International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 16, Number 2, pages 224–241, June 2026

Consideration of Network Anomaly Detection Models for DoS/DDoS Attacks in GNS3 Environments using Machine Learning

Souma Kai
Institute of Technology, Okinawa College, 905
Nago, Okinawa, 905-2192, Japan

Chikatoshi Yamada
Institute of Technology, Okinawa College, 905
Nago, Okinawa, 905-2192, Japan

Soichiro Hanashiro
Institute of Technology, Okinawa College, 905
Nago, Okinawa, 905-2192, Japan

Received: February 20, 2026
Revised: May 3, 2026
Accepted: June 8, 2026
Communicated by Shuichi Ichikawa

Abstract

In recent years, the rapid advancement of network technologies has led to increasingly sophisticated cyberattacks, posing significant challenges to traditional rule-based anomaly detection systems. This research proposes and verifies a dual-stage network anomaly detection pipeline that integrates machine learning models to identify both known and unknown threats effectively. The proposed system utilizes a Random Forest (RF) model for the classification of established attack patterns and an Autoencoder (AE) for the detection of unknown anomalies through a reconstruction-based approach.

To ensure a highly reproducible and realistic experimental environment, the virtual network simulator GNS3 was employed to construct a complex topology including attacker, client, and victim nodes. Network traffic data, comprising normal communications and DDoS attacks (SYN Flood and Slowloris), were collected and processed using the NFStream library to extract key statistical features.

The experimental results demonstrate the high performance of the proposed model, with the Random Forest achieving an accuracy of 1.00 in classifying known attacks even within complex traffic scenarios. Furthermore, the Autoencoder successfully identified anomalies with high precision (0.96 accuracy in GNS3-based experiments) by learning normal traffic baselines and detecting deviations through Mean Squared Error (MSE) analysis.

Keywords: machine learning, virtual network, DDoS attack, anomaly detection

[0]This is an abstract footnote

1 Introduction

In recent years, the rapid advancement of network technology has brought immense convenience to our daily lives. However, this progress has also led to the increasing complexity and diversification of cyberattacks. Among these, Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks remain significant threats, as they disrupt network services by transmitting massive amounts of traffic.

Conventionally, rule-based detection systems that identify attacks using registered patterns, known as signatures, have been utilized [1]. While effective against known attacks, these systems struggle to detect unknown "zero-day" attacks and variants of existing ones. Given the constant evolution of attack methodologies, there is an urgent need for more flexible and intelligent detection systems.

Consequently, this research investigates network anomaly detection models using machine learning as a powerful method capable of discovering even novel attacks. Machine learning allows systems to learn the statistical characteristics of network communications without relying on predefined rules, enabling the detection of "abnormal behavior" as anomalies.

A major challenge in constructing such models is the acquisition and evaluation of "realistic data." Many studies rely on outdated datasets that do not reflect contemporary network environments [2, 3]. Furthermore, while recent research, such as that by Wang et al. [4], proposes hybrid configurations combining Random Forest (RF) and Autoencoder (AE), their evaluations are primarily confined to improving accuracy within static benchmark datasets such as IDS2018 and BoT-IoT. Crucially, such studies often bypass the discussion of computational costs and processing delays associated with the feature extraction process itself, as they typically utilize pre-processed CSV data.

To address these limitations, this research adopts a two-pronged evaluation strategy. First, we utilize the NSL-KDD dataset as a benchmark to validate the fundamental detection capabilities of our model, ensuring comparability with a vast body of historical research. Second, to evaluate the system's practical feasibility, we employ GNS3 (Graphical Network Simulator-3) [5] to construct an original virtual network environment. Unlike static datasets, this approach allows us to evaluate the entire pipeline—from generating raw traffic to feature extraction—in a repeatable and controlled manner.

In this environment, we generate and verify the characteristics of attacks such as SYN Flood and Slowloris. Specifically, we focus on single-source "DoS attacks" to analyze the behavior of individual components of DDoS attacks in detail. Even in distributed DDoS attacks, the statistical characteristics observed at the target gateway (Router R1 in this configuration)—such as sudden spikes in packet frequency or anomalies in flow duration—originate from the nature of individual DoS attacks. Therefore, accurately detecting and analyzing single-node DoS attacks is an essential foundational step for defending against large-scale DDoS attacks.

A key contribution of this research is the integration of the NFStream library into our data processing pipeline. While our current evaluation is conducted on captured traffic, the use of NFStream allows us to measure the efficiency of transforming raw packets into flow-based features. Unlike traditional packet-based sequential analysis, NFStream is designed for high-throughput flow processing, providing a critical evaluation of the computational overhead required for future real-time implementations.

While the individual components employed here have appeared in prior studies, the contributions of this work, stated relative to existing research, are threefold. First, in contrast to Wang et al. [4], whose RF–AE evaluation is confined to pre-processed static NetFlow datasets (IDS2018 and BoT-IoT), we evaluate the complete pipeline from raw packet capture to detection within a reproducible GNS3 topology, in which traffic generation, capture, and feature extraction are all controlled and repeatable. Second, we explicitly quantify the feature-extraction latency (0.2181 ms/flow) as a real-time feasibility metric—a cost that evaluations based on pre-processed datasets bypass entirely. Third, we restrict the NFStream feature set to six DDoS-specific discriminators, reducing the extraction cost to approximately one-third of the standard 88-feature configuration reported in [9]. A direct comparison with the most relevant prior work is summarized in Table 9 in Section 8.

The objective of this paper is to validate the effectiveness and efficiency of a two-stage detection pipeline in a GNS3 environment, combining Random Forest for classifying known attacks and Autoencoder for discovering unknown anomalies. This research represents a significant step toward building a practical security system capable of defending against evolving network attacks by bridging the gap between theoretical model accuracy and real-world processing requirements.

The structure of this paper is as follows. Section 2 explains the principles of the machine learning models, and Section 3 describes the experimental environment and network topology in detail. Section 4 explains the data collection and preprocessing, and Section 5 describes the evaluation method. In Sections 6 and 7 and 8, we summarize the experimental results and discuss them, and finally, Section 9 provides the future outlook.

2 Principle

This research employs a two-stage analysis pipeline to simultaneously detect unknown threats and identify known attacks. The system architecture is designed to first classify known attack patterns using a Random Forest model and subsequently identify unknown anomalies through an Autoencoder-based reconstruction approach.

2.1 Classification Principle Using Random Forests

Random Forest (RF) is an ensemble learning method that constructs a multitude of decision trees during the training phase. It excels at classifying known attack patterns by aggregating the predictions of individual trees, which significantly reduces the risk of overfitting compared to a single decision tree architecture.

The classification process relies on the importance of features extracted via NFStream. During the construction of each tree, the optimal split for each node is determined using the Gini Impurity G, which is defined as:

$$G = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{1}$$

where p_i represents the probability of an element belonging to class i at a specific node. By evaluating these probabilities across various network features, the model can effectively distinguish between normal traffic and specific attack types, such as SYN Flood and Slowloris, based on their distinct statistical characteristics.

2.2 Principles of Detecting Unknown Attacks Using Autoencoders

While RF is highly effective for identifying known patterns, Autoencoders (AE) are utilized to detect unknown threats that may not be present in the training labels. An Autoencoder is a class of artificial neural network designed to learn efficient data encodings in an unsupervised manner.

The AE architecture consists of two primary components: an encoder that compresses the input x into a lower-dimensional latent representation h (often referred to as the bottleneck), and a decoder that attempts to reconstruct the original data into $\hat{x}$. The mathematical operations are expressed as follows:

- **Encoder**: $h = \sigma(Wx + b)$
- **Decoder**: $\hat{x} = \sigma(W'h + b')$

where W and W' are weight matrices, b and b' are bias vectors, and σ represents the activation function (such as ReLU or Sigmoid).

The model is trained exclusively on normal communication data, allowing it to internalize the typical characteristics of a stable network environment. The performance and degree of anomaly are

evaluated using the Mean Squared Error (MSE) between the original input and the reconstructed output:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2}$$

When abnormal or previously unseen attack traffic is processed, the AE fails to reconstruct the data accurately because the input deviates from the learned "normal" patterns, resulting in a significantly high MSE. By establishing a dynamic threshold based on the maximization of the F1-score, these statistical deviations are accurately identified as anomalies.

2.3 Feature Selection and Reasoning

To optimize detection performance and ensure computational efficiency, this study focuses on six key statistical features extracted by NFStream. The technical reasoning for selecting these specific metrics is detailed below:

1. **Duration**: This feature measures the total elapsed time of a network flow from its initiation to termination. It is a critical metric for identifying application-layer DoS attacks such as Slowloris. While legitimate HTTP requests are typically completed rapidly, Slowloris attacks intentionally hold connections open for extended periods by sending incomplete requests, resulting in duration values that far exceed normal operational baselines.

2. **Source/Destination Bytes**: These features track the total volume of data (in bytes) transferred in both directions. In many DDoS scenarios, such as SYN Flood, attackers aim to exhaust server resources using a high frequency of packets rather than large payloads. Consequently, a significant disparity or an unusually low byte count relative to the packet frequency often indicates malicious automation rather than human-generated traffic.

3. **Packets**: This represents the total number of packets transmitted during a single flow. It serves as the primary indicator for volumetric attacks like SYN Flood. Since the objective of such attacks is to overwhelm the target's connection table with a massive influx of SYN requests, the packet count per flow deviates sharply from the learned behavior of legitimate users.

4. **Byte per Packet**: This derived feature calculates the average payload size per packet. Attack tools often utilize small, repetitive packets to maximize the interrupt-processing load on network devices. By monitoring this ratio, the model can distinguish between data-heavy legitimate traffic (e.g., file downloads) and the "thin" but aggressive packet streams characteristic of flooding attacks.

5. **Asymmetry**: Legitimate TCP communication generally exhibits a high degree of symmetry between sent and received data due to the request-response nature of protocols like HTTP. Conversely, attack traffic is often highly asymmetric because attackers continuously transmit malicious packets without waiting for or processing the server's responses. This feature effectively captures the unidirectional nature of automated attack scripts.

6. **Flow Rate**: By calculating the frequency of packets or bytes sent over a specific time interval, the model can identify sudden bursts in traffic. This helps the Autoencoder distinguish between normal high-load periods and the sudden, sustained surges typical of a coordinated DDoS attack.

3 Experimental environment

In this study, we built a virtual network environment that is physically isolated from the real network to ensure that experiments are safe and can be repeated many times.

3.1 Selection of Composition Tools

The experimental environment was constructed using the following tools. We chose these tools because they are widely used by network professionals and are suitable for academic research.

- **GNS3 (Graphical Network Simulator-3)**: This is a software that can emulate network devices from many different vendors [5]. We used GNS3 because it allows us to design complex network topologies easily on a GUI and connect virtual routers and PCs in a way that feels like a real network.

- **VMware Workstation**: This is industry-standard virtualization software [6]. We used it to run multiple independent operating systems, such as Kali Linux and Ubuntu, simultaneously on a single physical PC. This makes the experiment very efficient.

- **Kali Linux**: This is a Debian-based Linux distribution made for security testing and digital forensics [7]. We used Kali Linux as the attacker's PC because it already has many attack tools like `hping3` and `slowloris` installed, so we could start the experiment quickly.

- **Ubuntu Server**: This is an open-source server OS that is very stable and lightweight [8]. We used Ubuntu Server for the victim server and normal client PCs because it is widely used in the real world and works fast even in a virtual environment.

3.2 Hardware Specifications

The performance of machine learning training and network simulation depends on the hardware resources of the host PC. In this study, we used a laptop PC (Mouse 20075N-CML) to run the GNS3 environment and train the models. The detailed specifications are shown in Table 1.

Table 1: Hardware specifications of the host PC

Component	**Specification**
Model Name	Mouse 20075N-CML
OS	Windows 11 Home
CPU	Intel Core i7-10750U (2.60GHz)
Memory (RAM)	16GB
GPU	NVIDIA GeForce MX350
Storage	512GB NVMe SSD

This hardware setup allowed us to run the GNS3 router and virtual machines simultaneously. Although the memory is 16GB, we optimized the resource allocation for each VM to ensure the stability of traffic collection and model training.

3.3 Network Topology and Design

The network configuration for this experiment is shown in Figure 1. We built three different LAN segments: the "Attacker LAN," "Client LAN," and "Victim LAN." These are all connected to a virtual router (R1) running a Cisco IOS image.

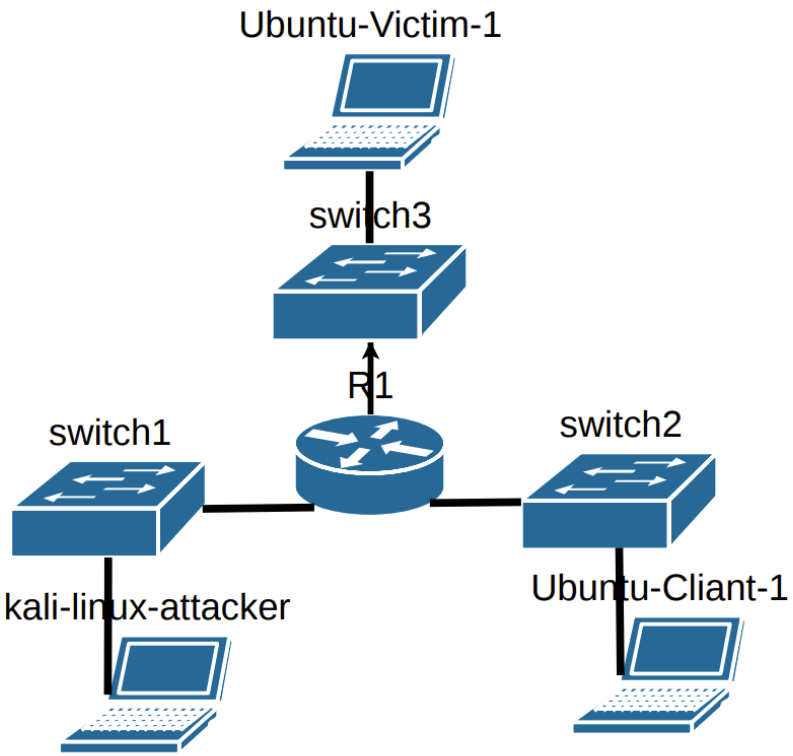

Figure 1: Network topology for simulation

This design is important because all communication between the different segments must pass through the router. This allows us to observe and capture all the traffic in one place, which is very helpful for collecting data for machine learning.

3.4 IP Address and Gateway Design

To make the communication work correctly, each device was assigned a static IP address as shown in Table 2. We also set the default gateway for each virtual machine to the router R1. By doing this, we achieved proper routing between all segments, allowing the attacker to reach the victim and the client to access the server just like in a real-world network.

Table 2: List of IP addresses

Device Role	**IP Address**	**Subnet Mask**	**Default Gateway**
Attacker PC	192.168.1.100	255.255.255.0	192.168.1.1
Client PC	192.168.2.100	255.255.255.0	192.168.2.1
Victim PC	10.0.0.100	255.255.255.0	10.0.0.1
Router (for Attacker)	192.168.1.1	255.255.255.0	—
Router (for Client)	192.168.2.1	255.255.255.0	—
Router (for Victim)	10.0.0.1	255.255.255.0	—

3.5 Software Stack and Implementation Details

To maintain the technical integrity and ensure the reliability of the experimental results, the specific software environment and library versions utilized in this study are explicitly documented. The entire analysis pipeline, including data preprocessing via NFStream and the implementation of the Autoencoder using PyTorch, was developed within a unified Python-based ecosystem.

The selection of these specific frameworks was based on their computational efficiency and their ability to handle the complex statistical features inherent in GNS3-captured traffic. By strictly managing the versions of each library, we aimed to prevent inconsistencies in numerical processing and to ensure that the detection performance remained stable across all experimental trials. Table 3 summarizes the core software stack employed for the implementation.

Table 3: Software stack and library versions

Category	**Library / Tool**	**Version**
Programming Language	Python	3.10.12
Traffic Analysis	NFStream	6.5.3
Machine Learning	Scikit-learn	1.3.0
Deep Learning	PyTorch	2.1.0+cu118
Data Manipulation	Pandas / NumPy	2.1.1 / 1.24.3
Visualization	Matplotlib / Seaborn	3.7.2 / 0.12.2

4 Data Collection and Preprocessing

4.1 Data Collection Scenarios

To evaluate the effectiveness of the proposed anomaly detection system, traffic data was collected from the GNS3-based experimental topology under four distinct scenarios. The objective was to capture a wide range of network behaviors, including both stable communication and statistical fluctuations inherent in real-world environments.

- **Normal Traffic (*normal.pcap*)**: Periodic HTTP requests were generated from the client PC to the victim server using the `curl` command. This dataset represents basic and predictable service access.
- **Complex Normal Traffic (*normal2.pcap*)**: To simulate more realistic network conditions, background network logs were captured from the attacker's PC over a 5-minute period. This dataset contains higher statistical entropy compared to the basic normal traffic, serving as a rigorous baseline for the Autoencoder's reconstruction task.
- **SYN Flood Attack (*attack.pcap*)**: A volumetric DDoS attack was simulated using the `hping3` tool. This attack aims to exhaust server resources by flooding TCP SYN packets with randomized source IP addresses.
- **Slowloris Attack (*slowloris.pcap*)**: A low-and-slow application-layer attack was executed using the `Slowloris` tool. By sending partial HTTP requests and holding connections open, this attack targets service availability while maintaining a low traffic volume.

4.2 Feature Extraction via NFStream

For feature extraction, the Python-based framework **NFStream** was utilized instead of traditional packet-by-packet inspection. This choice was motivated by the need for "Flow-based Analysis," which provides a more comprehensive behavioral profile suitable for machine learning models.

The captured pcap files were aggregated into bidirectional flows. NFStream automatically extracted over 80 statistical features, providing a multidimensional view of the network traffic. Key features utilized in this study include:

- `bidirectional_duration_ms`: Total duration of the flow.
- `bidirectional_packets`: Total number of packets exchanged within the flow.
- `bidirectional_bytes`: Total volume of data transferred.

This automated extraction process ensured that the dataset captured the temporal and behavioral characteristics of each traffic type effectively.

4.3 Data Labeling and Preprocessing

To prepare the dataset for supervised (RF) and unsupervised (AE) learning, a precise labeling strategy was implemented.

- **Direct Labeling**: Entries from *normal.pcap* and *normal2.pcap* were assigned class "0" (Normal), and entries from *attack.pcap* were assigned class "1" (SYN Flood).
- **Filtered Labeling for Slowloris**: During the analysis of *slowloris.pcap*, it was observed that legitimate traffic from the client PC was recorded simultaneously with the attack traffic. To ensure data integrity, a filtering logic based on source IP addresses was implemented. Traffic originating from the attacker's IP was labeled as class "2" (Slowloris), while concurrent legitimate traffic was correctly reassigned to class "0" (Normal).

This refined labeling process prevented the model from misidentifying legitimate communication as an anomaly, thereby improving the reliability of the classification and reconstruction results discussed in Section 6.

5 Model Evaluation Method

5.1 Accuracy Evaluation Method

In this experiment, we quantitatively evaluated model performance using the following metrics.

- Precision: The proportion of actual attacks among all cases the model classified as "abnormal." This indicates the system's low rate of false positives (false detections).
- Recall: The proportion of all attack packets present in the network that the model correctly detected. This indicates the system's low rate of false negatives (missed detections).
- F1-score: The harmonic mean of Precision and Recall. This metric provides a comprehensive evaluation of the trade-off between detection accuracy and coverage.

The threshold setting for the Autoencoder was determined by calculating

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

across the entire range of reconstruction errors. The threshold was set to the error range corresponding to the highest F1 score achieved.

5.2 Verification of the Learning Process Validity

To verify whether the Autoencoder (AE) has accurately learned the characteristics of normal traffic, the "Reconstruction Error" is utilized as the primary metric. The model is trained to minimize the Mean Squared Error (MSE) between the input x and the output x', where a decreasing loss indicates that the model is successfully internalizing the underlying patterns of the training data. The loss function L, which serves as the criteria for evaluating learning stability, is defined as follows:

$$L(x, x') = \frac{1}{n} \sum_{i=1}^{n} (x_i - x'_i)^2 \tag{4}$$

By monitoring this transition, we can objectively determine if the model has reached a state of convergence, meaning it has sufficiently learned to reconstruct normal communication with high fidelity.

5.2.1 Dual-Perspective Loss Monitoring

To ensure a robust evaluation, we adopted two different monitoring approaches for the training phase, considering the distinct characteristics of the datasets:

- **Batch-wise Iteration Tracking (for NSL-KDD)**: Initially, we attempted to use epoch-averaged plots for all experiments. However, preliminary tests revealed that the NSL-KDD dataset, with its 41 features and vast diversity, exhibited complex learning dynamics that were smoothed out in epoch averages. Therefore, we chose to record the loss for every mini-batch to observe the model's "struggle" with diverse traffic patterns.
- **Epoch-wise Trend Analysis (for GNS3)**: For the custom GNS3 environment, the goal was to verify if the model could reach a stable state within the specific topology. By plotting the average loss per epoch over 500 iterations, we aimed to confirm the steady convergence of the model, filtering out minor statistical noise to focus on the overall learning trend.

6 Results and Considerations in Random Forest

6.1 Results and Analysis Using Normal Traffic

Table 4 shows the classification results of the test data using a model trained on normal traffic, SYN attacks, and Slowloris attacks.

Table 4: Model Evaluation Results (3-class)

Class	Precision	Recall	F1-score	Support
Normal (0)	1.00	1.00	1.00	11
SYN Flood (1)	1.00	1.00	1.00	24,345
Slowloris (2)	1.00	1.00	1.00	445
Accuracy		1.00		24,801
Macro avg	1.00	1.00	1.00	24,801
Weighted avg	1.00	1.00	1.00	24,801

As a result, an accuracy rate of 1.0000 (100%) was achieved, enabling the model to correctly classify all flows contained in the test data as either normal or abnormal without any misclassifications. SYN flood attacks are characterized by a high volume of packets, so we visualized the packet distribution (see Figure 2).

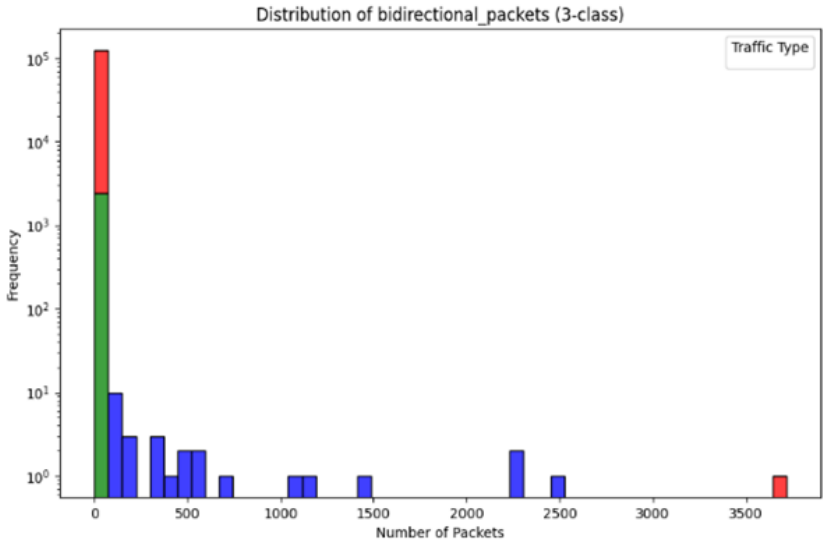

Figure 2: Distribution of Bidirectional Packet Counts

The distribution of bidirectional packet counts shows that SYN flood attack traffic (red) and Slowloris attack traffic (green) are concentrated near zero, while normal web access (blue) consistently involves a certain number of packets (see Figure 2). This reveals the characteristic of SYN flood attacks producing a high packet count.

The Random Forest model is believed to have effectively learned significant differences in prominent features such as this bidirectional packet count, achieving extremely high detection accuracy.

6.2 Results and Considerations Using Complex Normal Traffic

Table 5 shows the classification results of the test data using a model trained on complex normal traffic and two types of attacks: SYN and Slowloris, considering the possibility that normal traffic could be simple and reach 100%.

Table 5: Model Evaluation Results (3-class)

Class	**Precision**	**Recall**	**F1-score**	**Support**
Normal (0)	1.00	1.00	1.00	71
SYN Flood (1)	1.00	1.00	1.00	24,321
Slowloris (2)	1.00	1.00	1.00	472
Accuracy		1.00		24,864
Macro avg	1.00	1.00	1.00	24,864
Weighted avg	1.00	1.00	1.00	24,864

As a result, it achieved an accuracy rate of 1.0000 (100%) even for complex normal traffic, correctly classifying all flows in the test data as either normal or abnormal without misclassification. Slowloris attacks are characterized by prolonged flow durations, so we visualized the distribution of flow durations (see Figure 3). Normal communication was compared using complex normal traffic.

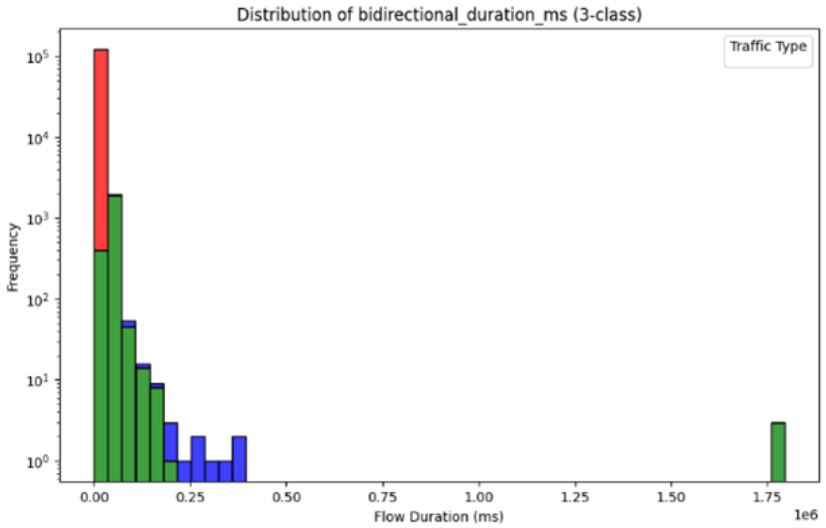

Figure 3: Distribution of Flow Duration

The distribution of flow durations clearly distinguishes SYN flood attack traffic (red) and Slowloris attack traffic (green) from normal web access (blue) (see Figure 3). Furthermore, the right side shows the characteristic longer flow durations of Slowloris attacks.

The random forest model is believed to have effectively learned significant differences in features such as flow duration, achieving extremely high detection accuracy.

7 Results and Discussion in Autoencoders

7.1 Results and Discussion Using the NSL-KDD Dataset

The classification results of the test data, the graph, and the loss transition graph for the model trained solely on the NSL-KDD dataset are shown in Table 6, Figure 4, and Figure 5.

Table 6: AE Model NSL-KDD Classification Report

Class	**Precision**	**Recall**	**F1-score**	**Support**
Normal	0.92	0.81	0.86	9,711
Anomaly	0.87	0.95	0.91	12,833
Accuracy		0.89		22,544
Macro avg	0.89	0.88	0.88	22,544
Weighted avg	0.89	0.89	0.89	22,544

The autoencoder achieved approximately 89% accuracy and maintained a high precision rate (see Table 6).

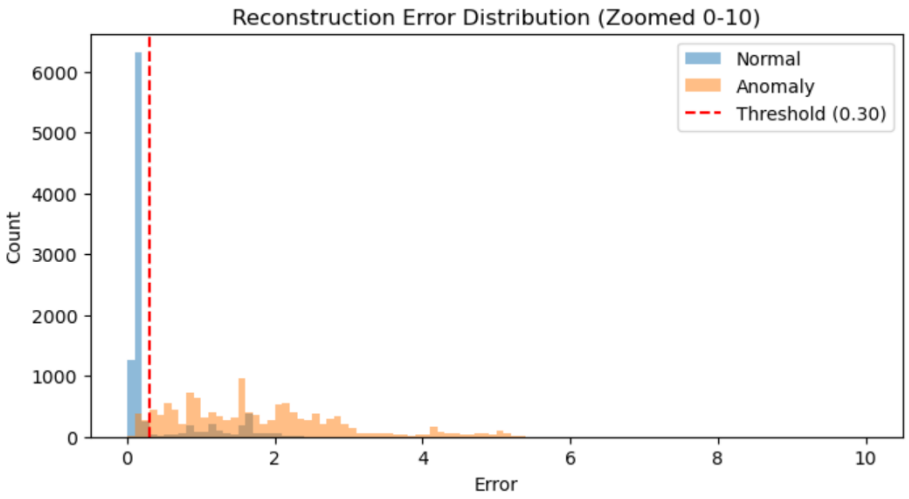

Figure 4: Learning Results Using the NSL-KDD Dataset

The verification of the autoencoder using the NSL-KDD dataset showed low reconstruction error for normal data, confirming from the graph that the model accurately learned the characteristics of normal communications.
For unknown attack traffic, reconstruction error significantly exceeded that of normal traffic, indicating successful identification as abnormal based on the set threshold. (see Figure 4).

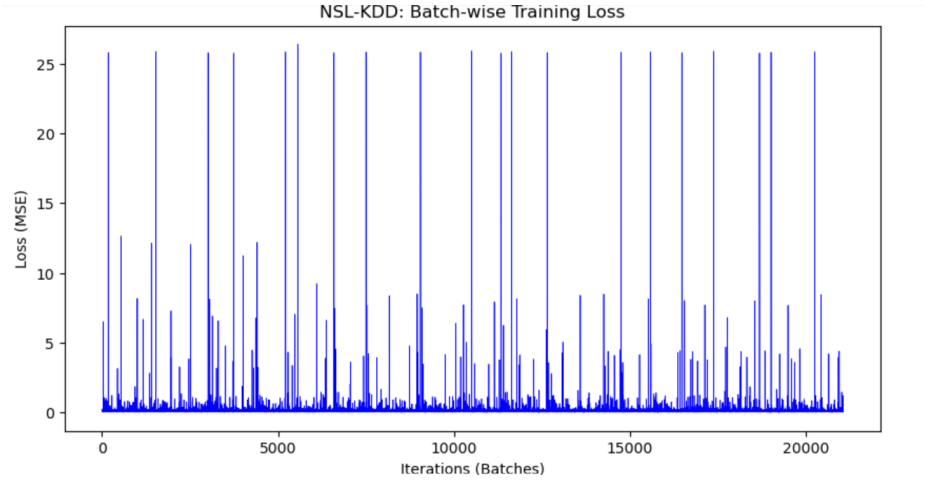

Figure 5: Training Loss Using the NSL-KDD Dataset

Figure 5, Training Loss Using the NSL-KDD Dataset, shows frequent and sharp spikes in loss throughout 20 epochs. Initially, we suspected a failure in the learning process, but upon closer inspection, the baseline loss was steadily decreasing. We interpreted these spikes not as errors, but as evidence that the model was encountering highly diverse and "difficult" normal communication patterns specific to the benchmark data. The fact that the baseline eventually stabilized near zero suggests the model successfully generalized the core characteristics of normal communication.
However, it is essential to clarify that while the NSL-KDD dataset is effective for validating the model's fundamental detection logic, it has significant limitations regarding practical performance evaluation. Since the dataset consists of pre-processed features in a static text format , it bypasses the most computationally intensive process in a real-world IDS: the transformation of raw network packets into flow-based features . Consequently, the processing speeds observed in this section do

not represent the actual system latency in operational environments . A comprehensive evaluation of the system's real-time feasibility using raw traffic and an optimized feature extraction pipeline is detailed in Section 8.

7.2 Results and Analysis Using GNS3 Data

The classification results of the test data using the model trained solely on GNS3 data, along with the graph and loss transition graph, are shown in Table 7, Figure 6, and Figure 7.

Table 7: AE Model GNS3 Classification Report

Class	Precision	Recall	F1-score	Support
Normal	0.95	0.89	0.92	906
Anomaly	0.96	0.98	0.97	2,400
Accuracy		0.96		3,306
Macro avg	0.95	0.94	0.94	3,306
Weighted avg	0.96	0.96	0.96	3,306

The autoencoder achieved approximately 96% accuracy (0.96) and maintained a high precision rate (see Table 7).

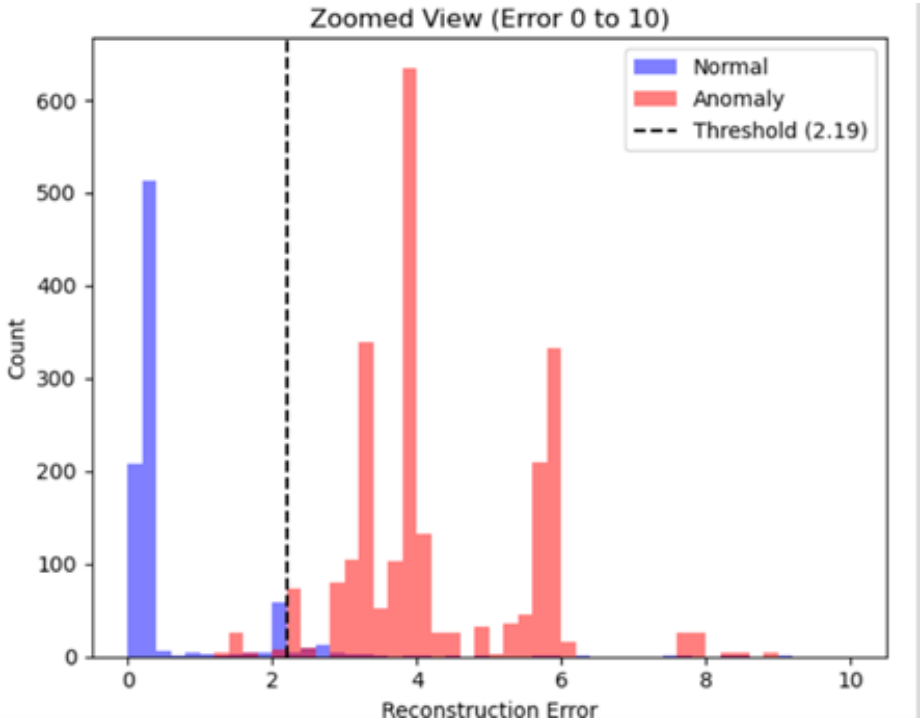

Figure 6: Learning Results Using GNS3 Data

Using GNS3 data to evaluate the autoencoder, the graph confirmed that the reconstruction error for normal data remained low, indicating the model accurately learned the characteristics of normal communication.
The reconstruction errors for normal and abnormal communication clearly diverged in magnitude, suggesting the derived threshold successfully identified anomalies with precision. (see Figure6).

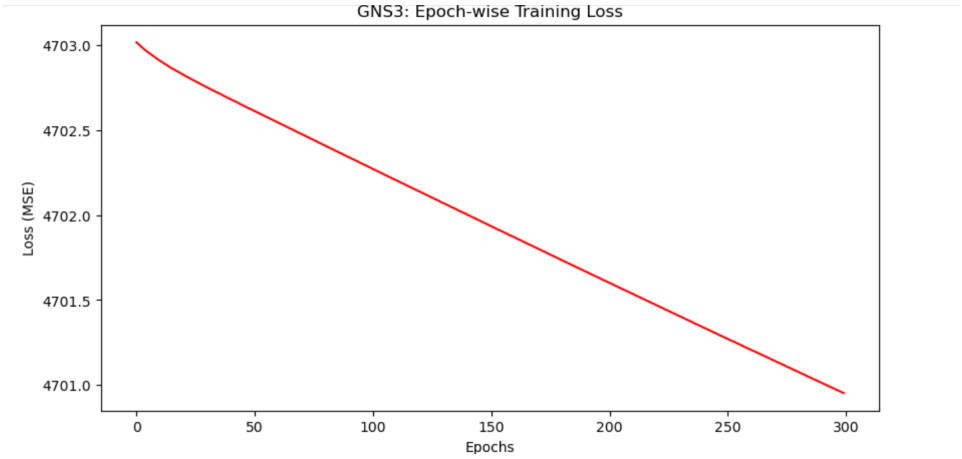

Figure 7: Training Loss Using GNS3 Data

The learning curve using GNS3 data (see Figure 7) shows that the loss function exhibited relatively stable values from the initial stage, with a gradual and consistent decrease observed throughout 300 epochs. This indicates that the features extracted by NFStream represent the network's steady-state with extremely high accuracy, suggesting the model captured the structure of normal communication largely from the early stages of training. As a result, although no steep descent was observed, the MSE converged steadily toward its minimum value. The relatively large absolute MSE values reflect the unnormalized scale of the extracted flow features rather than poor reconstruction, as the anomaly decision relies on the relative gap between normal and abnormal errors shown in Figure 6. This indicates the model sufficiently acquired the reconstruction capability necessary for anomaly detection.

8 Evaluation of Feature Extraction using NFStream

8.1 Optimization and Design Policy

In this study, the NFStream framework was employed for packet analysis to maximize the real-time capability of the DDoS attack detection system. As described by Aouini and Pekar [9], NFStream is designed to avoid bottlenecks by combining a packet observation layer implemented in C with Python through the C Foreign Function Interface (CFFI).

While the standard NFStream configuration can extract up to 88 features, this research optimized the computational cost by selecting 6 key features specifically relevant to DDoS attack characteristics, as detailed in Section 2.3. These features include *duration*, *src_bytes*, *dst_bytes*, *packets*, *byte_per_pkt*, and *asymmetry*. Streamlining the feature set in this manner directly contributes to minimizing processing latency and ensuring real-time performance, as the performance overhead can be significantly reduced by selecting only a subset of discriminators required for a particular use case.

8.2 Limitations of Measuring Processing Speed with the NSL-KDD Dataset

Although the NSL-KDD dataset is widely used for evaluating the accuracy of ML models, it is inadequate as a metric for measuring the practical "speed" of a system for the following reasons:

1. **Pre-extracted Features:** NSL-KDD is a static dataset where features have already been extracted from packets and stored in text formats.

2. **Misleading Metrics:** Measuring processing speed with NSL-KDD only reflects "file reading speed." In a real-world Intrusion Detection System (IDS), the primary computational load stems from analyzing raw packets and calculating flow statistics.
3. **Lack of Real-time Context:** In operational environments, evaluating the "extraction cost" from packet capture to feature transformation is essential.

Accordingly, this study evaluated true real-time performance by measuring the time required to generate flows from raw packets in a GNS3 environment using pcapng files.

8.3 Empirical Performance Results in the GNS3 Environment

The proposed method was executed within a GNS3-based network. Table 8 summarizes the extraction performance for each traffic file.

Table 8: Measured Feature Extraction Performance per File

Evaluation File	Flow Count	Extraction Time (s)	Speed (ms/flow)
normal2.pcapng	399	1.5010	3.7618
attack.pcapng	121,511	23.7022	**0.1951**
slowloris.pcapng	2,407	1.9163	0.7961
Total Sum	**124,317**	**27.1194**	**0.2181**

8.4 Comparison with Previous Research and Verification of Real-time Capability

Table 9 summarizes the qualitative differences between this work and the most relevant prior study by Wang et al. [4], while Table 10 compares the feature-extraction cost with benchmark values reported by Aouini and Pekar [9].

Table 9: Comparison with the most relevant prior work

	Wang et al. [4]	This work
Data source	Static NetFlow CSV	GNS3 raw pcap
End-to-end pipeline	No (CSV only)	Yes
Extraction latency	Not reported	0.2181 ms/flow
Feature count	∼300	6 (DDoS-specific)
Reproducible topology	No	Yes

Table 10: Comparison of Feature Extraction Costs

Extraction Tool / Environment	Speed (ms/flow)	Reference / Remark
Proposed (GNS3)	**0.2181**	**Current Work (6 features)**
NFStream (Aouini et al. [9])	0.6513	*Computer Networks*, 2022
CICFlowMeter	2.6230	Reported in [9]

Aouini et al. [9] demonstrated that NFStream is significantly faster than the widely used CICFlowMeter. By further narrowing the features to 6 items specialized for DDoS detection, this study achieved a speedup of approximately 12 times compared to CICFlowMeter and 3 times compared to the original NFStream benchmark.

The measured value of **0.2181 ms/flow** indicates that even under high DDoS traffic loads, the extraction process does not become a bottleneck, supporting real-time detection and mitigation. It should be noted that this comparison concerns the cost of the feature-extraction stage relative to the NFStream baseline configuration in [9], rather than a head-to-head comparison of end-to-end detection performance against other IDS frameworks.

9 Conclusion and Future Work

9.1 Summary of Research Findings

In this research, the effectiveness of machine learning models for the detection and classification of DDoS attacks (SYN Flood and Slowloris) was verified within an experimental environment constructed on the virtual network simulator GNS3.

9.1.1 Attack Classification and Anomaly Detection

- **Random Forest (RF) Performance**: The RF model using features extracted via NFStream achieved an Accuracy of 1.00, even under conditions including complex normal traffic. This suggests that the model effectively captured distinct statistical features, such as packet counts in SYN floods and flow durations in Slowloris attacks.
- **Autoencoder (AE) for Unknown Threats**: By training solely on normal communication data, the AE successfully identified unknown attack traffic by generating high reconstruction errors. This validates the effectiveness of the AE's ability to detect deviations from normality without relying on specific attack signatures.

9.1.2 Verification via Training Loss Transitions

A significant aspect of this study was the evaluation of learning progress through loss function transitions.

- For the NSL-KDD dataset, batch-wise monitoring revealed a stable baseline amidst stochastic fluctuations, indicating successful generalization across diverse patterns.
- For the GNS3 data, the smooth, gradual decline of the loss curve confirmed that the custom network environment provided high-quality, stable traffic, allowing the model to converge reliably.

9.2 Real-time Processing Performance Verification

A pivotal finding of this research is the quantitative verification of the system's real-time processing capability. By integrating the NFStream-based pipeline and optimizing the feature set to 6 key discriminators, the system achieved an average extraction speed of 0.2181 ms/flow.

The significance of this result is highlighted through a comparative analysis with existing methodologies:

- **High-Speed Extraction:** The recorded latency of 0.2181 ms/flow represents a significant improvement over traditional tools such as CICFlowMeter (approx. 2.62 ms/flow), as reported by Aouini et al. [9]. This proves that the proposed system can handle high-volume traffic without the preprocessing stage becoming a bottleneck.
- **Superiority over Benchmarks:** Even when compared to the standard NFStream configuration (0.65 ms/flow for 88 features [9]), the proposed 6-feature model achieved approximately three times faster performance. This validates that the strategy of streamlining features specifically for DDoS detection is highly effective for reducing computational overhead.
- **Practical Feasibility:** The results demonstrate that the system can process over 4,500 flows per second on commodity hardware. This throughput is sufficient to support real-time inline detection and immediate mitigation in GNS3-simulated network environments, ensuring that anomalies are identified within milliseconds of their occurrence.

In conclusion, the experimental results confirm that the proposed pipeline successfully bridges the gap between high-accuracy anomaly detection and the stringent latency requirements of real-world network security.

9.3 Future Work

Future research will focus on the following areas to further refine the detection performance and practical applicability:

- **Verification with Complex Normal Traffic**: Beyond the steady-state traffic used in this study, we aim to evaluate the model using more dynamic and complex normal traffic profiles, such as concurrent video streaming, large-scale file transfers, and mixed-application environments. This will allow for an assessment of the Autoencoder's robustness and threshold stability in environments with high statistical entropy.
- **Advanced Threat Detection**: Applying the system to sophisticated attacks, such as SQL injection, that are designed to blend into legitimate application-layer traffic.
- **Real-time Deployment**: Transitioning from offline batch analysis to a real-time, inline detection and mitigation system within the GNS3 environment to evaluate processing latency and defensive effectiveness.

References

[1] Andreas C. Muller and Sarah Guido. *Introduction to Machine Learning with Python (in Japanese)*. O'Reilly Japan, 2017. Hideki Nakata (Translator).

[2] Anatoliy Balyk, Mikołaj Karpiński, Artur Naglik, Gulmira Shangytbayeva, and Ihor Romanets. USING GRAPHIC NETWORK SIMULATOR 3 FOR DDOS ATTACKS SIMULATION. *International Journal of Computing*, 16(4):219–225, 2017.

[3] Martin Chovanec, Martin Hasin, Martin Havrilla, and Eva Chovancová. Detection of HTTP DDoS attacks using NFStream and TensorFlow. *Applied Sciences*, 13(11):6671, 2023.

[4] Chao Wang, Yunxiao Sun , Wenting Wang , Hongri Liu and Bailing Wang,*. Hybrid intrusion detection system based on combination of random forest and autoencoder. `https://pdfs.semanticscholar.org/58c5/064658cc8573877df6d9f847331543daa278.pdf`, 2023. (Accessed on 2025-08-27).

[5] GNS3 Technologies Inc. GNS3 — the software that empowers network professionals. `https://www.gns3.com/`, 2025. (Accessed on 2025-08-27).

[6] VMware, Inc. VMware Workstation Pro. `https://www.vmware.com/jp/products/workstation-pro.html`, 2025. (Accessed on 2025-08-27).

[7] Kali Linux. Kali Linux Documentation. `https://www.kali.org/docs/`, 2025. (Accessed on 2025-08-27).

[8] Canonical Ltd. Ubuntu server for scale-out computing. `https://ubuntu.com/server`, 2025. (Accessed on 2025-08-27).

[9] Adrian Pekar , Zied Aouini. Nfstream: A flexible network data analysis framework. `https://www.sciencedirect.com/science/article/pii/S1389128621005739`, 2022. (Accessed on 2026-04-30).