electronics

Article

Learn-IDS: Bridging Gaps between Datasets and Learning-Based Network Intrusion Detection

Minxiao Wang [1], Ning Yang [2,*], Yanhui Guo [3] and Ning Weng [1,*]

[1] The Computer Engineering Program in the School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University, Carbondale, IL 62901, USA; minxiao.wang@siu.edu
[2] The Information Technology Program in the School of Computing, Southern Illinois University, Carbondale, IL 62901, USA
[3] Department of Computer Science, University of Illinois, Springfield, IL 62703, USA; yguo56@uis.edu
* Correspondence: nyang@siu.edu (N.Y.); nweng@siu.edu (N.W.)

Abstract: In an era marked by the escalating architectural complexity of the Internet, network intrusion detection stands as a pivotal element in cybersecurity. This paper introduces Learn-IDS, an innovative framework crafted to bridge existing gaps between datasets and the training process within deep learning (DL) models for Network Intrusion Detection Systems (NIDS). To elevate conventional DL-based NIDS methods, which are frequently challenged by the evolving cyber threat landscape and exhibit limited generalizability across various environments, Learn-IDS works as a potent and adaptable platform and effectively tackles the challenges associated with datasets used in deep learning model training. Learn-IDS takes advantage of the raw data to address three challenges of existing published datasets, which are (1) the provided tabular format is not suitable for the diversity of DL models; (2) the fixed traffic instances are not suitable for the dynamic network scenarios; (3) the isolated published datasets cannot meet the cross-dataset requirement of DL-based NIDS studies. The data processing results illustrate that the proposed framework can correctly process and label the raw data with an average of 90% accuracy across three published datasets. To demonstrate how to use Learn-IDS for a DL-based NIDS study, we present two simple case studies. The case study on cross-dataset sampling function reports an average of 30.3% OOD accuracy improvement. The case study on data formatting function shows that introducing temporal information can enhance the detection accuracy by 4.1%.The experimental results illustrate that the proposed framework, through the synergistic fusion of datasets and DL models, not only enhances detection precision but also dynamically adapts to emerging threats within complex scenarios.

Keywords: network intrusion detection; datasets engineering; deep learning; feature fusion

check for updates

Citation: Wang, M.; Yang, N.; Guo, Y.; Weng, N. Learn-IDS: Bridging Gaps between Datasets and Learning-Based Network Intrusion Detection. *Electronics* **2024**, *13*, 1072. https://doi.org/10.3390/electronics13061072

Academic Editors: Abdussalam Elhanashi and Pierpaolo Dini

Received: 7 February 2024
Revised: 6 March 2024
Accepted: 10 March 2024
Published: 14 March 2024

1. Introduction

Network Intrusion Detection Systems (NIDS) are crucial components in ensuring the security and integrity of computer networks. By continuously monitoring and analyzing network traffic for suspicious or malicious activities, NIDSs help detect and prevent unauthorized access and attacks. NIDSs play a vital role in safeguarding sensitive data [1], protecting against cyber threats, and maintaining the confidentiality, integrity, and availability of network resources. With the ever-increasing sophistication of cyber attacks and the proliferation of network-connected devices, the importance of NIDS in identifying and mitigating security risks cannot be overstated. NIDSs provide organizations with early detection capabilities, allowing them to respond promptly to potential threats, minimize damage, and maintain the trust and reliability of their network infrastructure.

Deep Learning (DL) has been increasingly applied to NIDS due to its ability to automatically learn complex patterns and representations from data [2,3]. In DL-based NIDS methods, neural networks are used for mining the inherent patterns and characteristics within network traffic flow to distinguish malicious traffic from benign traffic. Compared with traditional data mining and machine learning-based NIDS, DL-based methods hold the potential to revolutionize intrusion detection in aspects such as streamlining the feature engineering process, reducing the manual effort required for designing handcrafted features, automatically extracting hierarchical representations, and adapting dynamically to new and unseen threats.

As a data-driven method [4], DL-based NIDSs require large amounts of labeled data to learn meaningful patterns and representations. Standardized datasets also allow researchers and practitioners to compare the effectiveness of different algorithms, architectures, and techniques on a common ground. Although many large-scale public NIDS datasets exist, we have observed many gaps between existing NIDS datasets and the DL-based NIDS studies.

The first gap is the mismatch between the provided tabular format and the diversity of DL models. In order to apply different types of neural networks, researchers have to design and implement a pre-processing module, which converts the raw traffic flows into samples with a particular data format [5–7]. Even for the provided tabular format, researchers also need to use the pre-processing module if they want to use some extra features. Using various data formats can introduce the missing information of tabular data, for example, the dynamic trends of traffic flow or the raw malicious bytes. Therefore, the public datasets need to be present in different formats.

The second gap is the mismatch between the fixed data instances (traffic flow) and the dynamic network scenarios. Although DL-based NIDSs achieved many successes, employing DL in NIDSs still has many limitations, open questions, and the unique challenges of [8,9]. For example, the dynamic network environments challenge the capability of DL-based NIDS to detect intrusion behaviors early and in real-time [10]. However, the fixed data samples cannot be used to study a dynamic case. Meanwhile, adversarial attacks challenge the robustness of DL-based NIDS [11], and researchers need to perturb the behavior of network traffic. Therefore, each data sample needs an interface to be modified.

The third gap is the mismatch between the isolated datasets and the cross-dataset requirement of DL-based NIDS studies. Given the various NIDS application scenarios, DL-based NIDS should be functional for different environments. The generalization of DL-based NIDS against distribution shifts is one of the most urgent problems [12]. The isolated datasets, which include unique handcraft features and different attack types, should be usable for the same DL-based NIDS model to study the generalization problem. Therefore, different datasets need to be uniformly processed for cross-dataset usage.

In order to bridge all the gaps at the same time and solve other challenges of DL-based IDS in Figure 1, this paper presents a novel tool, called Learn-IDS. Learn-IDS provides a comprehensive, efficient, and extensible framework for preparing customized NIDS data for diverse DL-based study purposes. The structure of Learn-IDS is shown in Figure 2. This framework can efficiently process the raw packet capture (PCAP) traces with a C language-based parser. Both the raw PCAP and tabular data are utilized to construct and label traffic flow for multiple public datasets correctly. By using the raw data, Learn-IDS can uniformly process and store multiple datasets together in a network traffic data bank to solve the third isolated dataset gap. The data sampling module of Learn-IDS supports variable data sampling functions. Hence, the stored data can be further selected and processed based on users' requirements to solve the second gap. The data formatting module provides flexible network traffic representation formats, such as time series, raw bytes arrays, and graphs, for solving the first gap between tabular data and the diversity of DL models.

The main contribution of this paper is as follows:

- We developed a comprehensive, efficient, and extensible tool (Learn-IDS) to prepare customized data for DL-based NIDS studies.
- Learn-IDS can uniformly process the raw PCAP from different datasets, sampling data samples based on research purposes, and generating different traffic representation formats to meet the DL model input requirements.

- We show that Learn-IDS can correctly process the raw data from multiple datasets and present two case studies, in which Learn-IDS plays an important role. The source code is available at https://github.com/wangminxiao/Learn_IDS.git (accessed on 9 March 2024).

The rest of this paper is presented as follows, in Section 2, we introduce the backgrounds of the NIDS dataset and DL-based NIDSs. In Section 3, we focus on analyzing the related work of DL-based IDS to show the gaps and our motivations. In Section 4, we design the details of our Learn-IDS framework, which includes four components: Raw Data Pre-Processing, Data Customizing, DL-based IDS Models, and Training and Evaluation. In Section 5, we evaluate the effect of Learn-IDS and its impact on DL-based NIDS. Section 6 is the conclusion.

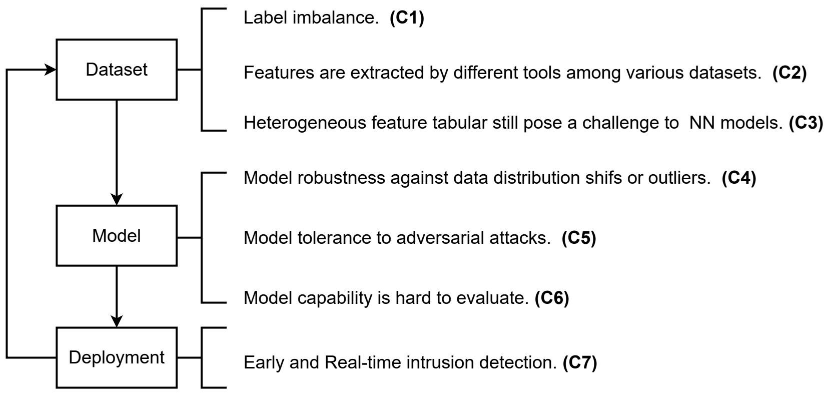

Dataset
Model
Deployment
Label imbalance. (C1)
Features are extracted by different tools among various datasets. (C2)
Heterogeneous feature tabular still pose a challenge to NN models. (C3)
Model robustness against data distribution shifs or outliers. (C4)
Model tolerance to adversarial attacks. (C5)
Model capability is hard to evaluate. (C6)
Early and Real-time intrusion detection. (C7)

Figure 1. The challenges within the three aspects of DL-based NIDS.

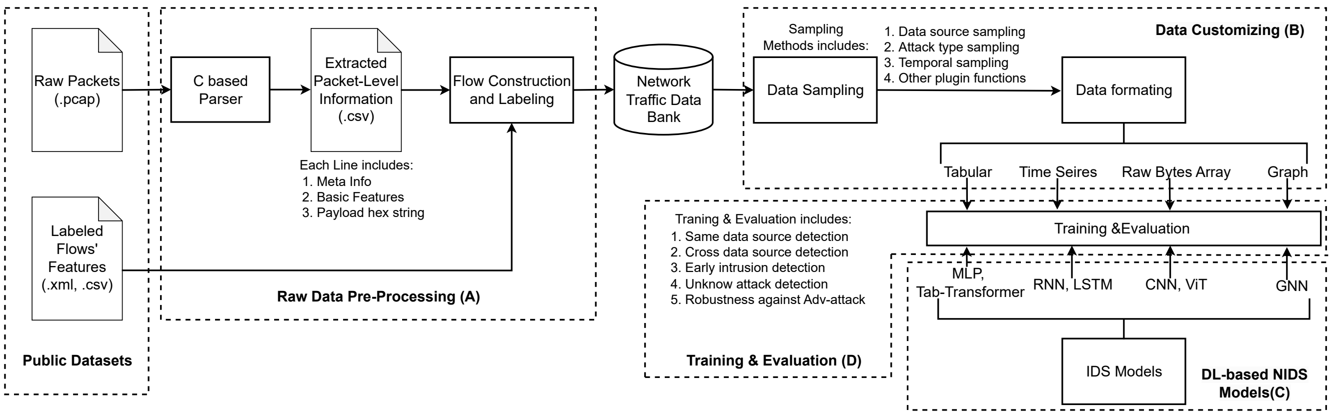

Raw Packets (.pcap)
Labeled Flows' Features (.xml, .csv)
Public Datasets
C based Parser
Extracted Packet-Level Information (.csv)
Each Line includes:
1. Meta Info
2. Basic Features
3. Payload hex string
Flow Construction and Labeling
Raw Data Pre-Processing (A)
Network Traffic Data Bank
Sampling Methods includes:
1. Data source sampling
2. Attack type sampling
3. Temporal sampling
4. Other plugin functions
Data Customizing (B)
Data Sampling
Data formating
Tabular
Time Seires
Raw Bytes Array
Graph
Traning & Evaluation includes:
1. Same data source detection
2. Cross data source detection
3. Early intrusion detection
4. Unknow attack detection
5. Robustness against Adv-attack
Training & Evaluation (D)
Training &Evaluation
MLP, Tab-Transformer
RNN, LSTM
CNN, ViT
GNN
IDS Models
DL-based NIDS Models(C)

Figure 2. Overview of the framework of Learn-IDS.

2. Background

2.1. NIDS Datsets

The NIDS datasets are crucial for data-driven intrusion detection methods. These datasets serve as the foundation for training, validating, and testing NIDS models, allowing researchers and practitioners to develop and assess the effectiveness of various techniques. The examples of the most popular datasets are introduced as follows, the UNSW-NB15 intrusion detection system dataset [13] encompasses source files in various formats, including PCAP, BRO, and CSV. The dataset was generated from an IXIA traffic generator. Within this data-generating testbed, two servers were linked to a router equipped with a TCP dump, interfacing with three clients and producing PCAP files. Simultaneously, the third server was connected to a distinct router, interacting with its own set of three clients. CIC-IDS2017 and CSE-CIC-IDS2018 datasets are presented by the Canadian Institute of Cybersecurity (CIC) and Communications Security Establishment (CSE) [14]. The CIC17 dataset consists of PCAP and CSV data and the CIC18 dataset consists of PCAP, system log, and CSV data. The two datasets both use a B-profile system to generate network behaviors, such as HTTP, HTTPS, FTP, SSH, and email protocols. The attack traffic is generated by different attack tools within Kali and Windows systems. The difference is CIC17 dataset is collected from a small testbed environment including 2 Web servers, 16 Public, 6 Ubuntu servers, 5 Windows servers, and a MAC server but the CIC18 dataset is collected from an AWS environment, the network topology includes an attack network of 50 machines, five departments holding 100 machines each and a server with 30 machines.

2.2. Deep Learning-Based NIDS Models and Formats

In recent years, Deep Learning (DL) has demonstrated significant success and maintained its influence across diverse fields, including computer vision (CV) [15], natural language processing (NLP) [16], and medical science [17]. Using different types of deep learning models for different types of data is a common and effective strategy in deep learning. Diverse data formats may require specialized architectures to extract meaningful patterns and features. Unlike other areas, network traffic data are uniquely shaped by human design, exhibiting full deformability into various formats such as tabular, time series, raw bytes array, or graphs. This distinct characteristic introduces a noteworthy aspect of DL-based NIDS.

For using different formats and DL-based NIDS models, KitNET [5] employs a collection of neural networks known as autoencoders to collaboratively distinguish between regular and anomalous traffic patterns. He et al. [18] present MS-DHPN, an LSTM model with the feature embedding technique IDS. This model uses feature embedding to combine categorical features with sequential information from network traffic data captured by the LSTM. Wang et al. [19] design HAST-IDS, a CNN-based model for raw byte array format. Another one is LUCID which is presented in [20], the CNN model uses a time window-based feature matrix for observing network traffic behavior. For the GNN-based model, E-GraphSAGE NIDS [21], a GNN approach that allows capturing both edge features of a graph as well as the topological information for network intrusion detection in IoT networks. Hu et al. [22] propose a graph embedding method to model the packet interactions in network traffic for NIDS.

3. Related Work and Motivations

Three fundamental aspects for constructing a deep learning-based IDS are as follows: (1) Dataset: Involves IDS data preparation, encompassing data generation/collection, annotation, and preprocessing. (2) Model: Encompasses the design, training, and evaluation of a DL-based IDS model. (3) Deployment: involves the integration of the trained model with the entire system for monitoring streaming network data.

3.1. Dataset Challenges

Commonly, existing datasets exhibit imbalances not only between benign and malicious traffic but also notably among different attack types. For example, in the CIC-IDS-2017 dataset [14], the “BENIGN” class has 2,359,087 flow records (83.34%), the majority of malicious classes is “DoS Hulk”, with 231,072 flow records (8.16%), and the minority malicious class “Heartbleed” only has eight flow records (0.00039%). Training an IDS model on an imbalanced dataset will directly impact its performance, particularly in multi-class classification tasks (**C1** in Figure 1).

Meanwhile, prevalent published datasets are typically presented in the form of tabular data, where network traffic flows serve as instances. Each instance is accompanied by a label and multiple features, providing the necessary information for training IDS models. For instance, the UNSW-NB15 dataset [13] has 47 features and the CIC-IDS-2017 dataset has 80 features, while only six features overlap between them. One reason for the difference is these two datasets use different network traffic analysis tools. UNSW-NB15 uses the matched features, which have been extracted by both the Argus tool and Bro-IDS tool. CIC-IDS-2017 used their own CICFlowMeter tool, whose lower version was known as the

ISCXFlowMeter which was used by the ISCX2012 dataset. The isolation of feature sets makes it challenging to train a model simultaneously on multiple datasets, hindering the potential to address the data distribution problem associated with a single dataset (**C2** in Figure 1).

Additionally, the most commonly used tabular data form of features poses challenges to convolutional neural networks (CNN) [23], even being called the last "unconquered castle" [24]. Because tabular data features for IDS are heterogeneous data with different attribute types, value scales, and a mix of discrete and continuous variables, there is often a lack of strong correlation among features (different from language data) and no spatial dependencies (different from image data). Many proven powerful methods based on CNN cannot be adopted for modeling this data form (**C3** in Figure 1).

3.2. Model Challenges

Enhancing the robustness of IDS models to accommodate data distribution shifts is crucial. In the realm of intrusion detection, models trained on curated datasets may prove unreliable and inefficient when deployed in real-network scenarios. Certain classical machine learning-based methods, such as Decision Trees, exhibit considerable strength in handling heterogeneous tabular data. However, they often face a significant limitation in terms of robustness when confronted with shifts in data distribution [25]. Given the inevitability of data distribution shifts in real-world deployments spanning both cross-time [10,26] and cross-domains [12], it becomes imperative to include robustness verification in the model assessment process (**C4** in Figure 1).

Adversarial attacks pose a significant threat to DL-based IDS models [27]. A network adversarial attack involves the intentional modification of a network attack to induce mispredictions in the IDS model while preserving the original impact of the network attack. This underscores the importance of fortifying IDS models against such adversarial manipulations to ensure their reliability and effectiveness in detecting network threats (**C5** in Figure 1).

The primary goal of DL-based IDS models, as well as other applications was to outperform the state-of-the-art on a single large-scale dataset [28]. However, outperforming on an IDS dataset has become challenging in the current landscape, given the multitude of works showcasing flawless performance across various datasets. This widespread success makes it increasingly difficult to assess the true potential capabilities of IDS models, crucial for fostering realistic deployments [29], as their evaluation becomes intricate on the majority of existing datasets (**C6** in Figure 1).

3.3. Deployment Challenges

The implementation of DL-based IDS in real-network deployments is still in its nascent stages [28]. The prompt detection of intrusions is paramount for the effective deployment of DL-based IDS. This capability allows for the timely execution of accurate counter-attack responses, preventing potential damage before it occurs [30]. Prompt detection entails discerning the nature of network traffic as benign or malicious in the initial stages of traffic flows. However, achieving early intrusion detection presents challenges, primarily stemming from the limited information available in the form of raw packets [31]. Many existing Deep Learning-based NIDS methods depend on handcrafted features as input, and the restricted nature of raw packets can potentially impact the quality of these input features (**C7** in Figure 1).

4. Framework

In this section, we will introduce the structure and modules of Learn-IDS. Learn-IDS is a general framework for DL-based IDS study. Its whole framework is shown in Figure 2, the whole pipeline consists of four components: (A) Raw Data Pre-Processing; (B) Data Customizing; (C) DL-based IDS Models; (D) Training and Evaluation.

4.1. Raw Data Pre-Processing

The raw data pre-processing component serves three main objectives: (1) the efficient extraction of raw information from PCAP files; (2) accurate labeling of traffic flow and (3) the storage of processed data for swift retrieval during both training and testing phases. To achieve these goals, we have developed modules such as parser and flow construction and labeling.

4.1.1. Parser Module

The parser module constructs traffic flow based on traffic termination and a customized timeout threshold; it labels traffic flow using public tabular data and independently stores all the extracted information for each traffic flow. This approach facilitates quick loading of data by the PyTorch Dataloader (in PyTorch, a DataLoader is a utility that helps efficiently load and iterate over a dataset during the training or evaluation of a machine learning model). The specific details of each module are elaborated below: This module is a specialized C program designed for parsing and processing PCAP files, enabling the effective utilization of extensive raw PCAP data from public datasets. Leveraging the PCAP library, the program meticulously reads and analyzes packets, extracting vital information such as IP addresses, ports, protocols, timestamps, and payload data. The parsed data are thoughtfully organized and exported to a CSV file, laying the groundwork for further analysis or visualization.

Key features of this program include the inclusion of header files for networking and PCAP functionalities, a robust packetHandler function for detailed packet processing, and thorough data extraction covering attributes like IP details, ports, timestamps, and payload. Notably, the code implements payload truncation for TCP and UDP packets, limiting it to a maximum length of 8000 bytes in hexadecimal format within the CSV output.

In addition to facilitating detailed packet analysis, the code maintains statistics by tracking the occurrences of TCP, UDP, ICMP, and other packet types during parsing. Furthermore, the program's offline processing capability makes it a valuable tool for historical network traffic data analysis, contributing to comprehensive retrospective examinations.

4.1.2. Flow Construction and Labeling Module

The flow construction module further processes the raw packet information in the CSV output of the parser module with the pandas library to construct and label network traffic flow. As shown in the (A) part of Figure 2, the construction module also takes the tabular data from datasets as inputs to group the packets into the unit of flow. The flow construction module uses seven fields (source IP, destination IP, source port, destination port, protocol, start timestamp, duration) as flow IDs to match packets to labeled flows in tabular data. An alternative method for determining the start timestamp and duration involves following public tabular data. However, this approach proves to be inefficient as it requires filtering the output CSV files N times, with N being the line number of the public tabular data. Moreover, it is worth noting that some errors have been identified in certain public tabular datasets [32,33] recently, further complicating the efficiency and reliability of this method.

Therefore, the process of flow construction plays a crucial role in determining the termination of a network flow, relying on both connection closure and session timeout considerations. In the context of connection-oriented protocols like TCP, the conclusion of a network flow often occurs through mutual agreement or upon the completion of the communication task. This can transpire when one entity sends a TCP FIN (finish) packet, and the other entity acknowledges it with an ACK packet, signaling the conclusion of the connection. Alternatively, network flows may terminate due to session timeout, which takes place when there is no activity or communication between entities for a specified duration, prompting the network infrastructure or application layer to close the connection.

Once all packets of a network flow are appropriately grouped, the flow is labeled, and the information in the output CSV file is extracted and saved into a TXT file named after the unique flow ID. All flow file paths and their respective labels are recorded in the network traffic data bank, and stored as a JSON file that preserves a Python dictionary structure for subsequent use. The data bank dictionary structure comprises three levels of keys: "dataset name", "attack type name", and "flow data path".

4.2. Data Customizing

In the network traffic data bank, all raw information of network flows from different public datasets is well organized and stored for further NIDS studies. The data customizing component includes a data sampling module and a data formatting module to prepare training and testing sets for different NIDS studies.

4.2.1. Data Sampling

The preparation of datasets for DL-based NIDS can vary based on the specific focus and requirements of the study. Therefore, we provide data sampling functions to select the needed data samples to ensure that the model is trained, validated, and tested on relevant and representative data.

Data Source and Attack Types Sampling: The data source and attack types sampling module provides an interface for users to obtain specific types of network flow from single or multiple datasets. Sampling instances from the Data Bank requires two elements: dataset name and class label. For a given class, sampling from different datasets can bring more information about how a type of traffic behaves in different environments. The information can help DL-based IDS models learn more general patterns to distinguish this type of instance from others. Meanwhile, in the original datasets, some classes of instances only occupy a small percentage. The imbalance class distribution harms IDS models' training effect. Instead of using an over-sampling method [34], a cross-dataset sample mitigates the imbalance problem by combining the same class from different datasets.

Temporal sampling for early detection: To simulate and study the early intrusion detection situation, we use the temporal sampling function to re-generate the labeled flow. We defined a concept of sampling window size, which is the number of the first of several packets. The generated flows are cropped out by the sampling window, and the parts in the sampling window will be regarded as the early state of flows, based on which tabulars or other formats are built for early intrusion detection.

4.2.2. Data Formatting

The network flow is the actual instance for a flow-based IDS model to "observe". However, there is no particular ML model that can directly process raw packets. Hence, the raw information in packets needs further processing to transform into a format that can be fed into the ML model, but what kind of format to use is still open. As shown in the (B) part of Figure 2, the data formatting module includes four typical data formats: (1) Tabular; (2) Time Series; (3) Raw Array and (4) Graph.

Tabular: The tabular format is the most common data format, which is normally provided by existing datasets. However, tabular contains a mix of discrete and continuous variables as well as various attribute types and value scales. Those characters challenge most neural network models [23]. Meanwhile, existing datasets normally use unique features that barely overlap. The impact of features for IDS receives a lot of attention, Binbusayyis et al. [35] presented an optimal features study to enhance IDS performance; Adhao et al. [36] argued that feature engineering is essential for DL-based IDS. Therefore, we provided API for users to define and extract customized feature sets for further IDS features to study.

Time Series: The time series format introduces temporal information that can show the behavior of malicious traffic better. The FTime Series is generated by dynamically extracting features when packets arrive. Hence, the sequence has the same length as the number of packets belonging to a flow. This type of data form is more suitable for recurrent neural networks (RNN) to learn the temporal pattern. For example, He et al. [18] proposed MS-DHPN, a combination of AE and LSTM-based models, to detect time seriess.

Raw Bytes Array: The raw bytes array format is proposed for taking advantage of powerful CNN and Vision Transformer (ViT). One benefit is this data format does not require extracting handcrafted features. Since the length of a packet is not fixed, its first N bytes are used to represent it as a whole. The headers of a packet can be fully included upon taking the appropriate value N, as well as some or all of the payload data. N is set to be a square number to make processing more convenient [7].

Graph: Many current graph-based NIDS transform network traffic flows into dynamic IP address graphs, serving as a representation of network topology. However, in the context of the data formatting module, the term "formats" pertains to the depiction of an individual traffic flow. The graph format provided in this context captures diverse correlations within dynamic feature series. Notably, graph formats and graph neural networks excel in explicitly modeling inter-temporal and inter-variable relationships, a task that proves challenging for traditional methods and other deep neural network-based approaches [37].

4.3. DL-Based NIDS Models

To accelerate the NIDS model building progress, Learn-IDS contains implementations of IDS methods with different basic neural network blocks: (1) MLP, (2) RNN, (3) CNN, (4) Transformer, and (5) GNN. We choose those models as representative models to perform DL-based NIDSs. In this phase, we only implement one or two methods for each block type to test the function of the toolbox, instead of duplicating more existing methods. Hence, we duplicated four existing IDS methods, which correspond to four different data formats mentioned in the last subsection. The following Table 1 lists the supported DL-based NIDS models.

Table 1. Supported DL-based NIDS models.

Formats	Basic Block	Models
Tabular	MLP	KitNET [5]
	Transformer	FT-Transformer [38]
Time Series	LSTM	MS-DHPN [18]
	Transformer	Informer [39]
Raw Byte Array	CNN	HAST-IDS [19]
	CNN	LUCID [20]
	Transformer	ViT [40]
Graph	GNN	E-GraphSAGE [21]

4.4. Training and Evaluation

In Learn-IDS, we design a unified Pytorch workflow pipeline for training and evaluating NIDS with different datasets, different data formats, and different IDS models so that studying different NIDS topics, for example, cross-data source detection generalization problems, early intrusion detection problems, unknown/unseen attack detection problem, and the robustness of DL-based NIDS model against adversarial attacks.

Most DL-based IDS methods share a similar workflow. Hence, we can summarize a basic training pipeline for most work in Figure 3. In addition, Learn-IDS also provides many customized training options for improving the training effect, for instance, learning rate schedule, optimizer, regularization, training logs, and checkpoints save/reload. We also introduce many existing methods on IDS models and IDS data structure, which are not our contribution.

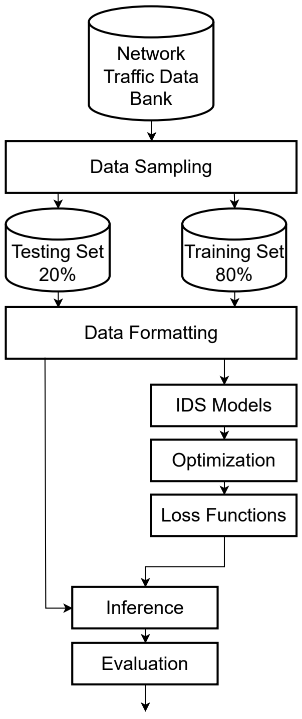


Figure 3. The training and evaluation workflow pipeline.

The metrics for evaluating models' detection performance in terms of

- **Precision:** The number of predictions of the positive class that actually belong to the positive class.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

- **Recall:** The number of predictions of the positive class out of the total number of positive class items in the dataset.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

- **F1-score:** A score that balances precision and recall.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3}$$

5. Experimental Results and Discussion

In this section, we begin by demonstrating the proficiency of Learn-IDS's raw data preprocessing component in accurately handling raw PCAP data. Subsequently, we delve into two illustrative examples that leverage Learn-IDS to investigate the cross-dataset generalization of DL-based IDS models and assess the impact of network flow representation formats.

5.1. Datasets Pre-Processing Result

Correct raw data pre-processing is the ground for the rest of the stages. The majority of IDS works utilized the labeled features tables to train their DL-based models. Taking into account the need for comparison with existing works, new methods with novel data formats should use the same sample instances as prior methods to isolate the improvement of performance from changing data instances. Almost all published datasets include Pcap files and flow-based feature tables, but only the tabular data are labeled. In order to benefit from utilizing the raw Pcap data, Learn-IDS needs to match those raw packets to the labeled flows recorded in the table, so that raw packets also have a label.

The flow can normally be identified by seven sets (source IP, destination IP, source port, destination port, protocol, start timestamp, duration), in which the element duration is chosen by receiving the FIN flag or flow time-out threshold. The time-out threshold is a configurable parameter [41] variable among different datasets. Learn-IDS will maintain the given flow duration threshold of each dataset instead of unifying it. Therefore, the Raw Data pre-processing module matches the raw packets to the flows defined by the original dataset.

We test Learn-IDS's raw data pre-processing module on three datasets: CIC-IDS-2017, ISCX-2012, and UNSW-15. After generating raw packet lists of each flow, we calculate the generated results with the original flow tables. To analyze the characteristics of datasets, we define several measures, which are "unique payload contents amount", "unique source IP number", "average duration per flow", and "average packets number per flow". Considering that different attacks have different behaviors, we count those measures independently among different classes.

The statistical information of three datasets is shown in Tables 2–4. For the unique source IP address, we found both UNSW-15 and CIC-IDS-17 have very few source IPs for most of their attack traffic flows. The UNSW-15's attack flows have four fixed source IPs and most of the attack traffic flows in CIC-IDS-17 only have one fixed source IP, which belongs to the gateway of the testbed. These fixed-source IPs could work as dataset bias, which can hinder the DL-based IDS model from learning the real features of attack behaviors.

Table 2. ISCX-2012 dataset.

Flow Class	Pub Num	Ext Num	Acc	Uniq Payload	Uniq SIP	Total Dur	Total Pkts	Ave Dur	Ave Pkts	Unique Ratio
DDoS_Botnet	37,460	22,595	60.32%	30,684	11	9,074,275.38	6,229,132	401.61	275.69	1.3579
Inside_Attack	20,358	6950	34.14%	20,695	160	62,459.89	53,343	8.99	7.68	2.9776
BruteForce_SSH	5219	5180	99.25%	30,029	1	14,602.88	94,266	2.82	18.20	5.7971
HTTP_DoS	3776	1412	37.39%	53,704	75	43,761.77	124,199	30.99	87.96	38.0339

Table 3. UNSW-15 dataset.

Flow Class	Pub Num	Ext Num	Acc	Uniq Payload	Uniq SIP	Total Dur	Total Pkts	Ave Dur	Ave Pkts	Unique Ratio
Generic	215,481	215,360	99.94%	135,321	4	15,903.51	745,353	0.073	3.46	18.15
Exploits	44,525	44,392	99.70%	947,990	4	100,209.53	2,587,983	2.25	58.29	36.63
Fuzzers	24,246	24,216	99.87%	150,088	4	65,600.74	488,487	2.70	20.17	30.72
DoS	16,353	16,292	99.62%	276,828	4	44,369.21	766,163	2.72	47.02	36.13
Reconnaissance	13,987	13,970	99.87%	63,920	4	14,854.26	174,148	1.06	12.46	36.70
Analysis	2677	2674	99.88%	6507	4	4643.56	26,136	1.73	9.77	24.89
Backdoor	1795	1791	99.77%	6510	4	5273.26	20,864	2.94	11.64	31.20
Shellcode	1511	1511	100.00%	6636	4	548.27	13,981	0.36	9.25	47.46
Backdoors	534	534	100%	735	4	691.42	3954	1.29	7.40	18.58
Worms	174	172	98.85%	6722	4	231.51	14,102	1.34	81.98	47.66

Table 4. CIC-IDS-2017 dataset.

Flow Class	Pub Num	Ext Num	Acc	Uniq Payload	Uniq SIP	Total Dur	Total Pkts	Ave Dur	Ave Pkts	Unique Ratio
Hulk	159,048	134,699	84.69%	81,294	1	9,854,295.53	2,247,117	73.15	16.68	3.61
PortScan	159,023	66,486	41.80%	2281	1	25,651.55	543,961	0.11	2.40	0.41
DDoS	95,123	93,425	98.21%	6151	2	2,833,760.22	1,280,601	30.33	13.70	0.48
GoldenEye	7647	7948	96.06%	11,771	1	226,891.78	111,174	28.54	13.98	10.58
FTP-Patator	3984	4007	99.42%	11,919	1	35,997.33	111,610	8.98	27.84	10.67
SSH-Patator	2988	3195	93.07%	85,424	1	36,978.08	169,583	11.57	53.06	50.37
slowloris	5707	7172	74.32%	983	1	415,723.94	58,612	58.08	8.18	1.67
Slowhttptest	5109	5742	87.61%	7867	1	324,997.07	51,534	56.59	8.97	15.26
Bot	738	735	99.59%	1580	5	72.47	9870	0.09	13.41	16.00
Brute Force	1365	1359	99.56%	9128	1	9773.93	30,121	7.18	22.14	30.30
XSS	679	678	99.85%	2780	1	4770.07	9637	7.02	14.19	28.84
Infiltration	48	33	68.75%	976	1	2818.63	59,753	82.90	1757.44	1.63
Sql Injection	12	12	100%	97	1	164.01	389	4.82	11.44	24.93
Heartbleed	11	11	100%	20,922	1	1217.80	49,295	110.70	4481.36	42.44

For the unique payload contents amount, we found both ISCX-2012 and UNSW-15 datasets always have large amounts of unique payload contents for all classes. However, CIC-IDS-17 has some classes that only own a very limited number of unique payload contents such as the DoS-slowloris attack in CIC-IDS-17 which only has 96 unique payloads for 46,327 packets, and the PortScan attack has 168 unique payloads for 321,208 packets. Meanwhile, we define "Unique ratio", which is the number of unique payload contents divided by the total flow amount. The unique ratio shows the diversity of packet content, and we believe that the training data with less diversity may lead the IDS model to be short of generalization.

Other measures "average duration per flow", and "average packets number per flow" are also shown as dataset bias because they are directly affected by the traffic-generating method or the bottleneck of testbed scenarios. We found that the traffic in UNSW-15 has a shorter average duration than the other two datasets. In the next subsection, we will show how the dataset bias "unique payload contents amount", and "unique source IP number" affect the DL-based model's performance.

5.2. Cross-Dataset Sampling Effect

To demonstrate Learn-IDS's cross-dataset sampling function, we adopt Learn-IDS to prepare training data across multiple datasets for studying the out-of-distribution (OOD) [42] generalization of DL-based NIDS.

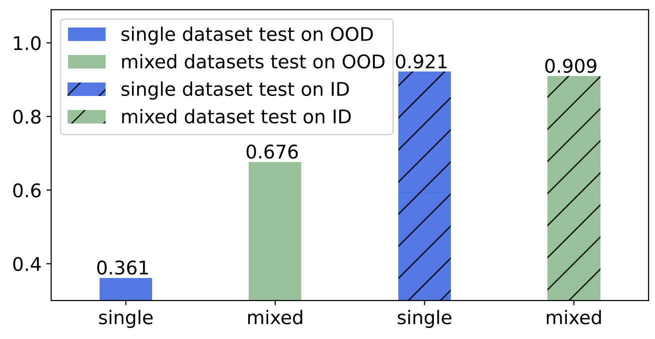
To evaluate the generalization of trained IDS models, we design experiments to determine the robustness of trained models to distribution shifts. We generate distribution shifts by testing trained models on samples from a new dataset, which is different from the training data. The data samples from the same dataset that is used to train the model have the same distribution, known as in-distribution (ID). In contrast, data samples from another dataset have a different distribution, known as out-of-distribution. We call the data samples set using cross-dataset sampling "mixed" set, and the data samples that are all from the same dataset "single" set. Our purpose is to observe the OOD test performance of models that are trained on mixed ID and single ID.

Experiment Setup. The training data were derived from three large-scale IDS datasets: ISCX-12, UNSW-15, and CIC-IDS-17. The training dataset included 8000 normal and 8000 DoS samples. Specifically, the ID train set was composed of data samples solely from dataset A, while the mixed ID train set included samples from both datasets A and B (4000 samples from each dataset). We pre-processed the raw data using the Learn-IDS, extracting unified tabulars for each instance. For the tabular format, we use an auto-encoder (AE) based model to conduct the DoS detection. The auto-encoder model has two components: the auto-encoder and a linear prediction head. Similar to prior methodology [43,44] on OOD performance evaluation. We adopted the Adam optimizer and employed L1 regularization during training. The use of L1 regularization was motivated by its ability to induce sparsity in model parameters, aiding feature selection. The training procedure involved 30 epochs, and we ensured that the model was exposed to a diverse range of normal and DoS flow patterns for robust learning.

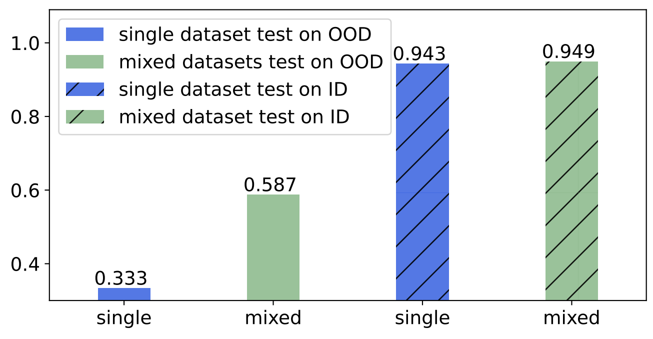
To evaluate the generalization of our trained IDS models, we performed both OOD and ID on a DoS detection task.

- **ID train set:** Single ID set contains 8000 data samples from dataset A; Mixed ID set contains 8000 data samples from dataset A and B.
- **ID test set:** Same as an ID train set, different samples set. (The model trained on dataset A will be tested on 2000 testing samples from dataset A. The model trained on datasets A and B will be tested on 2000 testing samples from datasets A and B)
- **OOD test set:** 2000 testing samples from dataset C for both single and mixed OOD.

We repeat the OOD evaluation experiments three times on different dataset combinations: (a) dataset C: ISCX-2012, A: UNSW-15, B: CIC-IDS-17; (b) dataset C: UNSW-15, A: CIC-IDS-17, B: ISCX-2012; (c) dataset C: CIC-IDS-17, A: ISCX-2012, B: UNSW-15; the performance results are shown in Figure 4. For each dataset combination, the left part shows the OOD test performance and the right part is the ID test performance. We can observe that models can achieve very good results (over 90% accuracy) on the ID test set, but poor performance (less than 70%) on the OOD test set. Although distribution shift will reduce the trained models' performance, we also can see that the model trained on a mixed ID set performs much better than the single one among all three experiments. The OOD test improvement from the mixed to the single shows that the model trained on the mixed dataset has better robustness against the distribution shift. Furthermore, it is noteworthy that the magnitudes of improvement vary among the three combinations. These discrepancies are likely due to the influence of dataset cross-correlation, a factor that diverges across the different dataset combinations. However, quantifying this cross-dataset correlation proves challenging at present. These more robust pre-trained models can work as a better starting point for further fine-tuning the target environment, which has a different distribution from the training data.

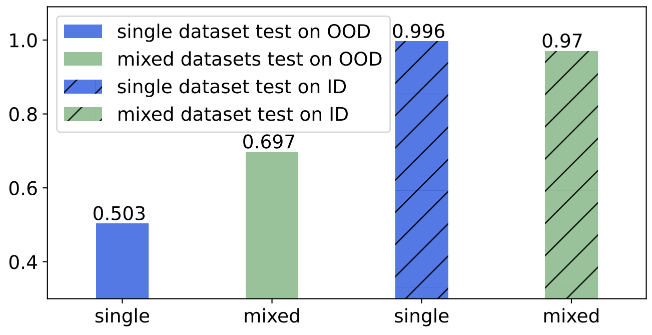


(a) OOD test on ISCX-2012



(b) OOD test on UNSW-15

Figure 4. *Cont.*



(c) OOD test on CIC-IDS-17

Figure 4. OOD performance of models trained on single dataset or mixed dataset for DoS detection task. For (**a**), we conduct OOD test on ISCX-2012 with models trained on UNSW-15 (single) and UNSW-15 + CIC-IDS-17 (mixed). For (**b**), we conduct OOD test on UNSW-15 with models trained on CIC-IDS-17 (single) and CIC-IDS-17 + ISCX-2012 (mixed). For (**c**), we conduct OOD test on CIC-IDS-17 with models trained on ISCX-2012 (single) and ISCX-2012 + UNSW-15 (mixed).

5.3. Influence of Data Format

To demonstrate Learn-IDS's data formatting function, we adopt Learn-IDS to prepare training data in both the tabular format and time series format for studying the impact of data format on DL-based NIDS.

To analyze the influence of data format for DL-based IDS, we design an experiment to show that a novel data format can provide more valuable information for the IDS model, such as time series which can provide additional temporal information. The novel data format is denoted relative to the traditional tabular format, which is statistical values extracted from network traffic. We can notice that the tabular works on projecting the attack's anomaly behaviors to the manual feature dimensions, yet cannot show the behaviors directly. It also should be noted that those statistical values can only be calculated after the traffic finishes its transmission. In the case of attack traffic, it means the attack has been executed. However, a novel data format, such as time series, can present the behaviors by showing features varying with time. Meanwhile, time series also can be partly obtained by the first few packets. Hence, we design experiments to compare the IDS models' performance with the tabular format and features sequence format.

To compare the performance of two different data formats, we noticed two important problems: First, the different data formats may lead the IDS models to have different structures and numbers of weights. If comparing the performance of different models, we cannot claim that additional temporal information can improve the performance; second, there is a conflict between the random packet number of each flow and the requirement of fixed input size of the IDS model. In order to exclude the influence of other factors, we use the same input data structure to represent both tabular and sequence as in Figure 5. As the figure shows, each flow can be represented as a 2D tensor with the shape $N \times L$, where N is the number of features and L is the length of the sequence. The features sequence is combined by L tabulars, the first vector is calculated from the first n packets of this flow, the second one is calculated based on the first $2n$ packets, and so on. The n should meet the requirement that $L \times n$ is less or equal to the total number of packets that belong to this flow. For the flow that only has less than L packets, $n = 1$ and the insufficient parts will be padded with the last tabular. Similarly, the tabular is calculated with n equals the number of packets and the insufficient parts will be padded with the tabular itself.

Experiment Setup. To show the effect of additional temporal information, we ran the comparison experiments with tabular format and features sequence format on four datasets: ISCX-12, UNSW-15, and CIC-IDS-17. On each dataset, each targeted model is trained on 20,000 benign data samples and 20,000 malicious samples and is tested on 5000 benign data samples and 5000 malicious samples. We adopted the Adam optimizer and cross-entropy loss function. Since we have already constructed the tabular and sequence into the 2D tensor with the same shape, we train the 2D CNN model to conduct the multi-class classification IDS task.

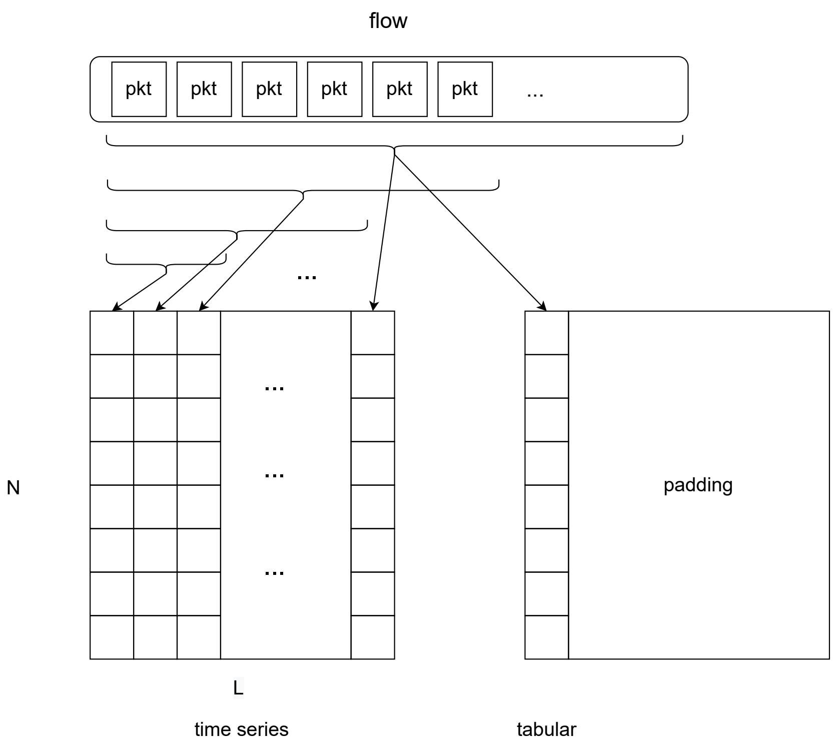


Figure 5. Represent flow in the form of tabular and features sequence in the same shape 2D tensor structure.

The main results are shown in Figure 6, the same CNN model can achieve better performance when using the features sequence format than tabular. We believe the reason is that the time series can provide additional temporal information for the IDS model.

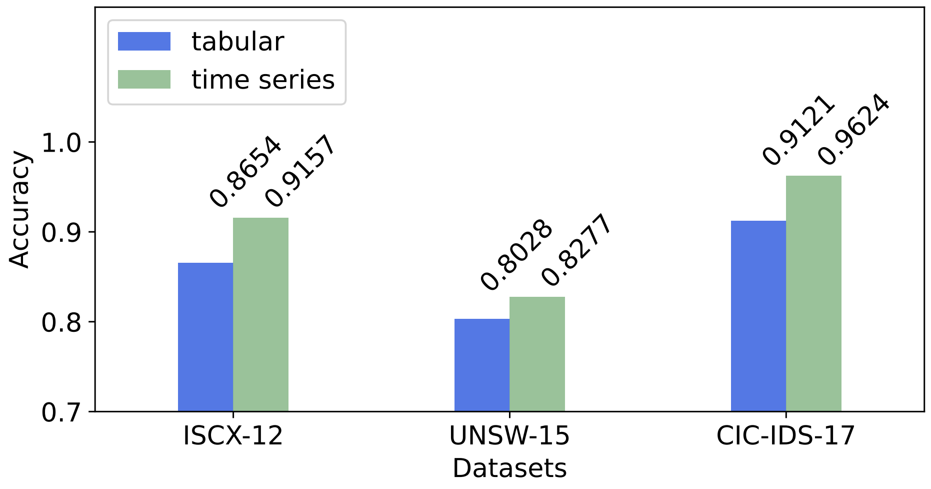


Figure 6. The performance comparison on three datasets with two formats of tabular and time series. The results show that using a features sequence format can achieve better performance than a tabular.

5.4. Results Summary and Discussion

The Learn-IDS framework presents comprehensive capabilities for raw data pre-processing, cross-dataset sampling, and flexible data format representation. The experimental results highlight the potential benefits of each component in enhancing the effectiveness and generalization of DL-based IDS models.

The cross-dataset sampling functionality addresses the challenge of model generalization across different datasets. The experiments in Tables 5 and 6 demonstrate that exposing models to mixed datasets during training improves their ability to handle OOD scenarios. This is particularly valuable in real-world situations where diverse and evolving threats may differ significantly from training data.

The influence of data format on IDS model performance emphasizes the importance of considering the temporal aspects of network traffic. The features sequence format, capturing temporal patterns in network flows, outperformed tabular representations. The results in Table 7 suggest that incorporating time series information can enhance the model's ability to detect complex and dynamic intrusion patterns.

Table 5. OOD performance of models trained on single dataset or mixed dataset for DoS detection task.

	Single				**Mixed**			
Dataset	**Acc**	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
ISCX-12	0.361	0.358	0.335	0.346	0.676	0.691	0.682	0.686
UNSW-15	0.333	0.347	0.358	0.353	0.587	0.578	0.597	0.576
CIC-IDS-2017	0.503	0.496	0.509	0.502	0.697	0.711	0.685	0.698

Table 6. ID performance of models trained on single dataset or mixed dataset for DoS detection task.

	Single				**Mixed**			
Dataset	**Acc**	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
ISCX-12	0.921	0.898	0.944	0.921	0.909	0.916	0.899	0.908
UNSW-15	0.943	0.931	0.949	0.94	0.949	0.960	0.923	0.941
CIC-IDS-2017	0.996	1.00	0.991	0.995	0.97	0.967	0.974	0.971

Table 7. Multi-class detection performance on three datasets with two formats of tabular and time series.

Format	**Tabular**				**Time Series**			
Dataset	**Acc**	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
ISCX-12	0.865	0.803	0.819	0.778	0.915	0.924	0.926	0.921
UNSW-15	0.808	0.799	0.796	0.912	0.827	0.841	0.827	0.829
CIC-IDS-2017	0.912	0.918	0.924	0.912	0.962	0.958	0.945	0.945

In summary, Learn-IDS effectively handles raw network data, demonstrating its potential to address challenges in intrusion detection. The framework's modular components work synergistically to provide a comprehensive solution. The experimental results affirm its capacity to enhance the robustness and performance of DL-based IDS models in diverse and evolving network environments.

6. Conclusions

In this paper, we introduced Learn-IDS, a framework designed to address critical challenges in the realm of deep learning-based Network Intrusion Detection Systems (NIDS). Recognizing the escalating architectural complexity of the Internet and the evolving cyber threat landscape, we identified three key gaps in existing datasets used for training deep learning models. These gaps encompassed issues related to data format diversity, static data instances, and the lack of cross-dataset compatibility. Learn-IDS emerges as a comprehensive, efficient, and extensible tool tailored to bridge these gaps simultaneously. Through a structured framework encompassing Raw Data Pre-Processing, Data Customizing, DL-based IDS Models, and Training and Evaluation, Learn-IDS facilitates the preparation of customized data for diverse DL-based NIDS studies. It efficiently processes raw PCAP traces, supports variable data sampling functions, and provides flexible network traffic representation formats, such as time series, raw byte arrays, and graphs, aligning with the diverse requirements of deep learning models. Our contribution lies in the development of Learn-IDS, a tool that not only overcomes the identified gaps but also showcases its efficacy through the correct processing of raw data from multiple datasets. Two case studies presented in this paper demonstrate Learn-IDS's pivotal role in enhancing the OOD accuracy by around 30% through the cross-dataset sampling function and improving multi-class detection accuracy by an average of 4.1% across three datasets.

Author Contributions: Methodology, software, validation, formal analysis, investigation, resources, data processing, writing—original draft preparation, visualization, M.W.; methodology, supervision, writing—original draft preparation, review and editing, project administration, funding acquisition, N.Y.; writing—review, supervision, Y.G.; writing-review, supervision, N.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the US National Science Foundation under Grant CC-2018919.

Data Availability Statement: The datasets used in this work are available as the following links: https://www.unb.ca/cic/datasets/ids.html, https://research.unsw.edu.au/projects/unsw-nb15-dataset, https://www.unb.ca/cic/datasets/ids-2017.html, https://intrusion-detection.distrinet-research.be/CNS2022/Dataset_Download.html (accessed on 9 March 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shu, X.; Yao, D.; Bertino, E. Privacy-Preserving Detection of Sensitive Data Exposure. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1092–1103. [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
3. Yi, T.; Chen, X.; Zhu, Y.; Ge, W.; Han, Z. Review on the application of deep learning in network attack detection. *J. Netw. Comput. Appl.* **2023**, *212*, 103580. [CrossRef]
4. Chou, D.; Jiang, M. A survey on data-driven network intrusion detection. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–36. [CrossRef]
5. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv* **2018**, arXiv:1802.09089.
6. Zhang, Y.; Chen, X.; Jin, L.; Wang, X.; Guo, D. Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data. *IEEE Access* **2019**, *7*, 37004–37016. [CrossRef]
7. Xu, C.; Shen, J.; Du, X. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [CrossRef]
8. Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 305–316. [CrossRef]
9. Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and don'ts of machine learning in computer security. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), USENIX Association, Boston, MA, USA, 10–12 August 2022.
10. Guarino, I.; Bovenzi, G.; Di Monda, D.; Aceto, G.; Ciuonzo, D.; Pescapé, A. On the use of machine learning approaches for the early classification in network intrusion detection. In Proceedings of the 2022 IEEE International Symposium on Measurements & Networking (M&N), Padua, Italy, 18–20 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
11. Sharon, Y.; Berend, D.; Liu, Y.; Shabtai, A.; Elovici, Y. Tantra: Timing-based adversarial network traffic reshaping attack. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3225–3237. [CrossRef]
12. Layeghy, S.; Baktashmotlagh, M.; Portmann, M. DI-NIDS: Domain invariant network intrusion detection system. *Knowl.-Based Syst.* **2023**, *273*, 110626. [CrossRef]
13. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
14. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.

15. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, NV, USA, 3–6 December 2012.
16. Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv* **2014**, arXiv:1412.5567.
17. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef]
18. He, H.; Sun, X.; He, H.; Zhao, G.; He, L.; Ren, J. A novel multimodal-sequential approach based on multi-view features for network intrusion detection. *IEEE Access* **2019**, *7*, 183207–183221. [CrossRef]
19. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **2017**, *6*, 1792–1806. [CrossRef]
20. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S.; Martinez-del Rincon, J.; Siracusa, D. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 876–889. [CrossRef]
21. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. E-graphsage: A graph neural network based intrusion detection system for IoT. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–9.
22. Hu, X.; Gao, W.; Cheng, G.; Li, R.; Zhou, Y.; Wu, H. Towards Early and Accurate Network Intrusion Detection Using Graph Embedding. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 5817–5831. [CrossRef]
23. Borisov, V.; Leemann, T.; Sessler, K.; Haug, J.; Pawelczyk, M.; Kasneci, G. Deep Neural Networks and Tabular Data: A Survey. *IEEE Trans. Neural Networks Learn. Syst.* **2022**, *early access*.
24. Kadra, A.; Lindauer, M.; Hutter, F.; Grabocka, J. Regularization is all you need: Simple neural nets can excel on tabular data. *arXiv* **2021**, arXiv:2106.11189.
25. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 2346–2363. [CrossRef]
26. Schwengber, B.H.; Vergütz, A.; Prates, N.G.; Nogueira, M. A method aware of concept drift for online botnet detection. In Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
27. Apruzzese, G.; Laskov, P.; de Oca, E.M.; Mallouli, W.; Rapa, L.B.; Grammatopoulos, A.V.; Franco, F.D. The Role of Machine Learning in Cybersecurity. *Digit. Threat. Res. Pract.* **2023**, *4*, 1–38. [CrossRef]
28. Apruzzese, G.; Pajola, L.; Conti, M. The cross-evaluation of machine learning-based network intrusion detection systems. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 5152–5169. [CrossRef]
29. Heine, F.; Laue, T.; Kleiner, C. On the evaluation and deployment of machine learning approaches for intrusion detection. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 4594–4603.
30. Pivarníková, M.; Sokol, P.; Bajtoš, T. Early-stage detection of cyber attacks. *Information* **2020**, *11*, 560. [CrossRef]
31. Ahmad, T.; Truscan, D.; Vain, J.; Porres, I. Early Detection of Network Attacks Using Deep Learning. In Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Valencia, Spain, 4–13 April 2022; pp. 30–39. [CrossRef]
32. Engelen, G.; Rimmer, V.; Joosen, W. Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; pp. 7–12. [CrossRef]
33. Liu, L.; Engelen, G.; Lynar, T.; Essam, D.; Joosen, W. Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In Proceedings of the 2022 IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, 3–5 October 2022; pp. 254–262. [CrossRef]
34. Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. *J. Phys. Conf. Ser.* **2019**, *1192*, 012018. [CrossRef]
35. Binbusayyis, A.; Vaiyapuri, T. Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach. *IEEE Access* **2019**, *7*, 106495–106513. [CrossRef]
36. Adhao, R.; Pachghare, V. Feature selection based on hall of fame strategy of genetic algorithm for flow-based ids. In *Data Science and Security: Proceedings of IDSCS 2021*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 310–316.
37. Jin, M.; Koh, H.Y.; Wen, Q.; Zambon, D.; Alippi, C.; Webb, G.I.; King, I.; Pan, S. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv* **2023**, arXiv:2307.03759.
38. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting deep learning models for tabular data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18932–18943.
39. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 2–9 February 2021; Volume 35, pp. 11106–11115.
40. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
41. Vormayr, G.; Fabini, J.; Zseby, T. Why are my flows different? a tutorial on flow exporters. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 2064–2103. [CrossRef]

42. Hendrycks, D.; Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv* **2016**, arXiv:1610.02136.
43. Shah, H.; Tamuly, K.; Raghunathan, A.; Jain, P.; Netrapalli, P. The pitfalls of simplicity bias in neural networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9573–9585.
44. Shi, Y.; Daunhawer, I.; Vogt, J.E.; Torr, P.; Sanyal, A. How robust are pre-trained models to distribution shift? In Proceedings of the ICML 2022: Workshop on Spurious Correlations, Invariance and Stability, Baltimore, MD, USA, 22 July 2022.