A Hierarchical Hybrid Deep Learning Framework for Unknown Attack Detection in SDN Networks

Francesco Di Gennaro[1,2,*,†], Andrea Giuliani[2,†], Christian Morbidoni[3,†], Alessandro Cucchiarelli[2,†] and Luca Spalazzi[2,†]

[1]*IMT School for Advanced Studies, Lucca, Italy*
[2]*Università Politecnica delle Marche, Ancona, Italy*
[3]*Università degli Studi G. d'Annunzio, Pescara, Italy*

Abstract

Software-Defined Networking (SDN) introduces new security challenges that require detecting both known and previously unseen attacks. This paper presents a novel hierarchical hybrid deep learning framework for SDN intrusion detection, combining a supervised 1D convolutional neural network with an unsupervised MLP autoencoder in a two-stage pipeline. The approach is evaluated on KRONOS-SDN, a realistic SDN-specific intrusion detection dataset generated in a controlled cyber-range. Evaluation is done using a leave-one-out attack strategy to simulate zero-day attacks. Obtained results show improved recall on unseen attacks compared to anomaly detection alone, while maintaining a stable false positive rate on benign traffic.

Keywords

IDS, SDN, ML, ML, Security

1. Introduction

Software-Defined Networking (SDN) has emerged as a key paradigm for modern enterprise network as they enable centralized control, programmability, and flexibility. While these features support dynamic traffic engineering and fine-grained policy enforcement, they also introduce new security challenges, as the centralization of control and the exposure of programmable interfaces increase the attack surface of SDN environments.

In recent years, machine learning and deep learning techniques have been widely adopted to improve Intrusion Detection Systems (IDSs) effectiveness, particularly in flow-based SDN monitoring scenarios. Supervised deep learning models, such as convolutional neural networks (CNNs), have demonstrated strong performance in detecting known attack patterns when sufficient labeled data is available. Nevertheless, their effectiveness is inherently limited when confronted with previously unseen or evolving threats. In this work, we use the term zero-day attack, in an experimental sense, to refer to attack classes that are completely unseen during training, rather than to previously unknown vulnerabilities in operational environments.

Unsupervised and semi-supervised anomaly detection approaches address this limitation by learning models of normal network behavior and flagging deviations as potential attacks. Autoencoders are widely adopted in this context due to their ability to compactly represent benign traffic and expose anomalies through reconstruction errors. However, when used in isolation, anomaly-based systems often exhibit higher false positive rates and lower accuracy on known attacks.

In this work, we propose a hierarchical two-stage deep learning architecture that combines the strengths of both supervised and unsupervised approaches for intrusion detection in SDN environments. The proposed framework operates as a sequential decision pipeline. In the first stage, a supervised one-dimensional CNN is trained on labeled flow-level traffic to detect known attacks with high precision. In

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
†These authors contributed equally.
s1117033@pm.univpm.it (F. Di Gennaro); s1114482@studenti.univpm.it (A. Giuliani); christian.morbidoni@unich.it (C. Morbidoni); a.cucchiarelli@univpm.it (A. Cucchiarelli); l.spalazzi@univpm.it (L. Spalazzi)

the second stage, an unsupervised Multi-Layer Perceptron Autoencoder (MLP-AE), trained exclusively on benign traffic, analyzes only those flows that are classified as normal by the CNN. By computing reconstruction errors and applying a statistically defined threshold, the second stage enables the detection of anomalous behaviors corresponding to previously unseen attacks.

The proposed design reflects a realistic deployment scenario in which known attack signatures can be effectively filtered by a supervised detector, while a lightweight anomaly detection module refines the decision process by capturing deviations from learned normality. This hierarchical structure not only improves overall detection coverage, but also reduces the computational and operational burden of applying anomaly detection indiscriminately to all traffic.

The effectiveness of the proposed approach is evaluated using flow-level data derived from the KRONOS-SDN dataset, a large-scale benchmark generated in a realistic SDN cyber-range environment. To assess the capability of the framework to detect zero-day attacks, we adopt a leave-one-out evaluation strategy, where each attack type is excluded from supervised training and treated as a zero-day during testing. Experimental results show that the proposed two-stage architecture achieves high recall across a wide range of attack types, significantly improving the detection of zero-day attacks that are missed by the supervised classifier alone, while maintaining a stable false positive rate on benign traffic.

2. Related works

The centralized control and programmability of SDN enable fine-grained traffic engineering and policy enforcement, but also reshape the threat model by concentrating logic in the controller and exposing programmable interfaces differently from traditional networks. Yoon *et al.* analyse systematically security threat that SDN architecture introduces with the separation of the application, control and data planes, with the introduction of vulnerabilities that may be exploited to degrade availability or compromise control policy endorsed through the controller[1]. These SDN-specific risks motivate the advancement in security monitoring solutions.

Supervised deep learning remains a dominant approach when labeled SDN traffic is available, as it can learn discriminative patterns for known attack classes directly from flow-level features. The InSDN dataset was introduced to support SDN-specific IDS research and is widely used as a benchmark for deep models [2]. Building on this line, ElSayed *et al.* propose a CNN-based IDS and explicitly discuss overfitting as a key factor limiting robustness, especially under zero-day conditions, proposing a regularization-oriented hybrid design to improve generalization on SDN traffic [3]. More recent comparative studies confirm that CNN-based architectures, often combined with temporal modules such as LSTM or transformer stages, can reach high performance on SDN flow records; for instance, Ataa *et al.* compare CNN-LSTM and transformer models on InSDN-derived settings, emphasizing the benefits of deep learning for SDN intrusion classification [4]. In addition, hybrid CNN-LSTM detectors have been explored as supervised classifiers for SDN flows, leveraging CNNs for local feature interactions and recurrent units for temporal dependencies [5].

Despite these advances, a key limitation of purely supervised IDS is its dependence on labeled examples of all relevant threats. This assumption is often unrealistic in operational conditions where attacks evolve or appear as previously unseen variants and where the test-time distribution can differ substantially from training [6]. This limitation motivates complementary anomaly-detection mechanisms that target unknown behaviors.

For the mentioned reason, hybrid IDS designs aim to combine the high precision of supervised classifiers on known attacks with the broader coverage of semi-supervised and unsupervised mechanisms for unknown behaviors.

A representative example of sequential hybridization is Deep ResNIDS, which explicitly combines supervised and unsupervised models to improve robustness against previously unseen threats[7]. The framework is motivated by the complementary failure modes of the two paradigms: supervised classifiers achieve excellent performance on in-distribution attacks but often miss out-of-distribution (OOD) or zero-day samples, whereas autoencoder-based anomaly detectors capture deviations from

benign behavior at the cost of increased false alarms. Deep ResNIDS concretizes this intuition working on packet-level via a staged pipeline in which an anomaly detector is applied only to traffic classified as benign by the first-stage classifier, thereby targeting false negatives and extending coverage to zero-day and OOD patterns. The anomaly decision is ultimately obtained from reconstruction errors through a statistically defined threshold.

The mentioned paper was related to traditional network, but the concern is even more relevant in SDN, where certain attacks may show evolving traffic footprints and where controller and switches introduce SDN-specific characteristics.

Khamaiseh *et al.* address the problem of detecting both known and unknown DDoS saturation attacks in SDN by integrating supervised and semi-supervised classifiers, showing that hybridization can improve resilience when test-time attacks are not represented in training data [8].

Hybrid detection has been investigated also by combining lightweight statistical indicators with learned models to improve effectiveness and operational practicality. Long and Wang [9] proposed a mechanism for DDoS detection and mitigation in SDN that integrates entropy-based measurements with a stacked sparse autoencoder and an SVM classifier (SSAE-SVM), with the purpose of capturing abnormal traffic dynamics while maintaining a structured decision process suitable for SDN deployment.

Ultimately, recent SDN-native frameworks explore hybridization between deep learning and classical supervised learners through a deep autoencoder combined with a Random Forest classifier. The mentioned pipeline has been proposed for intrusion detection in a native SDN setting, explicitly targeting improved performance and practical deployment considerations [10].

Building upon these foundations, this paper adopts a hierarchical decision pipeline that separates *known-attack recognition* from *unknown-attack discovery* and explicitly targets flow-level evidence derived from the switching layer.

2.1. Datasets

In recent years, many datasets have been proposed to support evaluation of IDSs, however most of them present many limitations, in particular with respect to network realism, temporal resolution and heterogeneity of client devices [11].

Early SDN-specific IDS datasets such as NSJ [12] and ZCA [13] introduced DDoS and scanning traffic in emulated SDN environments, but were limited by simplified topologies, restricted feature sets and poor reproducibility. Subsequent efforts, including InSDN [14] and NCLP [15], expanded attack coverage and scale, yet still lacked device heterogeneity, fine-grained temporal features and system-level statistics. More recent datasets (e.g., Y-NV-RP-DJ-M-C [16], AAHHBA [17], ASMK [18], HLD-DDoSDN [19]) explored physical testbeds or richer features, but remain constrained by narrow attack scopes, homogeneous devices and limited time awareness. In contrast, traditional IDS benchmarks such as CICIDS2017 [20], UNSW-NB15 [21], CSE-CIC-IDS2018 [22] and LSPR23 [23] offer high realism, heterogeneous traffic and temporal continuity, yet are not SDN-based and lack controller-level visibility, leaving a gap for comprehensive, time-aware and heterogeneous SDN intrusion datasets.

Recently, the KRONOS-SDN dataset, which we adopt in our experiments, was designed to address several of the mentioned limitations. Specifically, it includes heterogeneous client devices, including IoT nodes and embedded systems, within a realistic SDN environment, relying on a sophisticated architecture that reflects operational networks and captures high-resolution, time-referenced traffic metrics.

3. Methods and tools

3.1. Proposed Model

The proposed architecture combines a supervised convolutional neural network (CNN) with an unsupervised multi-layer perceptron autoencoder (MLP-AE) in a sequential decision pipeline.

The framework operates in two stages. In the first stage, a CNN-based classifier is trained on labeled SDN flow-level traffic to identify known attack patterns. In the second stage, an autoencoder trained exclusively on benign traffic analyzes only the samples classified as normal by the CNN, enabling the detection of previously unseen (zero-day) attacks.

During inference, each network flow is initially processed by the CNN. Flows classified as malicious are immediately flagged as attacks. Otherwise, the flow is forwarded to the autoencoder, which computes a reconstruction error. Samples whose reconstruction error exceeds a predefined threshold are classified as anomalous and flagged as zero-day attacks. The entire pipeline flow, along with details on the used neural networks structure, is illustrated in Figure 1.

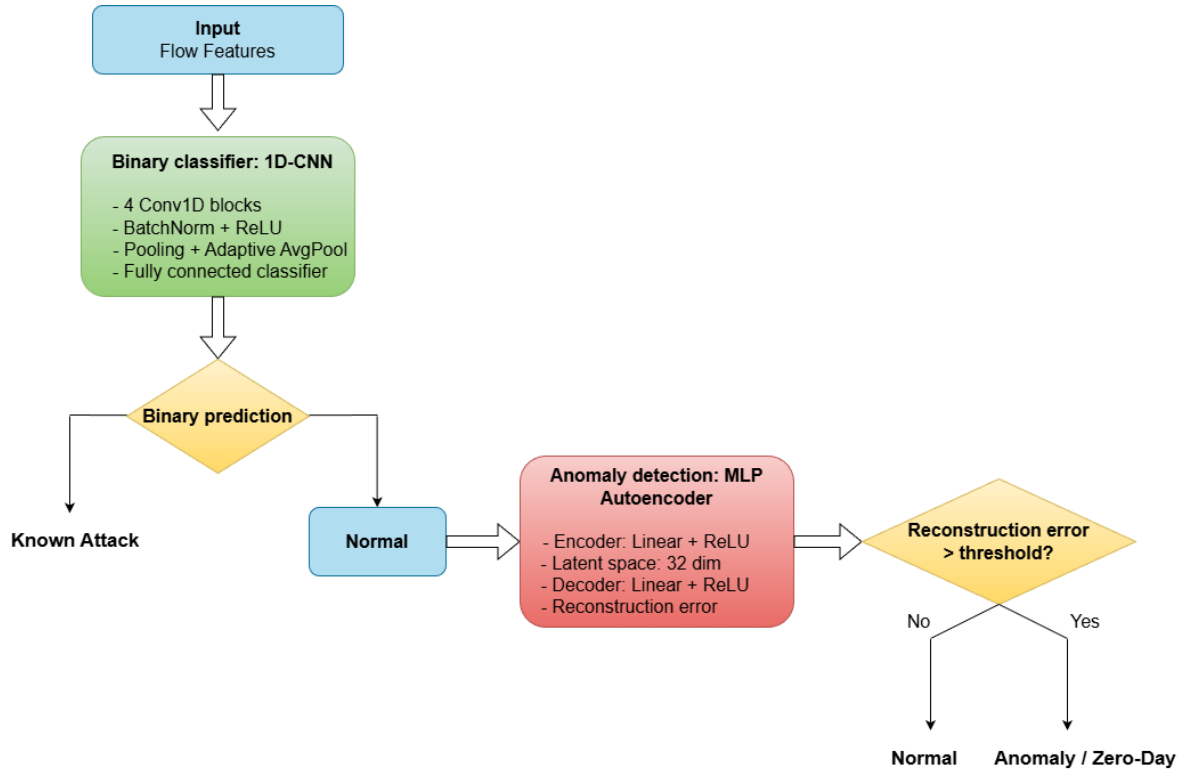

Figure 1: Pipeline flow diagram

3.2. Testbed Architecture

In this work we rely on the flow-level CSV exports derived from KRONOS-SDN, a large-scale Software-Defined Networking (SDN) intrusion detection benchmark dataset originally generated in a controlled cyber-range environment. KRONOS-SDN is derived from a testbed designed to reproduce the functional and architectural complexity of a medium-sized enterprise network. The network environment is orchestrated by an ONOS controller [1] that manages nine Open vSwitch (OVS) [2] instances via Openflow protocol [24] while a Linux kernel router provides layer 3 forwarding among different logically segmented subnets.

The network architecture is shown in the Figure 2 [25].

The logical network topology is organized into five functional areas, as detailed in Table 1.

The main novelties of the KRONOS-SDN dataset lie in its realistic SDN-based enterprise testbed, its temporal and behavioral richness. It is, to our knowledge, the largest publicly released SDN-specific IDS dataset, covering a wide diversity of attack vectors, devices, and services at a scale suitable for

[1]The documentation is available at: https://opennetworking.org/onos/.
[2]The documentation is available at: https://www.openvswitch.org/.

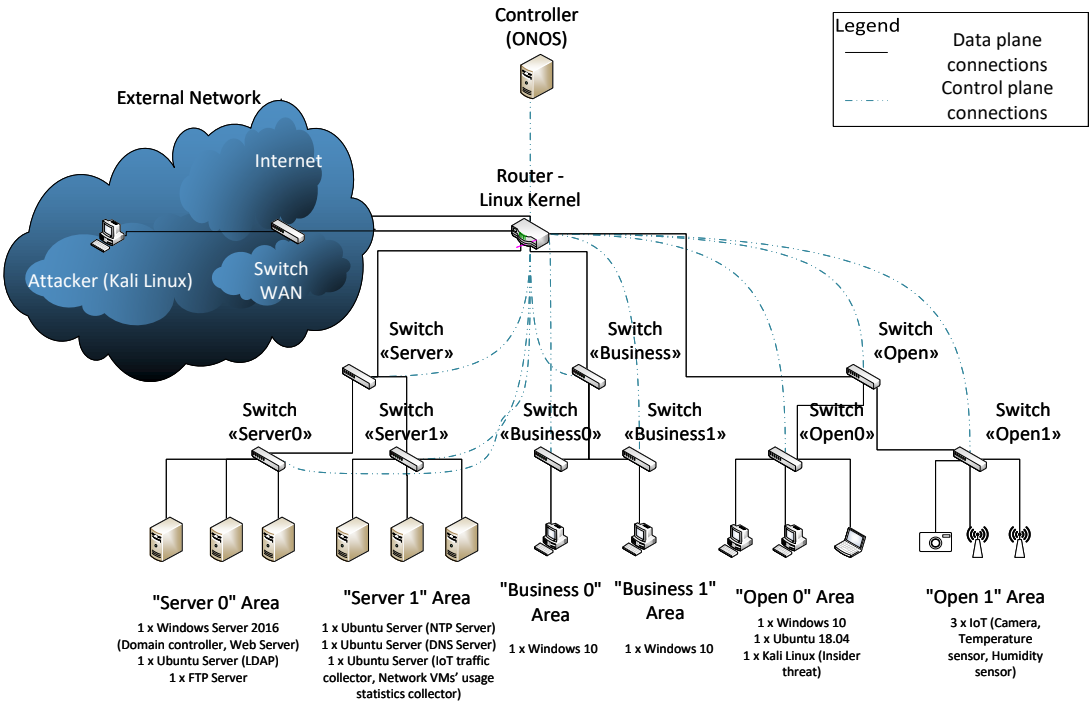

Figure 2: Dataset network architecture.

Area	Description
Server Area	Windows Server domain controller; web and FTP servers; DNS and NTP services; IoT collector for telemetry and usage statistics.
Business Area	Two segments with domain-joined Windows clients executing typical corporate workflows (internal web, FTP, DNS, LDAP).
Open Area	Two less-trusted zones with heterogeneous hosts and IoT devices; clients have Internet access, IoT nodes send telemetry to the IoT collector.
Controller Segment	Segment hosting the ONOS controller, managing all internal Open vSwitch (OVS) instances and enforcing SDN control-plane policies.
External Network (WAN)	Untrusted WAN-facing network emulating Internet connectivity and external attack sources, outside the SDN controller's direct management.

Table 1
Logical segmentation of the KRONOS-SDN testbed.

modern machine learning benchmarking and reproducible evaluation. This is the first study evaluating Deep Learning based IDS on this dataset.

3.2.1. Benign traffic generation

Benign traffic is generated through scripted activities whose intensity follows diurnal enterprise usage patterns. Business clients' activities include DNS, HTTP, FTP and Active Directory requests against the business servers, while the hosts of the "Open Area" produce traffic interfacing with Internet Servers,

that is to say generic application traffic in conjunction with MQTT-based telemetry coming from the simulated sensors towards the IoT Collector server.

3.2.2. Attack plan

Malicious traffic follows a week-long kill-chain schedule that progressively escalates from reconnaissance to disruptive campaigns, e.g. DDoS and DoS attacks. Two Kali Linux nodes, one placed in the Open Area to emulate internal threat and the second in the WAN Area, act as attackers. The attack plan includes host and service discovery, SSH and HTTP brute-force, control-plane resources exhaustion through OpenFlow packet crafting and flow-table flooding, volumetric DoS and DDoS (ICMP, SYN and UDP floods) and application-layer attacks such as Slowloris and LDAP/DNS amplification against critical services and the controller. These campaigns are overlapped with continuous benign activity, producing temporally coherent traces where attack phases emerge against a realistic background.

Aggregated Feature Group	Features
Flow-level statistics	Flow Duration, Flow Bytes/s, Flow Packets/s
Inter-arrival time statistics	Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min, Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min
Packet length statistics	Packet Length Min, Packet Length Max, Packet Length Mean, Packet Length Std, Packet Length Variance, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std
Traffic activity patterns	Active Mean, Active Std, Active Max, Active Min, Idle Mean, Idle Std, Idle Max, Idle Min
Directionality and rates	Down/Up Ratio, Fwd Packets/s, Bwd Packets/s, Bwd Bulk Rate Avg
Packet and byte counters	Total Fwd Packets, Total Bwd Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Act Data Pkts
Header and protocol metadata	Fwd Header Length, Bwd Header Length
TCP control flags	ACK Flag Count, SYN Flag Count, FIN Flag Count, RST Flag Count, PSH Flag Count, Fwd PSH Flags, ECE Flag Count, CWR Flag Count
Segment, window and subflow features	Fwd Seg Size Min, Avg Fwd Segment Size, Avg Bwd Segment Size, Average Packet Size, Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Bytes, Fwd Init Win Bytes, Bwd Init Win Bytes
Bulk transfer features	Bwd Bytes/Bulk Avg, Bwd Packet/Bulk Avg

Table 2
Full list of features in the KRONOS-SDN dataset

3.2.3. Traffic Collection

Network traffic is captured as PCAPs on all network virtual machine in fixed four-hour windows, consisting of 42 capture files per host over the week campaign. Next, the PCAPs are converted into flow-based CSV files using CICFlowMeter [26], which extracts more than 80 features per bidirectional flow, while a separate labeling pre-processing step assigns class labels by correlating timestamps and IP addresses with the scheduled attack timetable. The full list of extracted features is reported in Table 2 For the purpose of this paper, we use the post-processed flow records collected in the CSV files containing both the features of network flows and the label describing the ground-truth traffic classes.

3.3. Data Preprocessing

The experiments are conducted on a flow-based network traffic subset of the original dataset leveraging the files in CSV format, where each record corresponds to a network flow described by a set of numerical

features and an associated attack label.

In particular, for the experiments of this paper, the traffic collected on the ethernet interface of the virtual switch "Open" is used. The CSV file was extracted through the use of CICFlowMeter taking in input the file .pcap of the entire week of experiments and traffic collection.

The reason why we chose the traffic collected from switch Open, is its comprehensive visibility over heterogeneous traffic patterns. In the KRONOS-SDN testbed, the Open segment represents a less-trusted and Internet-facing area where benign user activities (e.g., web browsing, email-like application flows, and IoT telemetry) coexist with a wide range of attack campaigns originating both from internal and external adversary nodes.

As a consequence, flows observed at this switching point naturally include:

- lateral and north-south communications;
- high-volume volumetric attacks and low-rate application-layer attacks;
- mixed protocol interactions that resemble realistic enterprise conditions.

From an operational standpoint, this choice also reflects a practical IDS deployment model where monitoring is performed at the data plane rather than exclusively at the controller.

Table 3 reports the number of flow samples for each traffic class in the subset of KRONOS-SDN used in our experiments. The subset contains 129791 normal flows and all flows corresponding to the different attack campaigns observed at the Open segment.

Traffic Class	**Number of Samples**
Normal	129791
Full Port Scan	590213
DNS Amplification	382241
Final Reconnaissance	147536
DDoS Controller + Switch	131072
DoS HTTP	131072
HTTP Brute Force	40371
SSH Brute Force	34829
Flow Table Flooding	11827
DoS OpenFlow	11456
Advanced Nmap Recon	1201
Nmap OS Fingerprinting	1017
LDAP Amplification	349
Nmap Host Discovery	258
Slowloris HTTP	200
Packet Crafting OpenFlow	136
Brute Force SSH & Web	111
Smurf Attack	1

Table 3
Number of flow samples per traffic class in the experimental subset of KRONOS-SDN.

As regards the preprocessing phase of the input CSV file mentioned, some features are removed to prevent the model from learning spurious correlations that do not generalize beyond the training environment. In particular, flow identifiers, source and destination IP addresses, port numbers, timestamps, and protocol fields are excluded. Additionally, features exhibiting zero variance across the dataset are discarded, as they do not provide discriminative information. Specifically, a consistent set of six features (Bwd PSH Flags, Bwd URG Flags, Fwd Bulk Rate Avg, Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, and Subflow Bwd Packets) was discarded in every training iteration of the CNN model, due to zero variance.

The original multi-class labeling is preserved through the attribute *OriginalLabel* to enable a fine-grained, attack-wise evaluation of the proposed approach. While the learning task is formulated as a binary classification problem (Normal vs. Attack), retaining the original labels allows the assessment of detection performance separately for each attack type during the experimental analysis. For the purpose

of intrusion detection, a binary classification scheme is adopted, where *Normal* traffic is mapped to class 0 and all attack types are mapped to class 1.

3.3.1. Dataset Balancing

Network intrusion datasets are typically characterized by severe class imbalance, both between normal and malicious traffic and among different attack types. To mitigate this issue, a proportional undersampling strategy with minority-class protection is employed.

First, a maximum cap is applied to each attack type to prevent highly represented attacks from dominating the learning process. Subsequently, a proportional undersampling strategy is adopted, where the number of samples allocated to each attack class is determined based on the available amount of normal traffic. For training, approximately 100,000 Normal samples are selected, while the remaining Normal flows are reserved for testing.

To construct a balanced training dataset, an iterative undersampling procedure is applied to the attack classes with the goal of reaching the same number of Malicious samples as Normal ones (i.e., up to 100,000). Initially, a threshold T is computed as the number of Normal training samples divided by the number of unique attack classes:

$$T = \left\lfloor \frac{N_{\text{Normal}}}{C_{\text{attack}}} \right\rfloor$$

All attack classes with fewer than T samples are considered *minority classes* and are fully included, while *majority classes* are undersampled to match the remaining quota. This calculation is repeated iteratively: after each step, the quota is updated, and the procedure is reapplied to the remaining attack classes until the maximum possible number of Malicious samples is reached. As a result, the final training dataset contains equal numbers of Normal and Malicious samples, ensuring that minority attack types are fully represented while majority classes are fairly subsampled. This approach produces a training dataset that is roughly balanced 50/50, allowing the model to learn effectively across all attack types.

3.3.2. Feature Scaling

All numerical features are normalized using Min-Max scaling, mapping values into the $[0, 1]$ range. Formally, for a feature x, the normalized value x' is computed as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

where $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values of the feature in the training set. The scaler is fitted exclusively on the training data and stored for later use during the testing phase, ensuring consistency between training and evaluation.

3.4. Stage 1: CNN-Based Known Attack Detector

The first stage of the proposed intrusion detection pipeline is implemented as a one-dimensional Convolutional Neural Network (1D-CNN) designed to discriminate between normal traffic and known attack patterns. Each input sample is reshaped into a tensor of shape $(1, F)$, where F denotes the number of selected features, enabling the model to interpret the feature vector as a one-dimensional signal. After the preprocessing phase, for every iteration, 70 features were retained for model training, as 6 features exhibiting zero variance were consistently removed across all leave-one-attack-out iterations.

The CNN architecture consists of four convolutional blocks, each composed of a 1D convolutional layer followed by batch normalization, ReLU activation, and pooling operations. To ensure a fixed-size latent representation regardless of the input dimensionality, an adaptive average pooling layer is employed. The extracted features are then processed by a fully connected classifier with dropout regularization to mitigate overfitting. Table 4 summarizes the detailed configuration of each convolutional block.

Block	Filters	Kernel	Stride	Padding	Pooling Type	Pool Size
1	32	3	1	1	MaxPool1D	2
2	64	3	1	1	MaxPool1D	2
3	128	3	1	1	MaxPool1D	2
4	256	3	1	1	Adaptive Avg	1

Table 4
CNN architecture details.

This model acts as a supervised filter for known attacks, forwarding only samples classified as normal traffic to the subsequent anomaly detection stage.

3.4.1. Training CNN

Since the dataset is balanced during preprocessing, standard cross-entropy loss is employed without additional class weighting.

Training is performed using the Adam optimizer with an initial learning rate of 10^{-3}. To improve convergence and stability, a learning rate scheduler dynamically reduces the learning rate when the macro F1-score plateaus. Gradient clipping is applied to prevent exploding gradients, and early stopping is employed to halt training when no further improvement in macro F1-score is observed for a fixed number of epochs.

Upon completion of training, the trained CNN parameters, the feature scaler, and the list of selected features are stored to ensure reproducibility and to enable independent evaluation on unseen data.

3.5. Stage 2: MLP Autoencoder-Based Anomaly Detector

The second stage of the pipeline is designed to detect anomalies among flow samples classified as normal by the CNN-based detector (first stage) and is implemented as a Multi-Layer Perceptron Autoencoder (MLP AE) trained exclusively on normal traffic samples. By learning a compact latent representation of normal behavior, the model reconstructs normal inputs with low error while producing higher reconstruction errors for anomalous inputs.

Input features are first normalized to the [0,1] range using MinMax scaling. The MLP Autoencoder consists of an encoder that maps the input feature vector of dimensionality F=70 into a latent space of size $L = 32$, followed by a decoder that reconstructs the original input. Both encoder and decoder use fully connected layers with ReLU activations, as summarized in Table 5. No internal normalization (e.g., BatchNorm) or dropout is applied. The model is trained using the mean squared error (MSE) loss.

During inference, the reconstruction error for each input is computed as the MSE between the original and reconstructed features. A threshold is determined from the training set, corresponding to the 95th percentile of reconstruction errors on normal samples. Inputs classified as normal by the CNN stage that exceed this threshold are flagged as anomalies.

Module	Layer	Output Width
Encoder	Linear (Input → 128) + ReLU	128
	Linear (128 → 32) + ReLU	32
Decoder	Linear (32 → 128) + ReLU	128
	Linear (128 → Input)	$F = 70$

Table 5
MLP Autoencoder architecture details.

3.5.1. Training MLP AE

The autoencoder is trained in a fully unsupervised manner using only normal traffic. Mean squared error loss is employed, and the Adam optimizer is used with a learning rate of 10^{-3}. Training is performed for a fixed number of epochs ($E = 30$), with mini-batch stochastic gradient descent. The reconstruction threshold is set post-training based on the distribution of reconstruction errors observed on the training set.

After training, the parameters of the MLP Autoencoder, the feature scaler, and the list of selected features are saved to ensure reproducibility and to enable evaluation of the full two-stage pipeline on unseen data. Additionally, the reconstruction error threshold is stored for use during anomaly detection.

4. Experiment results and discussion

As previously mentioned, the CNN model of the proposed pipeline is trained following a leave-one-out attack strategy. Specifically, for each attack reported in Table 6, the corresponding attack samples are entirely excluded from the training set and used only during the testing phase. This evaluation setting allows us to assess the generalization capability of the supervised classifier when confronted with previously unseen attacks and to analyze the effectiveness of the subsequent anomaly detection stage in recovering such samples. Each sample is first processed by the CNN classifier. If the CNN predicts an attack class, the sample is counted as a *Known Attack*. Instead, if the CNN classifies the sample as normal, it is subsequently analyzed by the MLP autoencoder. Samples flagged as anomalous by the autoencoder are counted as *Zero-day Attacks*, while those reconstructed within the normality threshold are counted as *Normal*. Under the adopted leave-one-out attack evaluation, zero-day detections correspond to attack samples that are not recognized by the supervised classifier and are instead identified by the anomaly detection module. Recall is computed considering both known and zero-day detections as true positives.

Attack	Number of samples	Known attack detections	Zero-day detections	Normal detections	Recall
Full Port Scan	590213	590074	0	139	99.98%
DNS Amplification	382241	325523	4	56714	85.16%
Final Reconnaissance	147536	129137	18389	10	99.99%
DDoS Controller + Switch	131072	117918	13148	6	99.99%
DoS HTTP	131072	84479	46593	0	100%
HTTP Brute Force	40371	40371	0	0	100%
SSH Brute Force	34829	1993	181	32655	6.24%
Flow Table Flooding	11827	11823	4	0	100%
DoS OpenFlow	11456	0	11433	23	99.80%
Advanced Nmap Recon	1201	1009	4	188	84.35%
Nmap OS Fingerprinting	1017	1012	0	5	99.51%
LDAP Amplification	349	100	245	4	98.85%
Nmap Host Discovery	258	257	0	1	99.99%
Slowloris HTTP	200	0	200	0	100%
Packet Crafting OpenFlow	136	16	120	0	100%
Brute Force SSH & Web	111	107	1	3	97.30%
Smurf Attack	1	1	0	0	100%

Table 6
Leave-One-Out-Attack pipeline results

Regarding normal (benign) traffic, the proposed pipeline exhibits a stable false positive rate of approximately 5% across all experimental runs. This behavior is mainly due to the anomaly detection threshold of the MLP AE, set at the 95th percentile of the reconstruction error distribution computed

on the training set. A notable exception is represented by the *SSH Brute Force* attack, which achieves a recall of only 6.24%. This behavior can be attributed to the strong similarity between the traffic patterns generated by this attack and those of benign flows. As a consequence, most samples are classified as normal by the CNN and are not flagged as anomalous by the autoencoder. Regarding other attacks, an in-depth analysis of results highlights several attack scenarios in which the CNN alone is not able to achieve satisfactory detection performance. In particular, for attacks such as *DoS OpenFlow*, *Slowloris HTTP*, *DoS HTTP*, *LDAP Amplification*, and *Packet Crafting OpenFlow*, a significant portion of malicious samples is initially classified as normal by the CNN. However, these samples are successfully identified as anomalous by the MLP autoencoder, which substantially improves the overall recall. This behavior empirically demonstrates the effectiveness of the proposed hierarchical architecture, where the anomaly detection stage refines and complements the supervised classifier by capturing previously unseen attack patterns.

4.1. Standalone MLP Autoencoder

To better understand the contribution of the anomaly detection stage, we evaluated the MLP autoencoder on the test set independently of the CNN. Table 7 reports the detection performance of the autoencoder for each attack class and for normal traffic. Results indicate that attacks with highly distinctive patterns, such as *DoS HTTP*, *Slowloris HTTP*, *DDoS Controller + Switch*, *Flow Table Flooding*, and *Smurf Attack*, are almost fully detected, with recall values near 100%. Conversely, attacks exhibiting traffic characteristics similar to normal flows, including *SSH Brute Force*, *DNS Amplification*, *HTTP Brute Force*, and *Advanced Nmap Recon*, show significantly lower recall, ranging from 1% to 7%. This confirms the known limitation of autoencoders when attack patterns are subtle and largely indistinguishable from benign behavior.

Attack	Number of samples	Detected as Anomaly	Recall	False Positive Rate
Full Port Scan	590213	212	0.04%	–
DNS Amplification	382241	26349	6.90%	–
Final Reconnaissance	147536	65102	44.13%	–
DDoS Controller + Switch	131072	131068	99.99%	–
DoS HTTP	131072	131072	100%	–
HTTP Brute Force	40371	1935	4.80%	–
SSH Brute Force	34829	598	1.72%	–
Flow Table Flooding	11827	11800	99.77%	–
DoS OpenFlow	11456	11433	99.80%	–
Advanced Nmap Recon	1201	17	1.42%	–
Nmap OS Fingerprinting	1017	14	1.38%	–
LDAP Amplification	349	338	96.85%	–
Nmap Host Discovery	258	72	27.91%	–
Slowloris HTTP	200	200	100%	–
Packet Crafting OpenFlow	136	120	88.24%	–
Brute Force SSH & Web	111	97	87.39%	–
Smurf Attack	1	1	100%	–
Normal	29791	1483	–	4.97%

Table 7
MLP Autoencoder detection performance on individual attack classes

For the sake of a quick comparison, in Table 8 we report the recall obtained with the leave-one-out pipeline using the MLP-AE based anomaly detection and the proposed two-stage approach. It is evident that the hierarchical architecture effectively mitigates these limitations. Samples initially misclassified by the CNN as normal are often recovered by the autoencoder, substantially improving the overall recall for previously unseen attacks, as expected.

Attack	MLP-AE	CNN+MLP-AE
Full Port Scan	0.04%	99.98%
DNS Amplification	6.90%	85.16%
Final Reconnaissance	44.13%	99.99%
DDoS Controller + Switch	99.99%	99.99%
DoS HTTP	100%	100%
HTTP Brute Force	4.80%	100%
SSH Brute Force	1.72%	6.24%
Flow Table Flooding	99.77%	100%
DoS OpenFlow	99.80%	99.80%
Advanced Nmap Recon	1.42%	84.35%
Nmap OS Fingerprinting	1.38%	99.51%
LDAP Amplification	96.85%	98.85%
Nmap Host Discovery	27.91%	99.99%
Slowloris HTTP	100%	100%
Packet Crafting OpenFlow	88.24%	100%
Brute Force SSH & Web	87.39%	97.30%
Smurf Attack	100%	100%

Table 8
Comparison of recall obtained with MLP AE based anomaly detection and with the proposed two-stage approach.

5. Conclusion and future works

This work proposed a hierarchical hybrid deep learning framework for intrusion detection in SDN environments, combining a supervised CNN with an unsupervised autoencoder to effectively detect zero-day attacks. Results of of our experiments on the KRONOS-SDN dataset suggest the effectiveness of the proposed framework in enhancing robustness in realistic SDN scenarios. In the future we plan to consolidate the experiments reported in this paper by taking into account other nodes in the network. We also aim to extend the generalization analysis by conducting leave-one-family-out experiments, where entire families of attacks (e.g., DDoS variants, port-scanning campaigns, and other related classes) are removed from the training set and used exclusively for testing. Future research directions include investigating adaptive or data-driven dynamic thresholding mechanisms to improve robustness under changing traffic conditions and concept drift.

Declaration on Generative AI

During the preparation of this work the authors used GPT in order to review grammar and paraphrasing. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

[1] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, G. Gu, Flow wars: Systemizing the attack surface and defenses in software-defined networks, IEEE/ACM Transactions on Networking (2017). Accepted Aug. 15, 2017.

[2] M. S. ElSayed, N.-A. Le-Khac, A. Jurcut, InSDN: A novel SDN intrusion dataset, IEEE Access (2020). doi:10.1109/ACCESS.2020.3022633.

[3] M. S. ElSayed, N.-A. Le-Khac, A. Jurcut, A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique, Journal of Network and Computer Applications (2021). doi:10.1016/j.jnca.2021.103124.

[4] M. Ataa, et al., Comparative study of intrusion detection in software defined networks: CNN-LSTM and transformer with InSDN dataset, Scientific Reports (2024). doi:10.1038/s41598-024-79001-1.

[5] M. Abdallah, N.-A. Le-Khac, A. Jahromi, A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs, in: ACM International Conference on Availability, Reliability and Security (ARES), 2021. doi:10.1145/3465481.3469190.

[6] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE Symposium on Security and Privacy, 2010. doi:10.1109/SP.2010.25.

[7] S. Hore, J. Ghadermazi, A. Shah, N. D. Bastian, A sequential deep learning framework for a robust and resilient network intrusion detection system, Computers and Security 144 (2024) 103928. URL: https://www.sciencedirect.com/science/article/pii/S0167404824002311. doi:https://doi.org/10.1016/j.cose.2024.103928.

[8] S. Khamaiseh, A. Al-Alaj, M. Adnan, H. W. Alomari, The robustness of detecting known and unknown DDoS saturation attacks in SDN via the integration of supervised and semi-supervised classifiers, Future Internet (2022). doi:10.3390/fi14060164.

[9] Z. Long, W. Jinsong, A hybrid method of entropy and ssae-svm based ddos detection and mitigation mechanism in sdn, Computers and Security 115 (2022) 102604. doi:10.1016/j.cose.2022.102604.

[10] L. Mhamdi, et al., Securing SDN: Hybrid autoencoder-random forest for intrusion detection in a native SDN environment, Journal of Network and Computer Applications (2024). doi:10.1016/j.jnca.2024.103868.

[11] F. D. Gennaro, A. Cucchiarelli, C. Morbidoni, L. Spalazzi, A comparative analysis of datasets for intrusion detection in software-defined networks, in: 2025 Joint National Conference on Cybersecurity, ITASEC and SERICS 2025, 2025.

[12] Q. Niyaz, W. Sun, A. Y. Javaid, A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN), EAI Endorsed Transactions on Security and Safety 4 (2017). doi:10.4108/eai.28-12-2017.153515.

[13] C. B. Zerbini, L. F. Carvalho, T. Abrão, M. L. Proença, Wavelet against random forest for anomaly mitigation in software-defined networking, Applied Soft Computing 80 (2019) 138–153. URL: https://www.sciencedirect.com/science/article/pii/S1568494619301115. doi:https://doi.org/10.1016/j.asoc.2019.02.046.

[14] M. S. Elsayed, N.-A. Le-Khac, A. D. Jurcut, InSDN: A Novel SDN Intrusion Dataset, IEEE Access 8 (2020) 165263–165284. doi:10.1109/ACCESS.2020.3022633.

[15] M. P. Novaes, L. F. Carvalho, J. Lloret, M. L. Proença, Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment, IEEE Access 8 (2020) 83765–83781. doi:10.1109/ACCESS.2020.2992044.

[16] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Perez-Diaz, E. Jacob, C. Martinez-Cagnazzo, Physical Assessment of an SDN-Based Security Framework for DDoS Attack Mitigation: Introducing the SDN-SlowRate-DDoS Dataset, IEEE Access 11 (2023) 46820–46831. doi:10.1109/ACCESS.2023.3274577.

[17] M. A. Aladaileh, M. Anbar, A. J. Hintaw, I. H. Hasbullah, A. A. Bahashwan, S. Al-Sarawi, Renyi Joint Entropy-Based Dynamic Threshold Approach to Detect DDoS Attacks against SDN Controller with Various Traffic Rates, Applied Sciences 12 (2022). URL: https://www.mdpi.com/2076-3417/12/

12/6127. doi:10.3390/app12126127.

[18] N. Ahuja, G. Singal, D. Mukhopadhyay, N. Kumar, Automated DDOS attack detection in software defined networking, Journal of Network and Computer Applications 187 (2021) 103108. URL: https://www.sciencedirect.com/science/article/pii/S1084804521001296. doi:https://doi.org/10.1016/j.jnca.2021.103108.

[19] A. A. Bahashwan, M. Anbar, S. Manickam, G. Issa, M. A. Aladaileh, B. A. Alabsi, S. D. A. Rihan, HLD-DDoSDN: High and low-rates dataset-based DDoS attacks against SDN, PLOS ONE 19 (2024) 1–29. URL: https://doi.org/10.1371/journal.pone.0297548. doi:10.1371/journal.pone.0297548.

[20] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic profile for binary classification, in: Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), SciTePress, 2017, pp. 108–116. URL: https://doi.org/10.5220/0006097701030111. doi:10.5220/0006097701030111.

[21] N. Moustafa, J. Slay, Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, 2015, pp. 1–6. URL: https://doi.org/10.1109/MilCIS.2015.7348942. doi:10.1109/MilCIS.2015.7348942.

[22] I. Sharafaldin, A. Habibi Lashkari, A. A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, in: Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, INSTICC, SciTePress, 2018, pp. 108–116. doi:10.5220/0006639801080116.

[23] A. Dijk, E. Halisdemir, C. Melella, A. Schu, M. Pihelgas, R. Meier, Lspr23: A novel ids dataset from the largest live-fire cybersecurity exercise, Journal of Information Security and Applications 85 (2024) 103847. URL: https://www.sciencedirect.com/science/article/pii/S2214212624001492. doi:https://doi.org/10.1016/j.jisa.2024.103847.

[24] K. Nisar, E. R. Jimson, M. H. A. Hijazi, I. Welch, R. Hassan, A. H. M. Aman, A. H. Sodhro, S. Pirbhulal, S. Khan, A survey on the architecture, application, and security of software defined networking: Challenges and open issues, Internet of Things 12 (2020) 100289. URL: https://www.sciencedirect.com/science/article/pii/S2542660520301219. doi:https://doi.org/10.1016/j.iot.2020.100289.

[25] F. Di Gennaro, A. Cucchiarelli, C. Morbidoni, L. Spalazzi, KRONOS-SDN: A Large-Scale, Cross-Plane Dataset for Machine Learning Intrusion Detection in Software-Defined Networks, 2025. URL: https://ssrn.com/abstract=5971166. doi:10.2139/ssrn.5971166, SSRN Paper No. 5971166.

[26] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, A. A. Ghorbani, Characterization of encrypted and VPN traffic using time-related features, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), INSTICC, SciTePress, 2016, pp. 407–414. doi:10.5220/0005740704070414.