networks, where fine-grained flow aggregation leads to significant memory and CPU overhead.

By contrast, only a small number of recent approaches work at the *packet level*, examining traffic without any kind of aggregation [20,25,37]. This discrepancy likely arises because flow-level representations offer a more complete view of communication patterns, making them appealing for behavior-based detection, whereas packet-level features provide only a local snapshot of network activity.

Contribution. While the literature includes many flow-based methods, both supervised and unsupervised, packet-based techniques are few and all rely on supervision [20,25,37]. Interestingly, no existing method combines packet-level analysis with an unsupervised approach. To bridge this gap, we propose an unsupervised intrusion detection system for slow DoS attacks that operates at the packet level. Unsupervised methods eliminate the need for labeled data, as the detection models are trained solely on normal traffic, avoiding the costly and often impractical labeling effort required from domain experts. They are also more resilient to a broader range of attack variants, since they do not rely on predefined signatures; instead, anomalies are identified as deviations from learned benign behavior. Moreover, packet-level analysis is significantly less resource-intensive than flow-based approaches, enabling lightweight operation even in high-throughput environments and supporting real-time detection by design. This combination of unsupervised learning and lightweight packet-level analysis is highly practical for real-world applications, and offers a scalable and adaptive strategy for identifying subtle and previously unseen threats.

Our system follows a three-step approach: (1) raw network packets are converted into images; (2) a convolutional autoencoder, trained exclusively on benign traffic, classifies each image as either normal or anomalous; (3) to enhance precision, potential false positives are filtered, and the remaining anomalies are further analyzed to detect attacks and identify the originating IP addresses. While the idea of converting raw data into images is not novel in the cybersecurity domain [11], our image-generation task performed in step (1) is specifically designed as a risk-aware data preprocessing, which highlights packet features more relevant to slow DoS detection, to enhance the effectiveness of the subsequent anomaly-detection stage. The third step addresses the common challenge of high false positive rate, which often hinders the practical effectiveness of anomaly-based detection systems: based on the assumption that the anomaly rate is much lower during attack-free periods than when attacks are in progress, sporadic anomalies are treated as false positives and ignored. The remaining anomalous samples are further analyzed in search of evidence of an attack.

We show that our framework enables real-time detection by identifying slow DoS attacks in their early stages, providing sufficient time to deploy appropriate countermeasures. Our contributions can be summarized as follows:

- We introduce a novel, fast preprocessing technique specifically tailored for anomaly detection purposes, whereby categorical, continuous, and binary packet features are normalized following a risk-aware approach, with the aim of highlighting packet features more relevant to slow DoS detection.
- We design a neural architecture capable of effectively learning normal traffic patterns in an unsupervised manner, using packet-level features only. To this end, we introduce a novel loss function that amplifies the characteristics of normal traffic, simplifying the detection of anomalous traffic.
- We propose a postprocessing technique that detects misclassified samples, minimizing the negative impact of false positives.
- We validate the effectiveness of our design choices through an extensive experimental evaluation, demonstrating that: *(i)* the proposed preprocessing strategy significantly enhances detection performance; *(ii)* the neural architecture, together with the tailored loss function, outperforms alternative designs; *(iii)* the postprocessing module consistently reduces false positives, enabling more reliable attack detection and straightforward attacker identification; and *(iv)* the detection of slow DoS attacks occurs early enough to allow timely deployment of appropriate countermeasures.

The rest of the paper is organized as follows. In Section 2 we discuss related work; in Section 3 we provide some background knowledge on slow DoS attacks; in Section 4 we describe the IDS architecture and functionalities; in Section 5 we demonstrate the effectiveness of the proposed approach through an extensive set of experiments; finally, in Section 6 we conclude the paper.

2. Related work

The literature on slow DoS attack detection can be broadly categorized into two main groups: data-driven and rule-driven intrusion detection systems (IDSs). Data-driven approaches leverage machine learning algorithms to build models from historical data, which are then used to classify incoming traffic, whereas rule-driven methods rely on statistical techniques such as wavelet analysis [62], principal component analysis [67], entropy measures [63], and hidden Markov models [19], among others. In particular, several previous works (e.g., [1,29]) detect slow DoS attacks using predefined rules based on connection duration and/or low-rate traffic behaviors. However, these techniques rely exclusively on known attack patterns and do not incorporate any learning mechanism capable of adapting to local traffic characteristics. As

Table 1
Classification of related work.

Supervised–Flow based	Supervised–Packet based
[2–5,9,10,12,16,21,22,26,28,30,31,35,36, 39–41,43,46,48,51,53–55,58,64,66]	[20,25,37]
Unsupervised–Flow based	**Unsupervised–Packet based**
[6,14,17,42,56,57,59,65]	This work

a result, when traffic patterns evolve or attackers modify their timing strategies or techniques, the thresholds must be manually updated, and the system may fail to detect new or modified attack variants.

In this section, we focus on machine learning-based (ML)-approaches due to their superior performance and their ability to adapt to evolving network conditions by learning from newly observed traffic. ML-based IDS solutions can be further categorized based on two main aspects: the learning paradigm—either *supervised* or *unsupervised*—and the traffic representation granularity, which can operate at the *packet* or *flow* level. Table 1 summarizes related ML-based works according to this classification.

2.1. Supervised flow-based approaches

Nugraha and Murthy [35] proposed a deep learning-based approach for SDN environments, using a hybrid CNN-LSTM model trained on labeled traffic flows. They evaluated their method on a simulated SDN topology with synthetically generated benign and slow DoS traffic, achieving near-perfect precision and recall.

The LSTM also serves as the core component of the IDS proposed in [3,16,36,66], where the LSTM is integrated with a CNN [3], or feed-forward neural network layers [36] to capture temporal dependencies and improve feature representation.

Shallow ML models have been adopted in [9,10,40,53,58]. De Miranda Rios et al. [10] aggregated packets within a 1-second time window. Their approach, tested on both real and emulated traffic, achieved an F1-score close to 100%. However, the use of fixed time windows may yield very short flows for low-traffic hosts, potentially leading to atypical features and misclassification—an issue the authors do not address. Tang et al. [58] proposed a detection and mitigation framework where a performance-based detection module determines whether slow DoS attacks take place, and if so, a feature-based detection module attempts to identify the attack origin based on time-frequency analysis. Perez-Diaz et al. [40] proposed a framework for detecting and mitigating slow DoS attacks in SDN environments. Their mitigation strategy involves blacklisting IPs flagged as malicious and halting communication if malicious behavior persists. Tested on the CICDoS2017 dataset [18], their system achieved 95% accuracy.[1] Chen et al. [9] recently introduced a decision tree-based IDS for high-speed networks. Their method achieves near-perfect F1 scores using features that are assumed to characterize the signature of specific slow DoS attacks, such as the TCP packet rate, the TCP packet length standard deviation, and the number of TCP packets sent without payload, to name a few. Finally, Sudar and Deepalakshmi [53] proposed a flow-based detection and mitigation framework using Support Vector Machine, C4.5 Decision tree and Naïve Bayes classifiers. In mitigation phase, they inhibit the malicious flow by creating blocking port rule immediately.

Multi-layer Perceptrons (MLP) are at the basis of the IDSs proposed in [2,4,31]. Asad et al. [5] evaluated their IDS with the CICIDS2017 dataset [49], reaching a 98% F1-score on benign traffic but only 55% on malicious samples. With the same dataset, Al-Shukaili et al. [2] and Muraleedharan and Janet [31] achieved a detection accuracy close to 100%. Alashhab et al. [4] proposed an ensemble-based approach for SDN environments where the MLP is supported by the BernoulliNB, Passive-Aggressive, and SGD classifiers.

Xu et al. [64] modeled traffic as time series of packet counts sampled every 10 ms. Time series is then processed through a 1D-CNN followed by a bidirectional GRU, which achieved an F1-score of 96.7% in a multi-label detection and classification task.

Scala et al.[46] proposed a hybrid deep learning-based IDS that combines unsupervised and supervised components to improve detection performance in scenarios with limited labeled data. The unsupervised module extracts task-independent features from network traffic, while the supervised component learns task-specific representations in a few shot setting.

2.2. Unsupervised flow-based approaches

Tang et al. [56] observed that slow DoS attacks exhibit high variance and mean deviation in TCP/UDP packet rates. Based on this, they proposed an unsupervised IDS using SADBSCAN clustering. Flows are represented by these statistical features and compared to clusters derived from attack-free data. At runtime, outlier flows are flagged as slow DoS. The method provided comparable performance to supervised approaches achieving a detection accuracy of 98%. Tang et al. proposed two similar approaches [57,59], differing from [56] in the attack signatures and clustering algorithms used.

Garcia et al. [14] used a Gaussian Mixture Model to cluster normal traffic flows and estimate the likelihood of new flows belonging to these groups. When uncertainty is high, an autoencoder is applied to uncover deeper correlations. Their approach achieves 98% detection accuracy.

Pratomo et al. [42] proposed an unsupervised detection method for various low-rate attacks using an autoencoder trained on byte histograms of application-layer messages. During inference, HTTP messages are classified based on reconstruction error. However, this approach may be less effective against slow DoS attack variants such as Slowloris, where HTTP messages are deliberately left incomplete.

Cambiaso et al. [6] proposed enriching the training dataset with additional discriminative attributes obtained through non-linear transformations of the original input features, with the aim of facilitating the learning of more expressive latent representations. They then employed a U-Net-like autoencoder to detect anomalous network flows.

2.3. Supervised packet-based approaches

Packet-based techniques are less common than flow-based approaches. Kemp et al. [20] used shallow ML models on packet-level data, incorporating some flow-level features, such as the time elapsed since the first frame and previous frames, computed at each network packet. Their approach, evaluated on real traffic, achieved a 99% AUROC with the C4.5N algorithm. Interestingly, their feature analysis revealed that flow-level attributes were among the most effective for attack detection.

Liang and Znati [25] developed an LSTM-based system using sequences of packets described by 15 packet-level and 3 flow-level features. Tested on CICIDS2017, their model achieved an F1-score close to 100%.

Similar to our approach, Paolini et al. [37] propose aggregating packets into images and using them to train a residual network. However, they rely on supervised learning and require more complex preprocessing by aggregating packets from the same flow.

Notably, none of the methods above can be considered fully packet-based, as they still rely on some flow-level features to enhance detection performance.

2.4. Discussion

Flow-based techniques aggregate network packets into flows defined by the same source and destination IP addresses, ports, and transport

[1] While our evaluation is not conducted within an SDN-based architecture, we cite SDN-related works due to their conceptual similarities. The differences in SDN environments pertain specifically to data collection and deployment, whereas the core detection methodology is directly applicable to both environments.

protocol. For each flow, a large set of aggregated features—typically between 50 and 80 in most public datasets [13,18,45,49]—must be computed. This design introduces a fundamental trade-off among accuracy, latency, and computational cost. High-granularity approaches, which rely on frequently sampled flows, can improve detection accuracy and enable detailed forensic analysis, but the resulting data volume significantly increases computational overhead. Conversely, coarser aggregation reduces resource consumption and improves scalability, but at the cost of higher detection latency, meaning that malicious activities may be identified too late for effective mitigation.

To cope with this issue, some works propose sampling flows using small, non-overlapping time windows (e.g., 1s [10]). However, time-based flow sampling (also known as periodic sampling) may prove ineffective in high-volume web servers, as the delay introduced by the sampling interval may cause the system to miss key indicators, such as the crucial connection count increase that characterizes slow DoS attacks, which may fall across two consecutive sampling windows. As a result, achieving high accuracy, low detection latency, and low computational cost simultaneously is intrinsically challenging. Despite this, the impact of flow sampling on detection performance remains largely overlooked in existing literature. In fact, several flow-based approaches have been evaluated using public datasets [13,18,49] where network flows correspond to *entire* network conversations. This suggests that their effectiveness has primarily been demonstrated in detecting attacks *after* their completion (*off-line* detection) rather than *during* their execution (*real-time* detection).

Moreover, some of the flow-based techniques discussed above are designed as general-purpose anomaly detectors targeting multiple attack categories. As a result, their reported performance often reflects accuracy metrics aggregated across different attack types, making it difficult to assess their effectiveness specifically for slow DoS detection [12,17,21,22,26,48,55].

Packet-based techniques [20,25,37], on the other hand, cannot be considered fully packet-based, as they also rely on flow-level information in addition to packet features. Moreover, while they do enable attack detection, none of them offers a method for identifying the attacker. Finally, their supervised nature limits their applicability due to the scarcity of labeled data.

These challenges prompted us to explore a detection framework that *(i)* is independent of labeled, *(ii)* learns network traffic behavior exclusively from packet-level features, entirely avoiding aggregation, *(iii)* enables real-time detection, and *(iv)* provides the critical ability to pinpoint the IP address of potential attackers for immediate and effective countermeasure deployment.

3. Background on slow DoS attacks

Denial of service attacks are launched to make a network service/component unreachable to its intended users. While flooding-based DoS attacks overwhelm the target's bandwidth [61], other approaches exploit specific vulnerabilities where even just a single network packet can lead to a DoS on the victim host [32]. Slow DoS attacks represent an intermediate approach, as they target the application daemon by sending a limited number of packets [7]. While some slow DoS attacks exploit application level timeouts, others leverage TCP-related fields [50]. In this work, we consider both methods by analyzing the behavior of **Slowloris** and **Slow Read** attacks when targeting network services, also including **Slow HTTP POST** as a variant of the Slowloris attack.

Slowloris creates a large number of connections to the victim's listening daemon. After a connection is established, the attacker sends an HTTP request to the victim at an extremely slow rate to keep these connections open indefinitely, maintaining a transmission rate just high enough to prevent the server from closing them. Each HTTP request is cut into equally-sized HTTP packets, resulting in a constant, small size of Layer 7 traffic. As a result, all available concurrent connections are occupied by the attacker's requests, exhausting the server's capacity. In this state, legitimate client connections may still be established at the transport layer, but the application daemon does not process them, effectively causing a denial of service [7].

Slow Read exploits the server's response mechanism to slow down data transmission. They achieve this by advertising a small TCP window size to the server. The TCP window size field indicates how much data (in bytes) the sender is allowed to transmit before it must pause and wait for an acknowledgment. This mechanism is designed to prevent buffer overflows by informing the sender of the available space in the receiver's buffer, ensuring the receiver isn't overwhelmed. In a Slow Read attack, the attacker sends a legitimate HTTP request but sets a small TCP window size, causing the server to transmit data very slowly, one small chunk at a time. Then, the attacker reads the server's response very slowly or not at all, causing the server to pause frequently, and keeping the connection open as long as possible [7].

The two attacks rely on generating a large number of TCP connections with the victim. Establishing a TCP connection follows the *three-way handshake* process, which involves an exchange of three TCP packets: the client initiates the connection by sending a SYN packet to the server, the server responds with a SYN-ACK packet to acknowledge the request, and finally the client confirms the connection by sending an ACK packet.

Slow HTTP POST [7] is a Slowloris variant. The difference between the two attacks is that Slowloris delays the transmission of the header of the HTTP request, while Slow HTTP POST slows down the transmission of the body.

To ensure a comprehensive overview, we also provide descriptions for **Slow Next** and **Sockstress**, which are variants of the Slowloris and Slow Read attacks, respectively. However, they are excluded from our experimental analysis because the publicly available datasets utilized in this study do not contain instances of these specific attacks. Slow Next [8] is a Slowloris variant, whereby the attacker dispatches multiple HTTP requests for each TCP connection at precise time intervals, compelling the server to maintain all connections in an open state. The Sockstress attack [27] operates similarly to the Slow Read attack, exploiting the server's response mechanism to slow data transmission. It achieves this by advertising a zero-window (or a very small-byte window), signaling that it cannot receive additional data. The server continuously probes the client (sends window probes) and holds memory, connection table slots, and timers while waiting for the window to open. These probes persist indefinitely, as the attacker never clears the window. Each stalled connection occupies the victim' computational resources.

Other types of slow DoS attacks exist, such as **THC-SSL-DOS** (designed by the group "The Hacker Choice" in 2011[2]) and **Shrew** [23], but they fall outside the scope of this work as the former targets TSL/SSL rather than HTTP applications, while the latter focuses more on disrupting network flow control than application-layer connection exhaustion.

4. Proposed approach

Our intrusion detection system consists of four key modules, which are the **Pre-Processing (PP)** module, the **Anomaly Detection (AD)** module, the **Filter**, and the **Intrusion Detection (ID)** module. The role of the PP module is to convert raw network traffic into a format that the AD module can use, in particular it generates a 2-dimensional vectorial representation (or *image*) of network packets. The AD module consists of a U-Net Convolutional Autoencoder (UCAE) model trained with legitimate traffic and then used to classify the monitored traffic either as normal or anomalous. The classification task is threshold-based, i.e., the UCAE model reconstructs the images received from the PP module and marks them as normal if the reconstruction error remains below

[2] https://www.thc.org/.

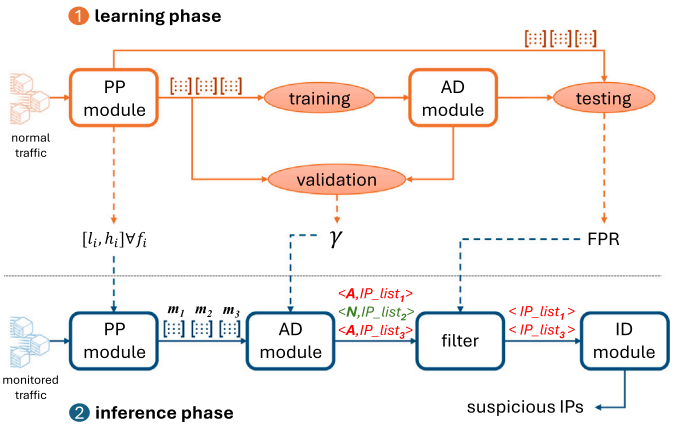

Fig. 1. The proposed intrusion detection framework.

the anomaly threshold γ, otherwise it marks them as anomalous. The Filter enhances classification accuracy by reclassifying some anomalous samples as normal if it deems them to be potential false positives. The name "Filter" refers to the fact that not all anomalies are forwarded to the next module, but only those considered most likely. Finally, the ID module analyzes anomalous samples to determine if there is evidence of an intrusion and, if so, identifies the attacker's IP.

Fig. 1 illustrates the two phases in the lifecycle of the four IDS components: an offline **learning phase**, which focuses on tuning the detection framework, and an online **inference phase**, which enables real-time detection.

In the learning phase, the UCAE model is first trained with legitimate traffic; then it is validated using the same training data, and the reconstruction errors from this step help in setting the anomaly threshold γ for the inference phase; finally, the UCAE model is tested with additional legitimate traffic to measure the false positive rate (FPR) under normal, attack-free conditions. Notice that normal traffic is not predefined using explicit rules, thresholds, or manually engineered characteristics. Instead, normal traffic is defined operationally as the traffic observed in the monitored network environment during the learning phase.

During the inference phase, the UCAE model classifies monitored traffic based on the anomaly threshold γ. The Filter receives the classified samples as $\langle class, IP\_list \rangle$ pairs, where $class \in \{normal, anomalous\}$ and IP_list contains the contributing IP addresses for the image. The Filter only forwards the anomalous samples to the next stage if the anomaly rate surpasses the FPR observed during the system's testing stage. The Filter works under the assumption that the UCAE model can make mistakes and thus expects the anomaly rate not to be zero during attack-free periods, but still significantly lower than when attacks are in progress. Therefore, it considers anomalies as false positives as long as the anomaly rate remains under the expected FPR. The ID module analyzes the sequence of IP_list forwarded by the Filter to check whether there is at least one IP address responsible for causing the last k anomalies. If such an IP address exists, it is flagged as a potential attacker, enabling the implementation of appropriate countermeasures (e.g., blocking the IP address at the firewall). If no IP address consistently appears across these k anomalies, the events are treated as false positives and discarded.

In both the learning and inference phases, the PP module converts raw network packets into images and forwards them to the AD module. While the AD module processes the current image, the PP module prepares the next one. This design ensures that the IDS never stores more than one image in memory for each monitored host at any given moment.

The next sections describe each module in detail.

4.1. Pre-processing module

The role of the pre-processing module is to generate a vectorial representation of network packets. **Feature extraction**, **image construction**, and **normalization** are the three sequential steps of this procedure, which yield a 2-dimensional vector for each network packet.

4.1.1. Feature extraction

Feature extraction involves selecting the most relevant packet features for intrusion detection. We use 11 packet features, namely: the size in bytes of Layer 7 data `size_L7`; the fields `window`, `FIN`, `SYN`, `RST`, `ACK`, `ECE`, `URG` and `CWR` of the TCP packet header; the transport protocol used to send the packet `protocol_L4`; and the destination port `dst_port`.

`size_L7` is useful for capturing the behavior of slow DoS attacks that send HTTP requests at an extremely slow rate, resulting in a constant, small size of Layer 7 traffic. Layer 7 refers to the Application Layer of the ISO/OSI model [47], where protocols such as HTTP operate and where slow DoS attacks typically manifest. TCP header fields are expected to capture the initial phase of a slow DoS attack, during which an unusually high number of TCP connections are established. If a packet uses a transport protocol other than TCP, these fields are set to 0. Features `protocol_L4` and `dst_port` help the IDS differentiate potentially harmful traffic from clearly legitimate activity. Indeed, malicious traffic typically consists of TCP packets targeting the listening ports of the vulnerable services. Therefore, packets that do not exhibit these characteristics can be confidently regarded as benign without requiring further disambiguation. Further details on this aspect are provided in the next section.

4.1.2. Image construction

In the second step, multiple feature vectors are stacked horizontally to form a matrix, where each vector represents a column in the matrix (see Fig. 2). Formally, a network packet p_i is represented as a matrix m_i of dimensions $(11, w)$, where:

- columns correspond to feature vectors from packets $p_{i-(w-1)}, p_{i-(w-2)} \cdots p_i$;
- all packets in the matrix are directed to the same IP address;
- w (for *width*) represents the number of packets observed before and including p_i.

This representation enables the encoding of the recent history of traffic directed to a single IP address, regardless of the source IP, in the form of 11 w-length time series, one for each packet feature.

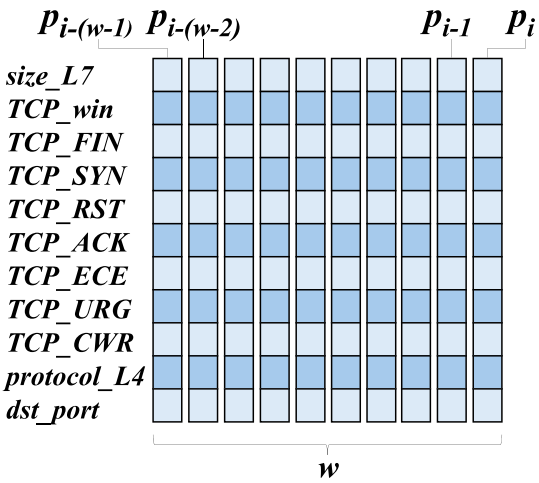

Fig. 2. The vectorial representation of the i^{th} network packet.

By aggregating packets into images, the model can capture short-term traffic context and structural relationships among consecutive packets. This representation provides a richer input to the learning model while still preserving the packet-level nature of the analysis. In fact, packet features are not aggregated as in flow-based approaches; instead, they are simply arranged into the cells of a matrix, allowing the model to analyze sequences of packets without losing their individual characteristics.

4.1.3. Normalization

The normalization task involves converting feature values to a numerical format that the UCAE model can process. We propose a normalization approach specifically designed to facilitate anomaly detection, which accounts for continuous, binary, and categorical features.

Slow DoS attacks specifically rely on the TCP protocol and target specific ports (i.e., those associated with targeted services). To capture this behavior, categorical features `protocol_L4` and `dst_port` are converted to binary features so that values that clearly indicate legitimate traffic (i.e., not rooted in TCP, nor destined for vulnerable-services ports) are encoded as 0, whereas those with a higher risk of being malicious are encoded as 1. We refer to this preprocessing approach as **risk-aware binarization**. Specifically, feature `protocol_L4` is set to 1 if the packet uses TCP, otherwise it is 0. Similarly, feature `dst_port` is set to 1 if the destination port corresponds to a service vulnerable to slow DoS attacks, and 0 otherwise. This encoding ensures that traffic with a higher risk of being malicious is clearly distinguished from unambiguously legitimate traffic. Additionally, a low variation in pixel values can make it easier for the UCAE model to learn meaningful patterns, but also shrinks the feature space, keeping the model's complexity low. In Section 5, we show that this trick is fundamental for achieving high detection performance.

For what concerns continuous features, in the learning phase they are normalized to the range $[0.25, 0.75]$, while in the inference phase the same features are normalized to $[0, 1]$. The rationale beyond this idea is to have the model learn normal patterns within $[0.25, 0.75]$, making it more likely to classify values outside this range as anomalies. The method is implemented as follows. For training, a feature f is scaled so that its lowest value l (computed over the whole training set) maps to 0.25, and its highest value h maps to 0.75. During inference, the same bounds l and h are used to normalize the monitored traffic, meaning any value of f outside this range will be mapped outside $[0.25, 0.75]$. If the normalized value exceeds 1 or drops below 0, it is clipped to 1 or 0, respectively. We refer to this normalization approach as **anomaly-aware scaling**.

The interval $[0.25, 0.75]$ offers a balanced compromise: it is sufficiently narrow to make out-of-range values stand out during inference, yet wide enough to preserve the natural variability of legitimate traffic during training. Additionally, choosing symmetric margins—leaving a 0.25-wide band both below and above the normality interval—ensures that the ranges representing "normal" and "anomalous" values have equal width, preventing the model from implicitly favoring one over the other. Empirically, we found that both tighter and wider ranges degrade detection performance, as the former increases false positives, while the latter increases false negatives.

In our framework, the continuous features are `size_L7` and `window`. As discussed in Section 3, Slow Read and Sockstress attackers may deliberately set a very small `window` value to throttle the victim's data transfer. Similarly, during Slowloris and Slow HTTP POST attacks, the Layer 7 data size (`size_L7`) may exhibit anomalous values. When such deviations occur, the proposed anomaly-aware scaling effectively amplifies the distinction between legitimate and attack traffic, enhancing detection.

Binary features `FIN`, `SYN`, `RST`, `ACK`, `ECE`, `URG` and `CWR` are mapped to $[0.25, 0.75]$ as well, such that 0 maps to 0.25, and 1 maps to 0.75.

We notice that within this setting there is a risk that a "legitimate-but-abnormal" image may be mistakenly classified as anomalous. However, we have empirically verified that this can only happen if legitimate-but-abnormal packets account for a significant portion of the image. Since images are generated from packets originating from different sources, misclassification would only occur if the sources exhibiting unusual behavior also contribute most of the monitored traffic. Furthermore, even if the AD module misclassifies some images, they will not trigger any alert unless the Filter detects an anomaly rate exceeding the critical threshold *and* the ID module finds a suspicious IP. All these conditions must occur simultaneously for misclassified images to result in false alarms. On the other hand, malicious traffic tends to exhibit abnormalities in binary features as well, as the result of the high number of connections established with the victim. In Section 5, we demonstrate the robustness of our anomaly-aware scaling approach against legitimate-but-abnormal samples, showing that false alarms arise only when such packets constitute a very large portion of the monitored traffic.

4.2. Anomaly detection module

The Anomaly Detection (AD) module is implemented as a neural network. We begin by describing its architecture, followed by the classification mechanism used to separate anomalous from normal traffic.

4.2.1. Neural architecture

The neural network we employ for unsupervised image classification is a U-Net Convolutional Autoencoder (UCAE), whose architecture is illustrated in Fig. 3. This architecture was empirically chosen as the best-performing among several candidate models, with the Vision Transformer intentionally excluded. Vision Transformers have been proven to be very effective in image classification, and have already been employed for intrusion detection purposes [11]. However, due to their complexity and large number of parameters, their practicality is limited by long training times and high computational requirements. For these reasons we opted for a more lightweight architecture.

An Autoencoder (AE) consists of two subnetworks: an encoder Enc, responsible for learning a function $\mathbf{z} = \mathrm{Enc}(\mathbf{x})$ that maps the input $\mathbf{x}$ to a latent representation $\mathbf{z}$, and a decoder Dec, which learns to reconstruct the input from $\mathbf{z}$, i.e., $\mathbf{y} = \mathrm{Dec}(\mathbf{z})$.

Unlike a traditional AE, a U-Net-like AE [44] uses *skip connections* to facilitate multi-scale feature fusion. These allow the network to integrate both low- and high-level features from different stages of the encoder, enhancing its ability to capture fine details and broader contextual information—ultimately improving performance. Skip connections help to mitigate the *degradation problem*, a phenomenon where adding more layers to a deep neural network leads to worse performance, and to maintain a balance between convergence speed and model expressivity/accuracy. The effectiveness of U-net networks is often attributed to their ability to be interpreted as ensembles of multiple transformation paths, where the final network output is derived by combining pathways of varying lengths, leading to a built-in ensemble effect.

Our UCAE model utilizes hierarchical feature extraction through downsampling and upsampling operations. This model consists of two types of layers: *convolutional layers*, which apply filters to input data to extract hierarchical features, and *pooling layers*, which downsample these features to reduce spatial dimensions and enhance computational efficiency by retaining only the most significant information. The convolutional layers focus on feature extraction, while the pooling layers help in reducing data complexity and improving model robustness against minor transformations.

Max pooling is used asymmetrically to downsample primarily along the width dimension while preserving height-related spatial features.

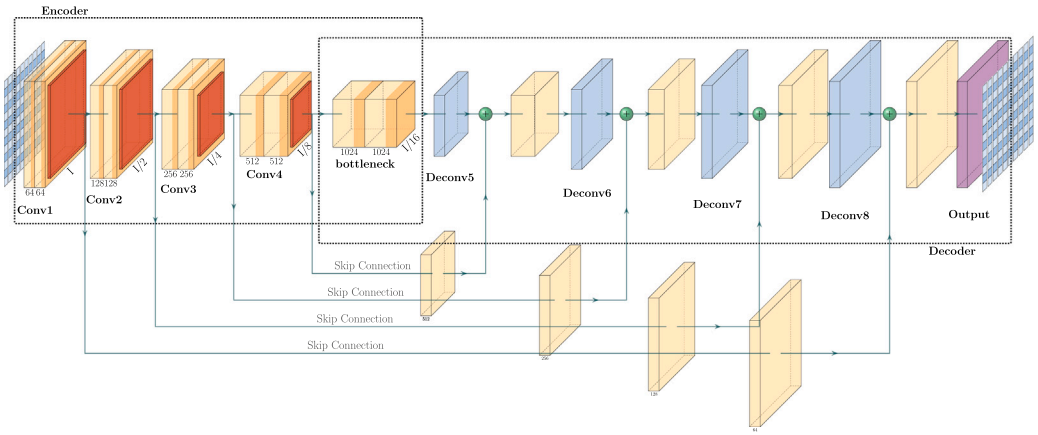

Fig. 3. The U-Net convolutional autoencoder architecture.

This selective downsampling helps retain essential vertical structures while progressively reducing width, making it well-suited for tasks involving sequential data. By reducing dimensionality in this way, the neural network efficiently extracts multi-scale features while maintaining important contextual information.

The input image undergoes a progressive reduction in spatial dimensions while its feature volume is doubled through a series of convolutional blocks, culminating in a compact core representation (*bottleneck*). During decoding, a sequence of deconvolutional blocks is applied, each merging with the respective residual block from the encoding phase. An additional convolutional block further processes the output at each stage. The final reconstruction is achieved using a sigmoid activation layer. Each processing block consists of a convolution layer followed by a rectified linear unit (ReLU) activation function.

A gradient descent procedure tunes the model weights by minimizing the loss function, which is defined as follows:

$$Loss(\mathbf{x}, \mathbf{y}) = \max_i \left(\frac{1}{M} \sum_{j=1}^{M} E(x_{ij}, y_{ij}) \right)$$

where

$$E(x_{ij}, y_{ij}) = \begin{cases} |x_{ij} - y_{ij}|, & \text{if } x_{ij} \in [0.25, 0.75] \\ 1, & \text{otherwise} \end{cases}$$

In the equations above

- $\mathbf{x} \in \mathbb{R}^{N \times M}$ and $\mathbf{y} \in \mathbb{R}^{N \times M}$ represent the original and the reconstructed images, with N and M being the number of rows and columns, respectively;
- x_{ij} and y_{ij} are pixels of $\mathbf{x}$ and $\mathbf{y}$, respectively, with i and j indexing rows and columns;
- $E(x_{ij}, y_{ij})$ represents the element-wise error, which equals the absolute difference between x_{ij} and y_{ij}, if x_{ij} is in the range of "normal" values $[0.25, 0.75]$ (see Section 4.1), otherwise it is set to 1.

This loss function, which we refer to as **threshold-driven reconstruction loss**, computes the maximum per-row average error across all rows. By doing so, it encourages the UCAE model to reconstruct images on a row-wise basis while ensuring values remain within the range $[0.25, 0.75]$. This approach serves two key objectives: *(i)* since rows represent time-series data of packet features, if a sequence of (not necessarily adjacent) columns corresponds to malicious packets, we expect at least one row in the reconstructed image $\mathbf{y}$ to deviate enough from the corresponding row in the original image $\mathbf{x}$ to push the error above the anomaly threshold γ; *(ii)* greater importance is given to packets exhibiting abnormal feature values, i.e., outside the range $[0.25, 0.75]$, by assigning the maximum penalty of 1 instead of the absolute difference $|x_{ij} - y_{ij}|$, thereby amplifying the impact of features that strongly deviate from expected values.

4.2.2. Anomaly detection mechanism

The detection process is based on the reconstruction error, which quantifies the discrepancy between an observed image $\mathbf{x}$ and its reconstructed counterpart $\mathbf{y}$. The reconstruction error is computed as the Mean Absolute Error (MAE) between $\mathbf{x}$ and $\mathbf{y}$. If this error exceeds the anomaly threshold γ, the system classifies $\mathbf{y}$ as anomalous. The choice of MAE over the Mean Squared Error (MSE), which is widely adopted for learning standard AE, is motivated by its robustness and stability in reconstruction-based anomaly detection settings. Since in our framework the autoencoder is trained exclusively on benign traffic, the reconstruction error directly defines the anomaly score used for threshold-based classification [33]. In this context, MSE may excessively amplify large reconstruction deviations due to its quadratic penalty, causing the training process to be disproportionately influenced by rare but legitimate variations of normal traffic. This effect may distort the reconstruction-error distribution and complicate the definition of a stable anomaly threshold. Differently, MAE applies a linear penalty to reconstruction deviations, preserving proportional differences between normal and abnormal samples without overemphasizing extreme values. This results in a more stable and interpretable reconstruction-error distribution, which is particularly important when the anomaly decision relies on percentile-based thresholding. Moreover, network traffic data are typically characterized by non-Gaussian distributions and occasional benign outliers; under these conditions, the L_1 loss provides greater robustness than the L_2 loss, leading to more reliable anomaly discrimination.

γ is determined in the learning phase by computing the reconstruction errors on the validation set, sorting these errors in ascending order, and selecting a specific percentile $\mathbf{P_q}$, with $0 < \mathbf{q} \leq 100$, as the normality threshold. The chosen percentile determines the sensitivity of the detector, with lower percentiles reducing false negatives and higher percentiles reducing false positives. Typically, $\mathbf{P_q}$ is set to a high percentile of the observed reconstruction errors so that the detector models the upper bound of normal behavior and flags as anomalous those inputs whose reconstruction error significantly deviates from the distribution observed on benign traffic. Nevertheless, in the experimental section we evaluate multiple percentiles, and demonstrate that our framework is robust with respect to γ, as any $\mathbf{P_q}$ with $50 \leq q \leq 95$ enables the detection of all attacks across three different network environments.

Other architectural hyperparameters, such as the number of layers, the latent-space dimension, the batch size, etc., can also be selected according to the target network environment by minimizing the reconstruction error on validation data while ensuring that the model does not overfit the training samples.

4.3. The filter module

A well-tuned IDS aims to minimize false positives while maintaining high detection accuracy without unnecessary noise. Indeed, false positives can be problematic in environments where automated responses are in place, as they can lead to unintended actions, such as shutting down critical services or denying legitimate users access. While neural networks demonstrate strong performance, they can still produce incorrect outcomes, especially in unsupervised settings. Moreover, since attacks are rare, IDSs operate in normal conditions most of the time, which increases the likelihood of false alarms.

Based on this assumption, the Filter's role is to correct potential errors made by the AD module. In particular, it considers anomalous samples as false positives as long as the rate at which the AD module outputs anomalies remains below a threshold Tr. Basically, it expects the anomaly rate not to be zero during attack-free periods, but still significantly lower than when attacks are in progress. The threshold Tr is computed as the maximum false positive rate (FPR) observed when testing the UCAE model with attack-free traffic (different from the traffic used for training). The FPR is not evaluated over a time period (indeed our IDS is *time-agnostic*) but rather over a sequence of n packets. Formally, for each packet p_i, FPR_i is computed as the number of anomalies detected between packets p_{n-i+1} and p_i, divided by n. Finally, the threshold is obtained as $Tr = \max_i(FPR_i)$. Fig. 4 illustrates a scenario where the UCAE model detects anomalies (circles), but only those that push the anomaly rate above Tr are considered *potentially harmful*, while the rest are filtered out. Among these, some are true positives (green circles), while others are false positives (red circles), i.e., mistakenly flagged as potentially harmful due to the high anomaly rate, despite being benign. The number of false positives that remain after filtering is substantially lower than the misclassifications produced by the UCAE model alone (red + yellow circles). This improvement is clearly demonstrated by the experimental results presented in Section 5.2.

Notice that evading detection by staying below the threshold Tr (Fig. 4) is impractical, as it requires knowing the threshold—information unavailable to the attacker without replicating the detection system. .

The window size n does not impact detection performance, however it must be chosen large enough to keep Tr below 1—or any other value x deemed appropriate by the user—to ensure the system can detect ongoing attacks. In fact, if n is too small, Tr may become so high that an ongoing attack would fail to raise the anomaly rate above it, causing it to go undetected. A suitable value for n can be determined through trial and error. For given values of x and n, if the calculated threshold Tr is greater than x, the user has two choices: s/he may either increase n or retrain the UCAE model by including the current test traffic in the training set and then recompute Tr using a different test set.

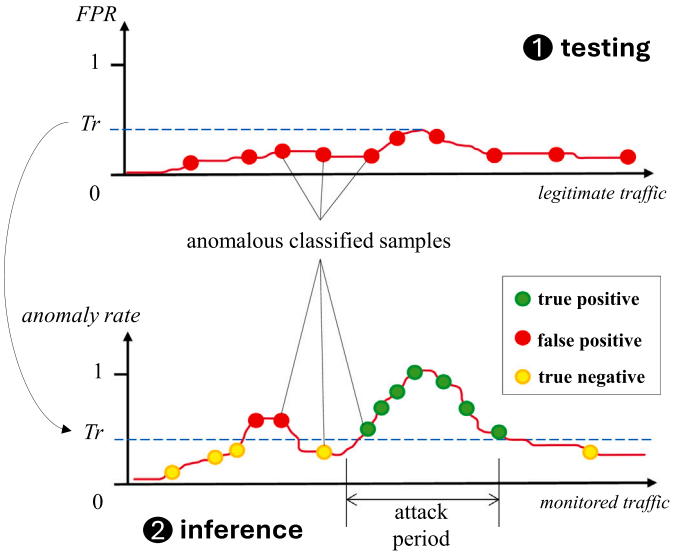

Fig. 4. Filter: threshold Tr is evaluated when testing the model with benign traffic (top), and then it is used during inference as a cutoff to reduce the number of false positives (bottom).

4.4. Intrusion detection module

The ID module is the final stage where samples originating from the PP module end their run. Here, they are examined to assess whether they are symptoms of a slow DoS attack or simply legitimate-but-abnormal traffic. In the first case, the attacker's IP address is identified so that appropriate countermeasures can be taken against it; otherwise, anomalies are considered as false positives and ignored. In flow-based approaches, identifying the source of an attack is straightforward because each flow is associated with a single source IP; thus, once a flow is flagged as anomalous, the attacker can be immediately identified. In contrast, our method aggregates packets without regard to their originating source, which means that attacker identification requires additional processing.

The approach works as follows: given an integer k, if at least one IP address is identified as being responsible for the most recent k anomalies, then it is flagged as a potential source of the attack and blocked from further data exchange. Conversely, if no such IP address is found, the anomalies are disregarded. We refer to the parameter k as the **Suspicious Activity Window** (SAW). As described in Section 4.1, each image represents a sequence of packets sharing the same destination IP address but originating from different sources, whose IP addresses are collected in *IP_list*. The detection logic is built on the premise that, during normal, attack-free operation, no single IP address should consistently contribute to all k consecutive anomalies; specifically, an IP address is not expected to be present in every *IP_list* across the last k anomalous samples. On the contrary, an actual attack is characterized by the attacker's IP address appearing persistently across all these *IP_list* records. By enforcing this strict consistency requirement, the ID module effectively filters out false alarms resulting from transient traffic fluctuations. Consequently, an alert is triggered only upon observing this unique, persistent pattern, ensuring that other isolated anomalies are correctly dismissed as false positives.

4.4.1. Estimation of the SAW

The size of the SAW directly affects detection performance. A small window allows for fast detection, as the ID module makes decisions after the first k anomalies. Therefore, the smaller the value of k, the quicker an attack can be identified, if any. However, a small k also increases the risk of false alarms, since it becomes more likely that legitimate sources contribute to anomalous images within a small SAW, regardless of whether the anomalies stem from malicious activity or simply benign but unusual behavior. Conversely, increasing the window size decreases the probability that a single legitimate IP address is responsible for *all* anomalies within the SAW, reducing false positives.

Unfortunately, slow DoS attacks can vary widely in traffic volume, ranging from as few as a thousand packets to several hundred thousand. This variability makes it impractical to rely on traffic volume alone when configuring the SAW size. Instead, the SAW must be estimated based on the characteristics of the monitored service along with common behavioral patterns shared across different types of slow DoS attacks, irrespective of their specific volumes.

An effective approach is to calibrate the SAW so that slow DoS attacks are detected before they can occupy a significant portion of the victim's available connections. To this end, we first estimate the average number of concurrent connections C_{avg} that the service maintains with clients during a representative observation period—typically around 10min, which is the minimum duration for many slow DoS attacks to be effective. Based on this, we compute the number of free connections C_{free} that on average are available to potential attackers as $C_{free} = C_{max} - C_{avg}$,

where C_{max} is the server's maximum supported number of concurrent connections. During the same time window, we also measure the victim's average incoming traffic volume, denoted V_v. Finally, we estimate the minimum traffic volume V_{DoS} required for a slow DoS attack to successfully deplete the remaining C_{free} connections. This estimation considers three typical behaviors of the slow DoS attacks: *(i)* opening multiple connections, *(ii)* manipulating TCP flow control mechanisms to throttle data transmission, and *(iii)* sending just enough packets to prevent the server from closing idle connections.

To initiate a TCP connection, the attacker begins with the standard three-way handshake, which requires only two packets from the attacker's side. The subsequent manipulation of the TCP flow control mechanism depends on the type of slow DoS attack. In the case of Slowloris, the attacker sends the initial part of an HTTP request, then continues to drip incomplete header or body lines at carefully timed intervals to keep the connection alive. For Slow Read, instead, the attacker sends a single HTTP GET request and then acknowledges each segment of the server's response with a separate ACK packet.

Assuming a worst-case scenario where the attacker sends a single packet immediately before the web server's connection timeout T expires, both Slow Read and Slowloris attacks would require approximately $\sim 600/T$ packets per connection. This translates to one packet being sent every T seconds over a total period of 600 s (i.e., 10min). If we denote the total number of packets per connection as P, the minimum traffic volume V_{DoS} required for a slow DoS attack to exhaust the victim's resources can be estimated as $V_{DoS} = C_{free} \cdot P$. Since T varies significantly depending on the specific web server (e.g., from 5 s for the Apache HTTP server to 120 s for Microsoft IIS), V_{DoS} fluctuates correspondingly.

Given the victim's average incoming traffic volume V_v, the total traffic volume observed during the attack would be $V = V_v + V_{DoS}$. The value of V serves as a reference point for selecting the size of the Suspicious Activity Window k. However, since V is a rough estimate derived from a worst-case scenario, it should be considered a guideline rather than an exact threshold. In practice, choosing k between $V/2$ and V is generally sufficient to ensure reliable and real-time detection performance.

This approach ensures that the SAW window is sufficiently sized to capture even low-volume slow DoS attacks while avoiding unnecessary delays in decision making. Nevertheless, absolute real-time guarantees cannot be formally established due to the inherent variability of network behavior. However, when the SAW size is estimated as described above, our experimental results show that most attacks are detected before reaching 50% of their progression, thereby confirming the practical real-time effectiveness of the proposed approach.

4.5. Comparison against flow-based analysis

Beyond its unsupervised nature and inherent real-time capabilities, the proposed approach requires a significantly lower computational footprint compared to traditional flow-based methods (excluding the resources necessary for the UCAE model itself). Flow-based approaches necessitate the IDS to maintain a large repository of active flow instances for every IP address communicating with the monitored host. Furthermore, as flows persist, numerous features must be continuously updated for all active flows, increasing overhead. Our approach, by contrast, minimizes both computation and memory requirements by employing a streamlined pipeline: the AD module processes m_i and forwards the classification result along with the associated *IP_list*; the Filter forwards only a fraction of the classified samples; the ID module processes a maximum of k text-based *IP_list* records, which require a negligible amount of memory; in the meanwhile, the PP module extracts 11 features from packet p_{i+1}, updates m_i by column shift, i.e., by removing the first column and appending the new feature vector as the last column to create m_{i+1}. Since each image is made from packets destined to the same IP destination address, this pipeline ensures that the detection system never stores more than a single image for each monitored host at any given moment. Indeed, each image m_i remains in memory only for the time interval between packets p_i and p_{i+1}. This characteristic enables our framework to be deployed on devices with limited memory resources. Fig. 5 illustrates the difference between our approach and the flow-based analysis.

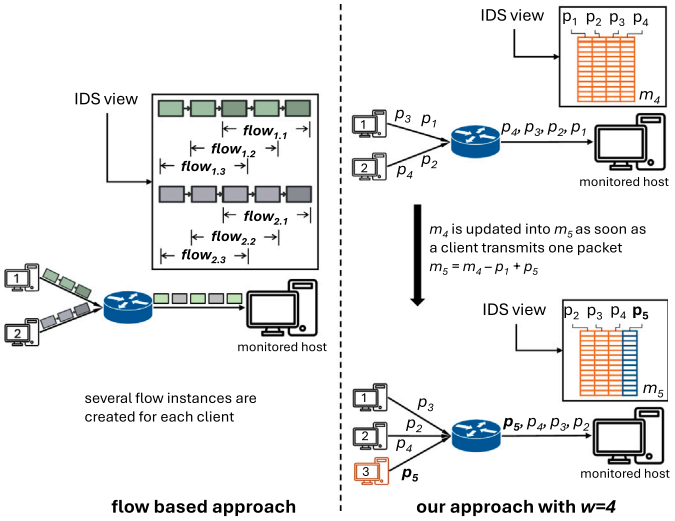

Fig. 5. Flow-based approach *vs* our approach.

4.6. Limitations

A limitation of the proposed intrusion detection approach is its inability to detect distributed slow DoS attacks. In these scenarios, multiple malicious IPs act in coordination, each mimicking the benign behavior of legitimate sources, which complicates detection. As a result, the ID module would fail to identify a single IP address responsible for the most recent k anomalies.

One possible mitigation is to flag all IP addresses associated with anomalous images as potentially malicious. While this may increase the false positive rate by blocking some benign sources, it can still halt the attack and maintain service availability, preventing a full denial of service. We note, however, that handling distributed slow DoS attacks falls outside the scope of this work and is left for future research.

A further limitation stems from the fact that anomaly-based detection systems are trained to model only the distribution of benign traffic, making them inherently sensitive to benign but out-of-distribution (OOD) samples at inference time. Because the model's understanding of normality is restricted to the patterns present in the training data, any legitimate behavior that falls outside this learned boundary is likely to be flagged as anomalous, even when it poses no security threat.

This issue arises because the space of benign behavior is broad, diverse, constantly evolving, and shaped by changes in user activity (e.g., flash events), software updates, network configurations, and environmental conditions. Since the training set can capture only a limited portion of this variability, the detector inevitably encounters novel yet harmless samples that do not match its learned representation. Lacking any prior knowledge of these unseen benign patterns, the model interprets them as deviations which may raise alarms. In essence, anomaly-based ML systems are inherently constrained by the fact that normal behavior is unbounded, dynamic, and only partially observable, while their learned representation of normality is necessarily finite and static. This structural mismatch ensures that benign OOD samples will always be a challenge.

Two potential, though not fully definitive, ways to mitigate this issue are: using a highly heterogeneous training set, since diverse benign samples broaden the model's understanding of what constitutes legitimate

behavior; or alternatively, continuously retraining the model, enabling it to adapt as the monitored environment evolves.

It is important to note, however, that this limitation is not specific to our solution; rather, it is an inherent characteristic of all anomaly-based detection approaches. Additionally, addressing this limitation is beyond the scope of this work.

A more critical scenario arises when malicious traffic is included in the training set and therefore learned as normal behavior. In this case the false negative rate could increase since the detector may partially absorb attack characteristics into its learned notion of normality. Such contamination may occur in two main situations: *(i)* the security operator unintentionally selects training traffic that already contains attacks, or *(ii)* an attacker poisons the training data to compromise the detector. These issues are not specific to our approach, rather, they represent a well-known limitation of machine learning based classifiers in general. At the same time, existing literature shows that anomaly-based detectors are typically able to tolerate poisoned or mislabeled traffic without significant degradation in performance when the contamination rate remains below 15% of the overall volume of benign training data [38]. Nonetheless, there exist several approaches to robust anomaly detection and poisoning-resistant training strategies, including data sanitization [52], robust statistical methods [34], trusted bootstrapping procedures [24], and adversarially robust learning techniques [15].

5. Experiments

In this section, we demonstrate the effectiveness of the proposed approach through a series of experiments. We focus on the Slowloris and the Slow Read attacks, along with the Slow HTTP POST variant. These attacks represent today the most known and common slow DoS. We do not consider other attack variants as they are not included in publicly available datasets. However, our detection method relies solely on identifying deviations from legitimate network behavior, not on specific attack patterns. Therefore, demonstrating its effectiveness against these three prominent attacks in diverse network environments provides a clear indication of its potential.

In Section 5.1 we describe the datasets used for evaluation; in Section 5.2 we show the performance of our intrusion detection system; in Section 5.3 we compare our approach with other methods; in Section 5.4 we provide an ablation study to highlight the contribution of all our architectural choices; in Section 5.5 we demonstrate the robustness of the anomaly-aware scaling preprocessing technique against legitimate-but-abnormal traffic; finally, in Section 5.6 we analyze the results of our evaluation.

5.1. Datasets

For our experiments we selected CSE-CIC-IDS2018 [49] and 5G-NIDD [45], which, to the best of our knowledge, are the only two publicly available datasets containing slow DoS attacks with ground truth available at the packet level. However, since they are relatively small and may not fully capture the diversity of real-world traffic, we also built our own dataset using traffic collected from the Institute of Electronics, Computer and Telecommunication Engineering of the National Research Council (CNR-IEIIT), located in Genoa, Italy. This additional dataset enhances the robustness of our evaluation by providing a broader and more representative sample. Using three distinct and independent datasets ensures that the results are robust and unbiased, as there is no overlap or relationship between traffic samples.

CSE-CIC-IDS2018 [49] (CICIDS18 for short), is a benchmark dataset developed by the Canadian Institute for Cybersecurity to analyze, test, and evaluate intrusion detection systems. It is built on a common LAN network topology on the AWS computing platform.

5G-NIDD [45] is a dataset built on a functional 5G test network at the University of Oulu, Finland. The network traffic is collected from two base stations, each connected to the attacker node, several benign 5G users, and the victim node.

Both datasets contain various attacks, however, for this work we only focus on the slow DoS attacks Slowloris and Slow HTTP POST, as they are the only ones relevant to our study.

Our dataset, referred to as OWN_DATA, comprises authentic (benign) network traffic captured from 34 hosts of the CNR-IEIIT research institute over an entire week, supplemented by manually injected slow DoS attack traffic. It encompasses a wide range of routine activities, including web browsing, email, video conferencing, instant messaging, cloud service interactions, and internal data transfers. The malicious traffic is generated by executing four different slow DoS attacks against an Apache2 web server: two Slowloris (differing in packet rate), one Slow Read, and one Slow HTTP POST. The test set is derived from the network traffic of a randomly selected weekday (excluding weekends) for each host, with slow DoS attack traffic injected every two hours, totaling 272 attacks (an average of 8 attacks per host). The traffic from the remaining six days is used as the training set. By reflecting the realistic network conditions and user behaviors typical of an enterprise environment, the dataset provides a representative model of how such attacks manifest in real-world operational networks.

The statistics for the three datasets are reported in Table 2, where "Monitored Hosts" refers to the number of hosts under IDS monitoring, while "Peer Hosts" denotes the number of external hosts that the monitored hosts communicate with. The latter is particularly relevant because, as the number of interacting hosts and the traffic volume increase, it becomes easier for an attacker to blend into benign traffic. The statistics for the slow DoS attacks are reported in Table 3. The heterogeneity of attacks in terms of traffic rate and duration allows us to cover a wide range of scenarios, ensuring a comprehensive evaluation of our IDS.

5.2. Experimental results

Here we illustrate the results achieved by evaluating our IDS against the three datasets described above. The UCAE model is trained with images generated from the incoming traffic across the entire monitored network. This approach helps prevent the UCAE from overfitting to the traffic patterns of a specific host. Testing is performed on a *per host* basis, i.e., the trained UCAE model is replicated for each monitored host, with each replica exclusively processing traffic destined for a single host.

Table 2
Dataset statistics.

Dataset	Monitored Hosts	Peer Hosts	#Attacks	Traffic Volume
OWN_DATA	34	6786	272	train. 12.5M test 4M
CICIDS18	2	248	2	train. 1.6M test 400K
5G_NIDD	2	141	4	train. 1.8M test 500K

Table 3
Statistics for slow DoS attacks across datasets.

Dataset	Attack	Traffic Rate (pkt/sec)	Total Traffic (pkts)	Duration (min)
OWN_DATA	Slowloris 1	2.5	1'376	9
	Slowloris 2	4.5	2'715	10
	Slow Read	221.8	133'334	10
	S. HTTP POST	4.2	2'498	10
CICIDS18	Slowloris	33.9	85'030	41
	S. HTTP POST	33.2	105'550	53
5G_NIDD	Slowloris 1	22.5	14'649	10.8
	Slowloris 2	31.9	16'557	8.6
	S. HTTP POST 1	219.9	138'174	10.5
	S. HTTP POST 2	212.0	139'363	11

Table 4
Parameter values used for evaluation.

Parameter	Description
$w = 50$	image width
$n = 500$	FPR evaluation window ($\rightarrow 0.23 \leq Tr \leq 0.78$)
$k = 1000$	Suspicious Activity Window

We use the parameters reported in Table 4, then we show how detection performance varies as parameter values change (except for parameter n which does not affect detection as explained in Section 4.3). With $n = 500$ we obtain the anomaly rate threshold Tr to vary between 0.23 and 0.78 across all the 38 monitored hosts. Note that this configuration should not be interpreted as universally optimal. Rather, as explained in Section 4, the proposed framework is designed to be adapted to different network environments by calibrating parameters based on the benign traffic of the specific network under monitoring. Here, for ease of presentation, we adopt the same configuration for the three datasets.

5.2.1. Sensitivity analysis on the anomaly threshold γ

To analyze how performance changes based on the anomaly threshold γ, we conduct tests for different percentiles $\mathbf{P_q}$, i.e. with $\mathbf{q} \in \{50, 75, 80, 85, 90, 95, 98, 100\}$. Experimental findings are reported in Table 5, where: "**AD mod. FP**" and "**Filter FP**" denote the percentage of normal images misclassified as anomalous by the AD module and the Filter, respectively; "**prec.**" and "**rec.**" are precision and recall of the attack detection task performed by the ID module; and "**legit. IP**" denotes the percentage of legitimate IP addresses that are erroneously marked as attackers by the ID module.

As expected, the proportion of normal images misclassified as anomalous by the AD module increases with $\mathbf{P_q}$. In addition, the experimental results reveal three further key insights. First, when the AD module misclassifies a significant number of images, the Filter effectively compensates for this by reducing false positives by half and, in some cases, by up to two orders of magnitude. Second, the ID module successfully detects nearly all attacks, achieving a recall between 0.993 and 1 in almost all cases, but raises some false alarms, as indicated by the precision which is below 1 in most cases. However, the effect of such false positives is minimal. Indeed, the "**legit. IP**" column shows that the percentage of legitimate IP addresses mistakenly flagged as malicious remains below 3% of total peer hosts and below 1% in most cases. Even if the system were to block these misclassified IPs from further communication, each monitored host would still function normally, maintaining connectivity with the remaining 97%/99% of legitimate peers, rather than being entirely disrupted by the attack. Finally, we can observe that the choice of $\mathbf{P_q}$ used to compute the anomaly threshold γ is not highly critical. In fact, any $\mathbf{P_q}$ with $50 \leq q \leq 95$ enables the detection of all attacks, demonstrating the robustness of the approach across different threshold settings.

5.2.2. Sensitivity analysis on the image width w

To assess how performance changes with image width, we conduct additional experiments using images with 25, 100, and 200 columns. The corresponding results are reported in Tables 6–8. For readability, Fig. 6 summarizes performance averaged across error percentiles, clearly showing that detection quality degrades as image width increases. In particular, we observe a general decline in precision and, consequently, in the ability to distinguish malicious IPs from legitimate ones. We attribute this behavior to two main factors. First, larger images encode a greater volume of traffic, which makes the reconstruction error more sensitive to benign random fluctuations, which are more likely to be captured in each image. Second, as image width grows, each image aggregates traffic from a larger set of distinct sources, increasing the probability that legitimate IPs co-occur within anomalous images and are therefore mistakenly flagged as malicious.

Notably, high detection performance is maintained across various values of w, with the exception of $w = 200$, where an increased number of false positives leads to a tenfold rise in the percentage of misclassified IPs. Nevertheless, this percentage remains acceptable, staying below 10%. Conversely, image width has a less pronounced impact on recall, which maintains very high values (≈ 1) for $50\% \leq \mathbf{P_q} \leq 95\%$, in the majority of cases.

Table 5
Results obtained with $w = 50$ on the OWN_DATA, CICIDS18, and 5G_NIDD datasets.

	OWN_DATA					CICIDS18					5G_NIDD				
	AD mod.	Filter	ID module			AD mod.	Filter	ID module			AD mod.	Filter	ID module		
P_q	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP
100	0.00%	0.00%	1.000	0.684	0.00%	49.92%	32.95%	0.667	1.000	0.40%	0.00%	0.00%	0.000	0.000	0.00%
98	3.04%	0.06%	1.000	0.875	0.04%	54.74%	35.36%	1.000	1.000	0.00%	0.00%	0.00%	0.000	0.000	0.00%
95	10.34%	0.15%	0.996	0.993	0.03%	69.38%	42.68%	1.000	1.000	0.00%	1.74%	0.00%	1.000	1.000	2.84%
90	18.82%	0.59%	0.982	1.000	0.12%	78.04%	47.00%	1.000	1.000	0.00%	1.90%	0.00%	1.000	1.000	2.84%
85	24.84%	0.94%	0.982	1.000	0.12%	82.86%	49.42%	1.000	1.000	0.00%	1.92%	0.00%	1.000	1.000	2.84%
80	36.99%	1.18%	0.971	1.000	0.15%	89.68%	52.83%	1.000	1.000	0.00%	1.99%	0.00%	1.000	1.000	2.84%
75	42.43%	0.94%	0.982	1.000	0.10%	96.51%	56.24%	1.000	1.000	0.00%	2.07%	0.00%	1.000	1.000	2.84%
50	68.38%	0.19%	0.993	1.000	0.06%	99.83%	57.90%	1.000	1.000	0.00%	11.32%	0.01%	1.000	1.000	2.84%

Table 6
Results obtained with $w = 25$ on the OWN_DATA, CICIDS18, and 5G_NIDD datasets.

	OWN_DATA					CICIDS18					5G_NIDD				
	AD mod.	Filter	ID module			AD mod.	Filter	ID module			AD mod.	Filter	ID module		
P_q	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP
100	0.00%	0.00%	1.000	0.202	0.00%	100.00%	100.00%	0.667	0.000	0.81%	0.00%	0.00%	0.000	0.000	0.00%
98	2.99%	0.17%	0.996	0.846	0.03%	100.00%	1.48%	1.000	0.000	0.40%	0.00%	0.00%	0.000	0.000	0.00%
95	10.04%	0.17%	0.996	0.996	0.01%	100.00%	1.48%	1.000	1.000	0.40%	1.09%	0.00%	1.000	1.000	0.00%
90	19.93%	0.61%	0.996	0.996	0.01%	100.00%	1.48%	1.000	1.000	0.40%	1.20%	0.00%	1.000	1.000	0.00%
85	30.59%	5.09%	0.986	1.000	0.06%	100.00%	1.48%	1.000	1.000	0.40%	1.26%	0.00%	1.000	1.000	0.00%
80	35.58%	0.64%	0.989	1.000	0.06%	100.00%	1.48%	1.000	1.000	0.40%	1.31%	0.00%	1.000	1.000	0.00%
75	40.52%	1.69%	0.993	1.000	0.04%	100.00%	1.48%	1.000	1.000	0.40%	1.43%	0.00%	1.000	1.000	0.00%
50	57.47%	0.10%	0.996	1.000	0.01%	100.00%	1.48%	1.000	1.000	0.40%	14.80%	0.03%	1.000	1.000	0.00%

Table 7
Results obtained with $w = 100$ on the OWN_DATA, CICIDS18, and 5G_NIDD datasets.

	OWN_DATA					CICIDS18					5G_NIDD				
	AD mod.	Filter	ID module			AD mod.	Filter	ID module			AD mod.	Filter	ID module		
P_q	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP
100	0.00%	0.00%	1.000	0.680	0.07%	100.00%	100.00%	0.667	1.000	0.40%	0.00%	0.00%	0.000	0.000	0.00%
98	3.31%	0.14%	0.978	0.993	0.44%	100.00%	1.70%	1.000	1.000	0.00%	0.00%	0.00%	0.000	0.000	0.00%
95	10.00%	0.19%	0.961	0.996	0.41%	0.00%	0.00%	1.000	1.000	0.00%	4.62%	0.04%	0.800	1.000	2.84%
90	22.20%	2.19%	0.889	1.000	1.09%	0.00%	0.00%	1.000	1.000	0.00%	5.72%	0.05%	1.000	1.000	2.84%
85	30.61%	3.42%	0.847	1.000	1.46%	0.00%	0.00%	1.000	1.000	0.00%	6.94%	0.07%	1.000	1.000	3.55%
80	37.62%	2.44%	0.850	1.000	1.49%	0.00%	0.00%	1.000	1.000	0.00%	8.31%	0.09%	1.000	1.000	2.84%
75	41.27%	1.86%	0.872	1.000	1.28%	0.00%	0.00%	1.000	1.000	0.00%	8.63%	0.09%	1.000	1.000	2.84%
50	55.26%	0.38%	0.938	1.000	0.78%	0.00%	0.00%	1.000	1.000	0.00%	17.41%	0.22%	0.800	1.000	3.55%

Table 8
Results obtained with $w = 200$ on the OWN_DATA, CICIDS18, and 5G_NIDD datasets.

	OWN_DATA					CICIDS18					5G_NIDD				
	AD mod.	Filter	ID module			AD mod.	Filter	ID module			AD mod.	Filter	ID module		
P_q	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP	FP	FP	pre.	rec.	legit. IP
100	0.01%	0.01%	0.000	0.000	0.00%	100.00%	100.00%	0.000	1.000	0.81%	0.00%	0.00%	0.000	0.000	0.00%
98	2.70%	1.45%	0.954	0.529	0.52%	0.00%	0.00%	0.000	1.000	0.00%	0.00%	0.00%	0.000	0.000	0.00%
95	9.02%	6.84%	0.935	0.904	0.99%	0.00%	0.00%	0.667	1.000	0.00%	0.23%	0.00%	0.778	1.000	11.35%
90	18.88%	9.05%	0.672	0.996	4.29%	0.00%	0.00%	0.667	1.000	0.00%	0.44%	0.00%	0.700	1.000	10.64%
85	25.14%	13.76%	0.534	1.000	6.40%	0.00%	0.00%	0.667	0.500	0.00%	0.58%	0.00%	0.778	1.000	12.77%
80	36.69%	15.92%	0.433	1.000	8.72%	0.00%	0.00%	0.667	0.500	0.00%	0.92%	0.01%	0.700	1.000	10.64%
75	42.67%	19.12%	0.389	1.000	10.18%	0.00%	0.00%	0.667	0.500	0.00%	1.02%	0.01%	0.778	1.000	11.35%
50	55.71%	24.53%	0.365	1.000	10.85%	0.00%	0.00%	0.667	0.500	0.00%	11.68%	0.13%	0.778	1.000	12.06%

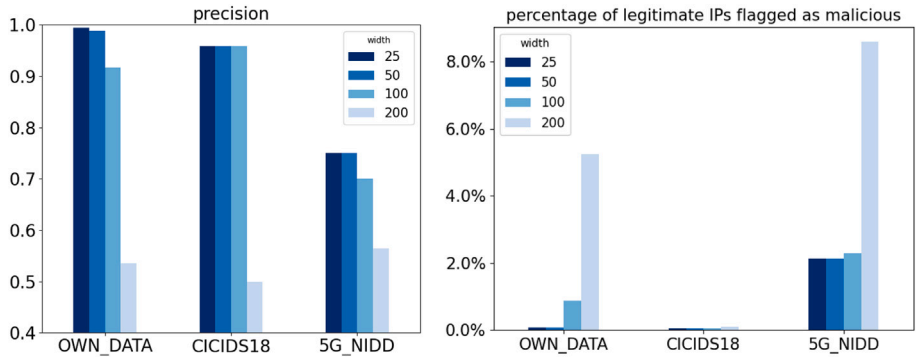

Fig. 6. Precision and percentage of misclassified legitimate IPs by varying the image width. Results are averaged across error percentiles from 50 to 100.

5.2.3. Sensitivity analysis on the SAW

Since the datasets used for evaluation lack the information required to estimate k as described in Section 4.4, we assess the impact of the SAW length on detection performance by conducting tests with $k \in \{1000, 2000, 4000\}$. These values are chosen to be smaller than, comparable to, and greater than the volume of the four attacks, thereby allowing us to clearly highlight the effect of k on detection performance.

The results (Fig. 7) align with the expected outcome: as k increases, the classification task improves in terms of precision. Specifically, there is a reduction in false alarms which leads to fewer legitimate IP addresses being mistakenly flagged as malicious. We recall that the SAW defines the length k of a sequence of anomalies where the ID module looks for evidence of an attack, that is, for an IP address responsible for causing the k anomalies. Therefore, increasing the window size reduces the likelihood of legitimate IP addresses being responsible for *all* anomalies within the SAW. On the other hand, increasing the window size too much worsen the classification task in terms of recall, as attacks with traffic volumes comparable to k may go undetected, or detected too late. The CICIDS18 and 5G_NIDD datasets face this issue only partially. In fact, all attacks in these datasets are detected regardless of the value of k, as their traffic volumes are at least one order of magnitude greater than the largest k used in these tests (see Table 3), so recall remains unaffected. However, precision can still drop when legitimate IPs appear within the SAW alongside the attacker during an attack. The OWN_DATA dataset, instead, presents a different scenario: 3 out of 4 attacks have traffic volumes comparable to the values of k. As a result, several attacks go undetected as k increases, in particular those targeting hosts with low-rate incoming traffic, for which smaller values of k yield better outcomes. These results confirm our rationale of setting k as smaller as possible, as detailed in Section 4.4.1.

5.2.4. Impact on detection time

For an IDS, it is crucial to detect attacks in real-time, that is, before they are fully executed but also early enough to implement effective countermeasures before the targeted service becomes entirely unavailable. Accordingly, we report detection times obtained with the OWN_DATA dataset in Fig. 8 in terms of the attack's progression rather than absolute time. In the histograms, the horizontal axis represents the percentage of attack progression, while the vertical axis represents the number of attacks detected.

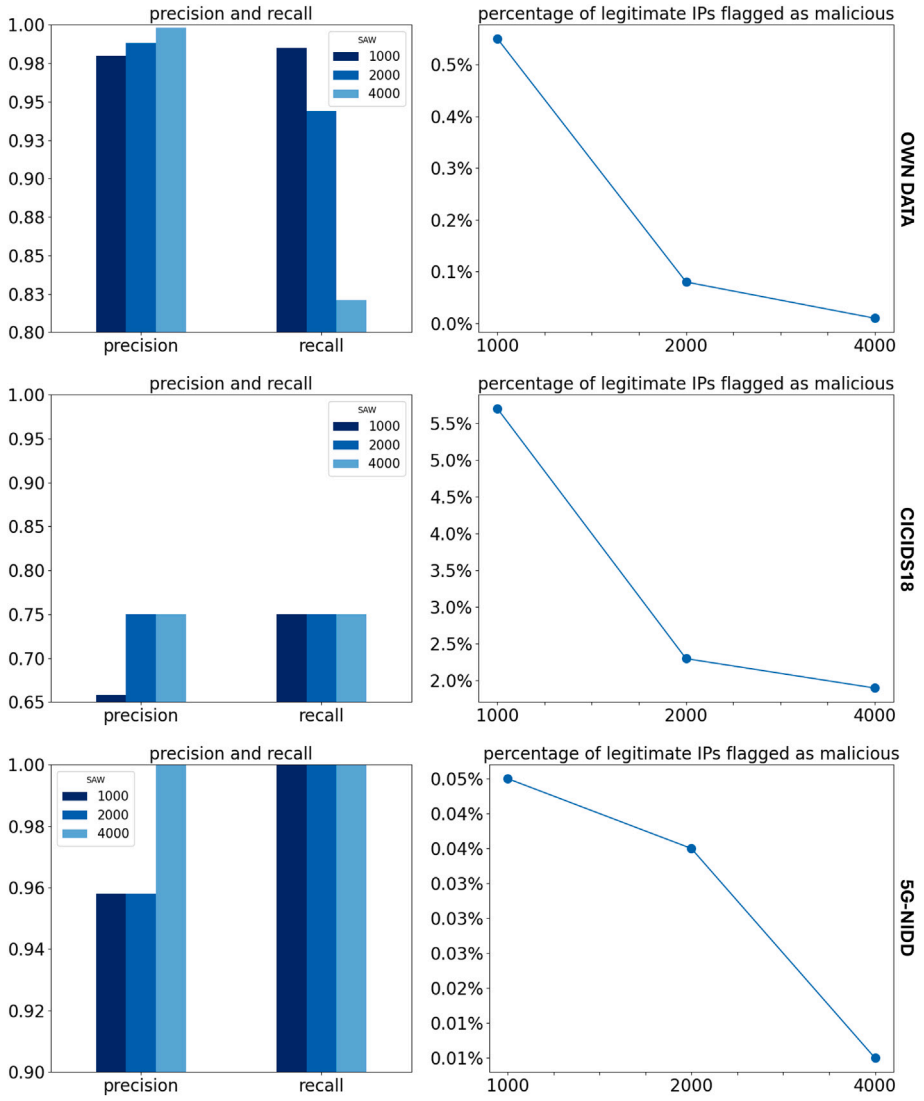

Fig. 7. Precision, recall and percentage of misclassified legitimate IPs by varying the Suspicious Activity Window. Results were obtained with $w = 50$ and averaged across error percentiles from 50 to 100.

As discussed in Section 4.4, the detection time is influenced by the size of the SAW. This is confirmed by the results shown in Fig. 8(top), which illustrate that the time required by the ID module to make a decision increases with the SAW length k. Specifically, we observe that as k grows, the detection time shifts closer to 100% of the attack's progression. Moreover, when the total number of malicious packets in an attack is close to, or smaller than k (as in Slowloris 1, Slowloris 2, and Slow HTTP POST), the detection time becomes more delayed. In contrast, when the attack volume significantly exceeds k (as in Slow Read), detection occurs much earlier. These findings further support our rationale for setting k based on the worst-case scenario, that is, the attack that generates the fewest traffic packets.

Notably, most attacks are detected before reaching 50% of their progression. The Slow Read attack, in particular, is identified in its early stages even with $k = 4000$, due to its traffic volume being two orders of magnitude greater than the SAW size. Overall, these results substantiate the real-time detection capabilities of the proposed method.

Conversely, image width has little impact on detection time, as illustrated in Fig. 8(bottom). This is because two consecutive images differ by only one packet—the last column, as shown in Fig. 5—regardless of the image size. As a result, *all* packets are evaluated by the UCAE, independently of the chosen image width.

5.3. Comparison study

Although directly comparing our approach with existing literature is unconventional due to fundamental differences—ours being packet-based and unsupervised, while most existing solutions are flow-based or supervised—we nevertheless provide a benchmarking of detection methods to contextualize our work within the state of the art. To this end, we compare our work to SOTA approaches that have been evaluated using the 5G_NIDD and CICIDS18 datasets, in terms of precision and recall. We do not use the OWN_DATA dataset for comparison because none of the papers discussed in Section 2 provided implementation code, preventing us from testing SOTA approaches with our traffic data.

Comparison results are summarized in Table 9. Determining which solution best addresses the challenge of slow DoS detection is not straightforward. This is because most existing works focus on detecting network attacks in general rather than specifically targeting slow DoS, and often report evaluation results aggregated across multiple attack scenarios. However, we can assert that our IDS is the only approach capable of detecting *all attacks of interest* (recall = 1), even though it does produce some false positives (precision < 1). However, as demonstrated earlier, these false positives have a negligible impact on monitored hosts, as they result in only a minimal number of benign IP addresses being

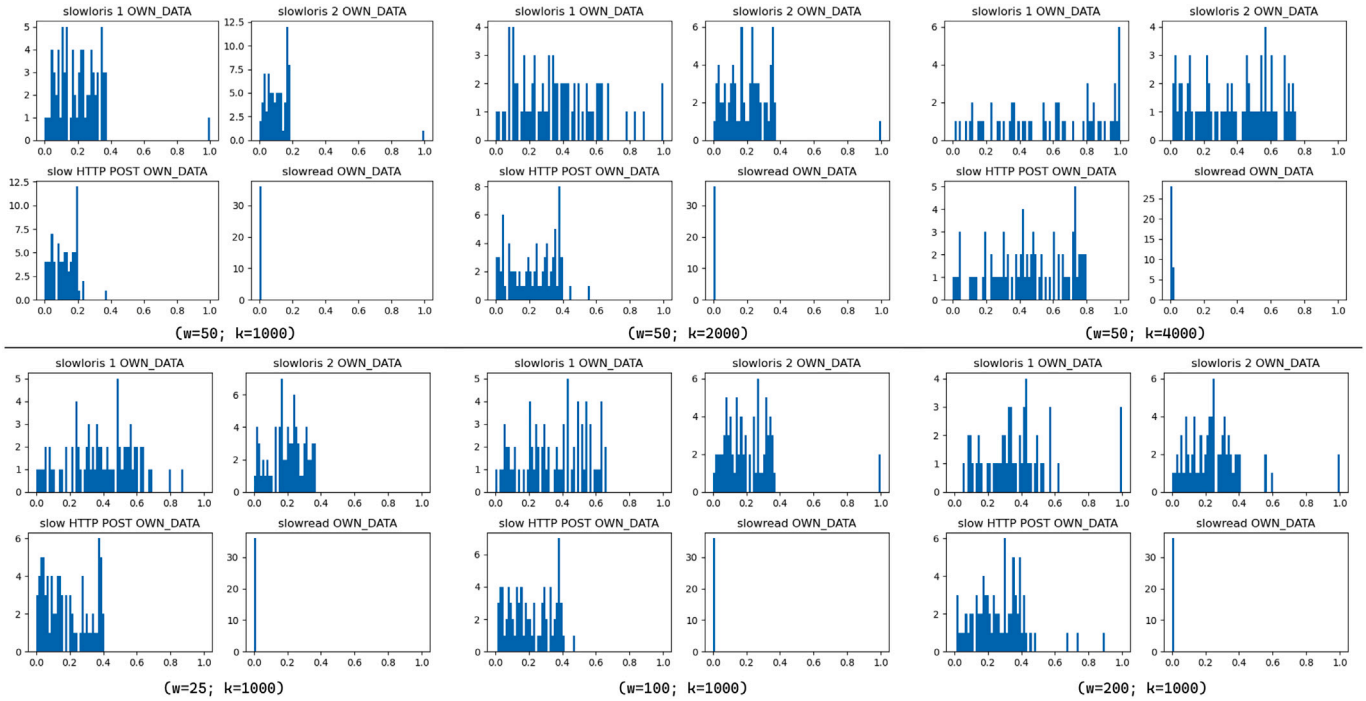

Fig. 8. How detection time varies with the SAW (top), and with the image width (bottom) in the OWN_DATA dataset with $\mathbf{P_q} = 95\%$.

Table 9
Comparison between our method and SOTA approaches evaluated on the 5G_NIDD and CICIDS18 datasets.

references	learning mode	traffic analysis	precision	recall	F1	note	dataset
[28,30,39]	supervised	flow based	0.98	0.99	0.985		5G_NIDD
[37]	supervised	packet based	0.99	0.99	0.99		
[65]	unsupervised	flow based	**1.00**	0.84	0.91		
[17]	unsupervised	flow based	0.88	0.90	0.89	results aggregated over multiple attacks	
[12]	supervised	flow based	0.97	0.90	0.93	results aggregated over multiple attacks	
this work	unsupervised	packet based	**1.00**	**1.00**	**1.00**	$w = 25$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			1.00	**1.00**	**1.00**	$w = 50$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			0.93	**1.00**	0.96	$w = 100$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			0.75	**1.00**	0.86	$w = 200$, results averaged for $\mathbf{P_q} \in [50, 95]$	
[48,55]	supervised	flow based	0.99	0.96	0.975	results aggregated over multiple attacks	CICIDS18
[22]	supervised	flow based	0.81	0.82	0.815	results aggregated over multiple attacks	
[21,26]	supervised	flow based	0.96	0.97	0.965	results aggregated over multiple attacks	
this work	unsupervised	packet based	**1.00**	**1.00**	**1.00**	$w = 25$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			1.00	**1.00**	**1.00**	$w = 50$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			1.00	**1.00**	**1.00**	$w = 100$, results averaged for $\mathbf{P_q} \in [50, 95]$	
			0.67	0.67	0.67	$w = 200$, results averaged for $\mathbf{P_q} \in [50, 95]$	

mistakenly blocked. In contrast, false negatives (recall < 1) pose a significantly greater risk, as a single undetected attack could disrupt all communications.

A further comparison study that we propose juxtaposes the performance of UCAE against other competing machine learning models. To this end, we conduct experiments using both neural architectures and shallow methods. To highlight the advantages of the image-based approach, we also include models that operate on one-dimensional vector representations. Specifically, the one-dimensional models include: U-Net autoencoder (**UAE**), which is similar to the UCAE but without the convolutional component; Gaussian Mixture Model (**GMM**); K-Nearest Neighbors (**KNN**); One Class Support Vector Machine (**OCSVM**); and Isolation Forest (**ISF**). For the models that process two-dimensional vectors we consider: Convolutional Autoencoder (**CAE**), which is similar to the UCAE but without skip connections; and Sparse Convolutional Autoencoder (**SCAE**).

Tables 10 and 11 present the F1 score for the attack detection task when the AD module is implemented with each of the previously mentioned models. Three key observations emerge from these results. First, DL based models outperform shallow methods. Second, while the UCAE model does not consistently achieve the highest performance in all scenarios, it is the most reliable across different error percentiles and datasets, consistently delivering acceptable results (see Table 11). Third, image-based approaches outperform traditional "flat" methods,

Table 10
Comparison of the UCAE model with shallow methods in terms of F1-Score. Results obtained with $w = 50$ and $\mathbf{P_q} = 95\%$.

model	OWN_DATA	CICIDS18	5G_NIDD
UCAE	**0.945**	**0.982**	**0.944**
GMM	0.819	0.254	0.927
OCSVM	0.703	0.337	0.418
KNN (K = 5)	0.935	0.561	0.936
ISF	0.744	0.330	0.884

Table 11
Comparison of the UCAE model with other neural architectures in terms of F1 score. Legend: **U**-Unet; **C**-Convolutional; **AE**-Auto Encoder; **S**-Sparse. Results obtained with $w = 50$.

	OWN_DATA				CICIDS18				5G_NIDD			
P_q	UCAE	UAE	CAE	SCAE	UCAE	UAE	CAE	SCAE	UCAE	UAE	CAE	SCAE
100	0.420	**0.644**	0.409	0.511	0.465	0.465	0.465	**0.468**	0.219	**0.387**	0.219	0.219
98	**0.973**	0.824	0.957	0.958	0.464	0.283	**0.987**	**0.987**	0.219	**0.449**	0.219	0.219
95	**0.945**	0.933	0.935	0.936	**0.982**	0.282	0.980	0.980	**0.944**	0.929	0.806	0.799
90	**0.884**	0.869	0.883	0.878	**0.981**	0.480	0.458	0.860	**0.935**	0.922	0.924	0.924
85	0.822	0.825	**0.836**	0.833	**0.979**	0.477	0.449	0.826	0.920	0.921	**0.929**	**0.929**
80	0.780	0.786	0.798	**0.800**	**0.976**	0.475	0.446	0.821	**0.918**	0.908	**0.918**	0.916
75	**0.784**	0.757	0.758	0.778	**0.975**	0.472	0.439	0.811	**0.912**	0.899	0.900	0.898
50	0.670	0.634	**0.689**	0.687	**0.942**	0.462	0.429	0.805	0.775	0.000	**0.807**	0.805

Table 12
Ablation study: F1 score computed with the TDRL, AAS and RAB methodologies (**all**); with MAE in place of the Threshold-Driven reconstruction loss (**-TDRL**); without the Risk-Aware binarization (**-RAB**); without the Anomaly-Aware scaling approach (**-AAS**). Results obtained with $w = 50$.

	OWN_DATA				CICIDS18				5G_NIDD			
P_q	all	-TDRL	-RAB	-ASS	all	-TDRL	-RAB	-ASS	all	-TDRL	-RAB	-ASS
100	**0420**	0368	0411	0367	**0465**	0464	**0465**	0464	**0219**	**0219**	**0219**	**0219**
98	**0973**	0362	0956	0362	0464	0464	**0968**	0464	**0219**	**0219**	**0219**	**0219**
95	**0947**	0937	0945	0784	**0982**	0464	0475	0464	**0944**	0749	0791	0316
90	0884	0871	**0886**	0784	**0981**	0464	0425	0464	**0935**	0762	0913	0320
85	**0844**	0833	0822	0824	**0979**	0464	0303	0362	0920	0766	**0929**	0295
80	**0811**	0806	0780	0799	**0976**	0464	0259	0350	**0918**	0771	0886	0285
75	**0784**	0752	0773	0773	**0975**	0464	0241	0344	**0912**	0775	0866	0270
50	**0670**	0606	0622	0641	**0942**	0464	0155	0335	0775	**0922**	0760	0147
mean→	**0.792**	0.692	0.774	0.667	**0.846**	0.464	0.411	0.406	**0.730**	0.648	0.698	0.259

with the UAE model being the sole exception in only 3 out of the 24 tests conducted. These results reinforce the validity of our choice to represent traffic data as images rather than one-dimensional vectors.

5.4. Ablation study

We conclude this experimental section with an ablation study aimed at demonstrating the effectiveness of our design choices. Specifically, we analyze the impact of three key methodologies—Risk-Aware Binarization (**RAB**), Anomaly-Aware Scaling (**AAS**), and Threshold-Driven Reconstruction Loss (**TDRL**)—on the classification performance of the AD module. To this end, we conduct three experiments per dataset, where each experiment replaces one of these methodologies with a more traditional alternative. The experimental setups are as follows:

- **[- RAB]** : instead of using the RAB-based approach, categorical features are mapped to a set of equally spaced real values in the range $[0, 1]$. The involved features are `protocol_L4` and `dst_port`.
- **[- ASS]** : instead of using the AAS-based approach, continuous and binary features are normalized directly to the range $[0, 1]$. The involved features are `size_L7` and `TCP_win`.
- **[- TDRL]** : the TDRL is replaced with the Mean Absolute Error (MAE), i.e., the reconstruction error is computed as the pixel-wise average error. Using the notation introduced in Section 4.2, the MAE between the reconstructed image **x** and the original image **y** is defined as

$$MAE(\mathbf{x}, \mathbf{y}) = \frac{1}{N \cdot M} \sum_{i \in \{1,N\}} \sum_{j \in \{1,M\}} |x_{ij} - y_{ij}|$$

Table 12 presents the F1 score for the classification task performed by the UCAE under the settings described above. Column "**all**" shows the F1 score achieved when combining all three methodologies. Results confirm that each methodology contributes its own value to the image classification task. Notably, AAS has the most significant impact, as its exclusion results in the lowest average F1 score across all error percentiles for the three datasets, as shown in the last row of Table 12.

Table 13
The FPR observed after image classification (**AD module**), after filtering (**Filter**), and the number of legitimate IPs erroneously marked as malicious out of a total of 281 IPs (**ID module**). Results obtained with $w = 50$ and $\mathbf{P_q} = 95\%$.

perc	AD module	Filter	ID module
100%	0.062	0.041	4/281 (1.42%)
75%	0.047	0.030	3/281 (1.06%)
50%	0.021	0.000	0/281 (0.00%)
25%	0.000	0.000	0/281 (0.00%)

5.5. Robustness of the AAS approach

We demonstrate that the use of the anomaly-aware scaling (AAS) methodology does not degrade the performance of the IDS when it analyzes legitimate traffic characterized by anomalous data. To this end, we randomly select a benign IP address from the OWN_DATA dataset and modify a certain percentage *perc* of its incoming traffic so that the continuous feature values (AAS applies to continuous features only) fall outside the "normal" range observed during training. Finally, we measure the false positive rate (FPR) in terms of misclassified images, that arise when testing the IDS with this legitimate-but-abnormal traffic, for different values of *perc*. The results in Table 13 indicate that when abnormal traffic is up to 50%, the FPR generated by the Filter remains at 0, preventing the ID module from triggering any alarms. Even when abnormal traffic reaches 100%, the FPR after filtering remains very low (0.041), and the percentage of legitimate IPs mistakenly flagged as malicious is still minimal (4 out of 281 IPs - 1.42%). This confirms that: *(i)* under normal traffic conditions, the AAS approach does not negatively affect the detection process, except in cases of high abnormal traffic volume, where its impact remains minimal; and *(ii)* since the FPR produced by the AD module is very low, we can conclude that, for the

reconstruction error to exceed the anomaly threshold, anomalous values must occur not just in the continuous features but also in the binary and/or categorical features, which is typically what happens during a slow DoS attack.

5.6. Results discussion

We analyze the results of our evaluation focusing on four key findings that underscore the practicality and robustness of our approach in real-world scenarios.

Robustness to Parameter Tuning. The results we have just presented demonstrate that our approach maintains high performance across a wide range of error percentiles ($50\% \leq \mathbf{P_q} \leq 95\%$) and image sizes ($25 \leq w \leq 100$). Notably, the method does not rely heavily on parameter tuning to achieve strong performance; in practice, we only recommend choosing $\mathbf{P_q}$ between 50 and 95 and w between 25 and 100. This suggests the method is largely robust to parameter selection which, combined with its unsupervised nature and real-time detection capabilities, is an important advantage for real-world deployment.

Early Detection Capability. The majority of attacks are detected before reaching 50% of their progression. This demonstrates the effectiveness of the proposed method in identifying threats in real-time and early enough to enable timely countermeasures, thereby ensuring the continued availability of the service. Note that the real-time aspect of the proposed IDS does not refer to strict temporal guarantees measured in absolute time units. Instead, the framework is time-agnostic since it does not rely on time-related features. In this context, real-time detection means identifying attacks during their execution while the targeted service is still partially available, rather than after attack completion.

Effectiveness of Packet-Level Analysis. The loss of flow-level semantics is not a limitation when a more lightweight, packet-level analysis already delivers strong detection performance. This is supported by our empirical results which demonstrate that our method always achieves recall ≈ 1 for $25 \leq w \leq 100$ on three datasets, and that it generally outperforms other flow-based approaches.

Contribution of pre- and post-processing. The proposed pre- and post-processing techniques are essential to the overall detection pipeline. Pre-processing enhances the UCAE's ability to differentiate between normal and malicious traffic, as confirmed by our ablation study, where AAS emerges as the most impactful component. Post-processing, which comprises the Filter and the ID modules, reduces false alarms and enables timely identification of the attacker, thereby mitigating the consequences of false positives. In practice, the neural model alone is not sufficient, it becomes truly effective only when embedded within a broader framework that incorporates complementary mechanisms designed to improve classification accuracy and reduce errors.

6. Conclusions

In this paper, we introduced a deep learning-based approach able to reveal slow DoS attacks working at the packet level and without relying on prior knowledge of attack patterns. The proposed framework follows a three-phase approach: during preprocessing, raw network packets are transformed into images; next, a convolutional autoencoder, trained in an unsupervised fashion, is leveraged to classify images as either normal or anomalous; finally, in the postprocessing phase, the classification results are refined, and the attacker's IP, if any, is identified.

Although flow-level representations provide a more comprehensive view of communication patterns, our packet-level approach offers a valid alternative that avoids the computational overhead of flow reconstruction, remains fully unsupervised, and is designed to support real-time detection by processing traffic as packets arrive. The real-time claim is substantiated by the system's packet-based decision mechanism and by experimental results showing that attacks are detected during their execution rather than after completion. The empirical evaluation demonstrates that most attacks are identified before reaching 50% of their progression.

The proposed approach is also a novel contribution to network intrusion detection, as the combination of unsupervised learning with packet-level traffic analysis has not been previously explored for the detection of slow DoS attacks. A likely reason for this gap is the inherent challenge of extracting sufficient information for effective detection from a limited set of data. Indeed, most existing IDSs leverage traffic flow characteristics and/or predefined attack patterns, which offer significantly more information for detection than merely analyzing normal traffic patterns in terms of a handful of packet-level features. Nonetheless, our experiments demonstrate that achieving effective unsupervised, packet-based detection is still possible.

As future work, we plan to develop model explanation techniques, as our image-based framework is well-suited for such extensions. This would support operators in more accurately assessing uncertain cases, thereby enhancing the system's interpretability and trustworthiness. We also plan to extend our framework as a foundation for developing unsupervised, real-time intrusion detection systems targeting attacks beyond slow DoS. To this end, we will explore how parameter tuning can be leveraged to adapt the system's focus to different types of threats, enabling the creation of multiple IDS instances—each tailored to a specific threat, yet built on a shared core engine. This modular approach would offer a potential alternative to the *one-fits-all* solutions commonly found in the literature, which often struggle to maintain high detection performance across diverse attack types.

CRediT authorship contribution statement

Antonino Rullo: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Enrico Cambiaso:** Software, Data curation. **Massimo Guarascio:** Writing – review & editing, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and by Project PERUN (101225653) - Protecting Sensitive Cyber Ecosystems from Upcoming Next Generation and AI-generated Malware Threats under the Horizon Europe Program funded by the European Union.

Data availability

The authors do not have permission to share data.

References

[1] M. Aiello, E. Cambiaso, M. Mongelli, G. Papaleo, An on-line intrusion detection approach to identify low-rate DoS attacks, in: 2014 International Carnahan Conference on Security Technology (ICCST), IEEE, 2014, pp. 1–6.
[2] N.A. Al-Shukaili, M.L.M. Kiah, I. Ahmedy, Optimizing feature selection and deep learning techniques for precise detection of low-rate distributed denial of service (LDDoS) attack, Discov. Internet Things 5 (1) (2025) 80.
[3] A.F. Al-Zubidi, A.K. Farhan, S.M. Towfek, Predicting DoS and DDoS attacks in network security scenarios using a hybrid deep learning model, J. Intell. Syst. 33 (1) (2024) 20230195.
[4] A.A. Alashhab, M.S. Zahid, B. Isyaku, A.A. Elnour, W. Nagmeldin, A. Abdelmaboud, T.A.A. Abdullah, U.D. Maiwada, Enhancing DDoS attack detection and mitigation in SDN using an ensemble online machine learning model, IEEE Access 12 (2024) 51630–51649.
[5] M. Asad, M. Asim, T. Javed, M.O. Beg, H. Mujtaba, S. Abbas, Deepdetect: detection of distributed denial of service attacks using deep learning, Comput. J. 63 (7) (2020) 983–994.
[6] E. Cambiaso, F. Folino, M. Guarascio, A. Liguori, A. Rullo, Seeing the invisible: detection of stealth DoS attacks using variational U-Net-like models, J. Inf. Secur. Appl. 97 (2026) 104356.

[7] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Slow DoS attacks: definition and categorisation, International Journal of Trust Management in Computing and Communications 1 (3–4) (2013) 300–319.

[8] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Designing and modeling the slow next DoS attack, in: International Joint Conference: CISIS'15 and ICEUTE'15, Springer, 2015, pp. 249–259.

[9] J. Chen, H. Wu, X. Wang, S. Wang, G. Cheng, X. Hu, IEA-DMS: an interpretable feature-driven, efficient and accurate detection method for slow HTTP DoS in high-speed networks, Comput. Secur. 150 (2025) 104291.

[10] V. de Miranda Rios, P.RM. Inácio, D. Magoni, M.M. Freire, Detection of reduction-of-quality DDoS attacks using fuzzy logic and machine learning algorithms, Comput. Netw. 186 (2021) 107792.

[11] L. De Rose, G. Andresini, A. Appice, D. Malerba, VINCENT: cyber-threat detection through vision transformers and knowledge distillation, Comput. Secur. 144 (2024) 103926.

[12] B. Farzaneh, N. Shahriar, A.H. Al Muktadir, M.S. Towhid, M.S. Khosravani, DTL-5G: deep transfer learning-based DDoS attack detection in 5G and beyond networks, Comput. Commun. 228 (2024) 107927.

[13] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda, Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking, in: Proceedings of the 6th International COnference, 2010, pp. 1–12.

[14] N. Garcia, T. Alcaniz, A. González-Vidal, J.B. Bernabe, D. Rivera, A. Skarmeta, Distributed real-time SlowDoS attacks detection over encrypted traffic using artificial intelligence, J. Netw. Comput. Appl. 173 (2021) 102871.

[15] J. Geiping, L. Fowl, G. Somepalli, M. Goldblum, M. Moeller, T. Goldstein, What doesn't kill you makes you robust (ER): how to adversarially train against data poisoning, arXiv preprint arXiv:2102.13624, 2021.

[16] M. Hiari, Y. Alraba'nah, I. Qaddara, A deep learning-based intrusion detection system using refined LSTM for DoS attack detection, Eng. Technol. Appl. Sci. Res. 15 (4) (2025) 25627–25633.

[17] A. Islam, S.-Y. Chang, J. Kim, J. Kim, Anomaly detection in 5G using variational autoencoders, in: 2024 Silicon Valley Cybersecurity Conference (SVCC), IEEE, 2024, pp. 1–6.

[18] H.H. Jazi, H. Gonzalez, N. Stakhanova, A.A. Ghorbani, Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling, Comput. Netw. 121 (2017) 25–36.

[19] J. Kang, M. Yang, J. Zhang, Accurately identifying new QoS violation driven by high-distributed low-rate denial of service attacks based on multiple observed features, J. Sens. 2015 (1) (2015) 465402.

[20] C. Kemp, C. Calvert, T.M. Khoshgoftaar, Detection methods of slow read DoS using full packet capture data, in: 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), IEEE, 2020, pp. 9–16.

[21] M.A. Khan, HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system, Processes 9 (5) (2021) 834.

[22] J. Kim, J. Kim, H. Kim, M. Shim, E. Choi, CNN-based network intrusion detection against denial-of-service attacks, Electronics 9 (6) (2020) 916.

[23] A. Kuzmanovic, E.W. Knightly, Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003, pp. 75–86.

[24] J. Li, R. Socher, S.CH. Hoi, Dividemix: learning with noisy labels as semi-supervised learning, arXiv preprint arXiv:2002.07394, 2020.

[25] X. Liang, T. Znati, A long short-term memory enabled framework for DDoS detection, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[26] P. Lin, K. Ye, C.-Z. Xu, Dynamic network anomaly detection system by using deep learning techniques, in: Cloud Computing–CLOUD 2019: 12th International Conference, Held as Part of the Services Conference Federation, SCF 2019, Proceedings 12, Springer, 2019, pp. 161–176.

[27] J.C. Louis, R.E. Lee, Introduction to sockstress, a TCP socket stress testing framework, in: SEC-T Security Conference, 2011.

[28] R. Mohammad, F. Saeed, A.A. Almazroi, F.S. Alsubaei, A.A. Almazroi, Enhancing intrusion detection systems using a deep learning and data augmentation approach, Systems 12 (3) (2024) 79.

[29] M. Mongelli, M. Aiello, E. Cambiaso, G. Papaleo, Detection of DoS attacks through Fourier transform and mutual information, in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, pp. 7204–7209.

[30] A. Moubayed, A complete EDA and DL pipeline for softwarized 5G network intrusion detection, Future Internet 16 (9) (2024) 331.

[31] N. Muraleedharan, B. Janet, A deep learning based HTTP slow DoS classification approach using flow data, ICT Express 7 (2) (2021) 210–214.

[32] I. Muscat, Web vulnerabilities: identifying patterns and remedies, Network Security 2016 (2) (2016) 5–10.

[33] A.A. Neloy, M. Turgeon, A comprehensive study of auto-encoders for anomaly detection: efficiency and trade-offs, Mach. Learn. Appl. 17 (2024) 100572.

[34] C. Northcutt, L. Jiang, I. Chuang, Confident learning: estimating uncertainty in dataset labels, J. Artif. Intell. Res. 70 (2021) 1373–1411.

[35] B. Nugraha, R.N. Murthy, Deep learning-based slow DDoS attack detection in SDN-based networks, in: 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2020, pp. 51–56.

[36] S.Z. Omer, F. Hashim, A. Sali, F.A. Ahmad, Binary classification of low-rate DoS attacks using long short-term memory feed-forward (LSTM-FF) intrusion detection system (IDS), Eng. Sci. Technol. Int. J. 66 (2025) 102049.

[37] E. Paolini, L. Valcarenghi, L. Maggiani, N. Andriolli, Real-time network packet classification exploiting computer vision architectures, IEEE Open J. Commun. Soc. 5 (2024) 1155–1166.

[38] A. Paracha, J. Arshad, M.B. Farah, K. Ismail, Deep behavioral analysis of machine learning algorithms against data poisoning: A. paracha et al, Int. J. Inf. Secur. 24 (1) (2025) 29.

[39] C. Park, K. Park, J. Song, J. Kim, Distributed learning-based intrusion detection in 5G and beyond networks, in: 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, 2023, pp. 490–495.

[40] J.A. Perez-Diaz, I.A. Valdovinos, K.-K.R. Choo, D. Zhu, A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning, IEEE Access 8 (2020) 155859–155872.

[41] T.V. Phan, TM.R. Gias, S.T. Islam, T.T. Huong, N.H. Thanh, T. Bauschert, Q-MIND: defeating stealthy DoS attacks in SDN with a machine-learning based defense framework, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[42] B.A. Pratomo, P. Burnap, G. Theodorakopoulos, Unsupervised approach for detecting low rate attacks on network traffic with autoencoder, in: 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2018, pp. 1–8.

[43] V. Punitha, C. Mala, N. Rajagopalan, A novel deep learning model for detection of denial of service attacks in HTTP traffic over internet, International Journal of Ad Hoc and Ubiquitous Computing 33 (4) (2020) 240–256.

[44] O. Ronneberger, P. Fischer, T. Brox, U-Net: convolutional networks for biomedical image segmentation, in: Proc. of Int. Conf. on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.

[45] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, M. Ylianttila, 5G-nidd: a comprehensive network intrusion detection dataset generated over 5G wireless network, arXiv preprint arXiv:2212.01298, 2022.

[46] F. Scala, M. Guarascio, C. Parrotta, L. Pontieri, Learning fast to detect slow: a few-shot neural approach to slow DoS attack detection, in: International Conference on Discovery Science, Springer, 2025, pp. 347–362.

[47] J. Verschuren, R. Govaerts, J. Vandewalle, ISO-OSI security architecture, in: Computer Security and Industrial Cryptography: State of the Art and Evolution ESAT Course, Leuven, Belgium, May 21–23, 1991, Springer, 2023, pp. 179–192.

[48] S. Seth, G. Singh, K. Kaur Chahal, A novel time efficient learning-based approach for smart intrusion detection system, J. Big Data 8 (1) (2021) 111.

[49] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, et al., Toward generating a new intrusion detection dataset and intrusion traffic characterization, ICISSp 1 (2018) 108–116.

[50] M. Sikora, A. Krivulcik, R. Fujdiak, P. Blazek, Design of advanced slow denial of service attack generator, in: 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE, 2020, pp. 99–104.

[51] K.J. Singh, T. De, MLP-GA based algorithm to detect application layer DDoS attack, J. Inf. Secur. Appl. 36 (2017) 145–153.

[52] J. Steinhardt, P.W.W. Koh, P.S. Liang, Certified defenses for data poisoning attacks, Adv. Neural Inf. Process. Syst. 30 (2017).

[53] K.M. Sudar, P. Deepalakshmi, Flow-based detection and mitigation of low-rate DDoS attack in SDN environment using machine learning techniques, in: IoT and Analytics for Sensor Networks: Proceedings of ICWSNUCA 2021, Springer, 2021, pp. 193–205.

[54] Y.M. Swe, P.P. Aung, A.S. Hlaing, A slow DDoS attack detection mechanism using feature weighing and ranking, in: Proceedings of the International Conference on Industrial Engineering and Operations Management, 2021, pp. 4500–4509.

[55] M.A. Talukder, M.M. Islam, M.A. Uddin, K.F. Hasan, S. Sharmin, S.A. Alyami, M.A. Moni, Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction, J. Big Data 11 (1) (2024) 33.

[56] D. Tang, R. Dai, L. Tang, X. Li, Low-rate DoS attack detection based on two-step cluster analysis and UTR analysis, Hum.-centric Comput. Inf. Sci. 10 (1) (2020) 6.

[57] D. Tang, J. Man, L. Tang, Y. Feng, Q. Yang, WEDMS: an advanced mean shift clustering algorithm for LDoS attacks detection, Ad Hoc Netw. 102 (2020) 102145.

[58] D. Tang, Y. Yan, S. Zhang, J. Chen, Z. Qin, Performance and features: mitigating the low-rate TCP-targeted DoS attack via SDN, IEEE J. Sel. Areas Commun. 40 (1) (2021) 428–444.

[59] D. Tang, S. Zhang, J. Chen, X. Wang, The detection of low-rate DoS attacks using the SADBSCAN algorithm, Inf. Sci. 565 (2021) 229–247.

[60] N. Tripathi, N. Hubballi, Application layer denial-of-service attacks and defense mechanisms: a survey, ACM Computing Surveys (CSUR) 54 (4) (2021) 1–33.

[61] D. Tymoshchuk, O. Yasniy, M. Mytnyk, N. Zagorodna, V. Tymoshchuk, Detection and classification of DDoS flooding attacks by machine learning method, arXiv preprint arXiv:2412.18990, 2024.

[62] Z. Wu, L. Zhang, M. Yue, Low-rate DoS attacks detection based on network multifractal, IEEE Trans. Dependable Secure Comput. 13 (5) (2015) 559–567.

[63] Y. Xiang, K. Li, W. Zhou, Low-rate DDoS attacks detection and traceback by using new information metrics, IEEE Trans. Inf. Forensics Secur. 6 (2) (2011) 426–437.

[64] C. Xu, J. Shen, X. Du, Low-rate DoS attack detection method based on hybrid deep neural networks, J. Inf. Secur. Appl. 60 (2021) 102879.

[65] L. Yuan, J. Sun, S. Zhuang, Y. Liu, L. Geng, J. Zou, P. Xin, W. Huang, W. Ma, Manticore: an unsupervised intrusion detection system based on contrastive learning in 5G networks, in: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2024, pp. 4705–4709.

[66] N.M. Yungaicela-Naula, C. Vargas-Rosales, J.A. Pérez-Díaz, SDN/NFV-based framework for autonomous defense against slow-rate DDoS attacks by using reinforcement learning, Future Gener. Comput. Syst. 149 (2023) 637–649.

[67] X. Zhang, Z. Wu, J. Chen, M. Yue, An adaptive KPCA approach for detecting LDoS attack, Int. J. Commun. Syst. 30 (4) (2017) e2993.