Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Hi-MLIC: Hierarchical Multilayer Lightweight Intrusion Classification for Various Intrusion Scenarios

YUNJI KIM[1], JIHYEON KIM[2], and DONGHO KIM[3]
[1]Dept. of Artificial Intelligence, Dongguk University, Seoul, Korea (e-mail: 2019112107@dgu.ac.kr)
[2]Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea (e-mail: jiwlgus048@dongguk.edu)
[3]Software Education Institute, Dongguk University, Seoul, Korea (e-mail: dongho.kim@dgu.edu)

Corresponding author: Dongho Kim (e-mail: dongho.kim@dgu.edu)

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (No. S-2021-A0496-00167). This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2023-2020-0-01789), and the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2023-RS-2023-00254592) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation). (Yunji Kim and Jihyeon Kim are co-first authors.)

ABSTRACT The need to develop an effective system to detect and classify intrusions in extensive network data exchanges is increasing. We propose ***Hi-MLIC***, a hierarchical multilayer lightweight intrusion classification model that addresses various intrusion types. To address more kinds of intrusion scenarios and validate efficient data formats for intrusion detection, we consolidated packet capture data from two popular benchmark datasets into a new dataset with the two different original dataset formats. We introduced a hierarchical multilayer approach to reduce the misclassification rate of intrusion types caused by an imbalance between benign and malicious data. Layer-1 separates network traffic into malicious and benign. Layer-2 classifies malicious traffic into four groups and Layer-3 identifies 23 specific intrusion types. By performing misclassification analysis and eliminating unnecessary features, we not only improved performance in relation to non-hierarchical classification but also reduced model complexity. Our model achieved a recall metric performance of up to 98.8%.

INDEX TERMS Network Intrusion Detection; Hierarchical Classification; Lightweight Model; Data Format Conversion; Data Consolidation; Machine Learning; Feature Selection

I. INTRODUCTION

In cybersecurity, intrusion detection systems (IDS) play a critical role in protecting network infrastructure by identifying malicious activities and potential security breaches. These systems monitor and analyze network traffic to detect anomalies and suspicious patterns that could indicate an ongoing or imminent cyber intrusion [1]. Therefore, this study aims to create a lightweight classification model for real-time detection capable of detecting and classifying various intrusion types.

Packet capture (PCAP) data, which contains a record of all system interactions, serves as a comprehensive source of information in IDS [2]. However, PCAP data is unsuitable for real-time detection due to processing time and storage capacity constraints. Accordingly, we utilized the CIC-IDS2017 [3] and UNSW-NB15 [4] datasets to extract useful information from PCAP data and provide more kinds of recent intrusion types. These two datasets were consolidated through a data format conversion process to confirm a suitable data format for intrusion detection and enable the detection of various intrusion types.

While the consolidated dataset could handle more kinds of intrusion types, overlapping intrusion types were discovered during the process of merging the two datasets, resulting in classification errors. Additionally, class imbalance issues were identified, where malicious intrusions are either underrepresented compared to benign data or certain intrusion types are overly prevalent. This imbalance in the data made it difficult to classify less frequent intrusion types effectively. To solve these problems, we introduced a hierarchical multilayer approach ***Hi-MLIC*** to solve data imbalance and design an effective classification model. Furthermore, to enhance real-time detection, feature selection techniques were considered to streamline information gathering.

Three feature selection methods have been implemented to achieve a lightweight model with fast classification capabilities for real-time detection. These methods target the removal of features that contribute to misclassification, ensuring that only crucial features are retained. This strategic elimination process enhances the efficiency of the model, facilitating rapid and accurate classification.

Through this, we propose a hierarchical intrusion classification model that identifies various intrusion types based on rich information and contributes to real-time detection.

In summary, we first consolidated two popular benchmark datasets into a new dataset with the two different original dataset formats to detect and classify more kinds of intrusion types by preprocessing and integrating PCAP data. Secondly, we implemented a hierarchical multilayer approach to enhance the accurate classification of specific intrusion types. This approach aims to mitigate misclassification rates arising from the imbalance between benign and malicious data, primarily caused by the prevalence of benign traffic. Finally, we built a lightweight model for real-time and efficient processing that considers more kinds of intrusion types by eliminating unnecessary features that lead to misclassification.

The rest of the paper proceeds as follows: Section 2 discusses previous studies on NIDS systems and studies involving CIC-IDS2017 and UNSW-NB15 datasets. Section 3 explains the dataset creation and preprocessing, hierarchical multilayer approach, and feature selection methods. Section 4 presents experimental results demonstrating the advantages of the hierarchical multilayer approach and analyzes the selected features. Section 5 offers a discussion of future studies. Finally, Section 6 summarizes this study.

II. RELATED WORKS

A. SELECTING AND PREPARING PCAP-BASED DATASETS FOR VARIOUS INTRUSION ANALYSES

In [5], various datasets that can be used for training network intrusion detection models were compared and analyzed. We focused on investigating datasets that effectively reduce the size of large-scale PCAP data while providing various intrusion types, emphasizing datasets suitable for IDS applications. It examined datasets such as CIC-IDS2017, DARPA, KDD CUP 99, NSLKDD, and UNSW-NB15. In this study, we evaluated various datasets like those mentioned, and we decided to use the CIC-IDS2017 and UNSW-NB15 datasets. They are sufficient for modern cybersecurity, extensively employed in various study experiments, and offer detailed insights into packet data and transformation processes.

[3] captured all incoming and outgoing traffic on the main switch of the victim network's major switches and generated packet capture files for the CIC-IDS2017 dataset. The raw logs in PCAP format can be reconstructed into flow-based datasets using the CICFlowMeter sensor proposed in the paper. The CIC-IDS2017 dataset includes various intrusion types such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan, and Botnet intrusions.

[4] connected each server through a router and captured network traffic from the router to create a packet capture file of UNSW-NB15. The PCAP file can be reconstructed into a Flow-based Dataset through the Argus and Bro Sensors proposed in the study. The UNSW-NB15 dataset contains intrusion types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

CIC-IDS2017 is a dataset related to flow time-related statistics and contains information about statistics measured for a fixed time of 1 second. UNSW-NB15 contains connection-related information and flow sequence-related information. Detailed information is mentioned in section 3.1.1. In this study, our objective is to analyze the characteristics of the two benchmark datasets by integrating them. Through this integration, we aim to identify a more effective method for converting PCAP data that enhances intrusion detection and diverse intrusion type classification.

B. MACHINE LEARNING FOR INTRUSION DETECTION AND CLASSIFICATION

[6] proposed a machine learning based data-driven system for active data monitoring. In [7], [8], [9], Deep Belief Networks, Random Forest, and Deep Feedforward Neural Network specification models were each selected, and a study was conducted to compare the performance with several detection models. These studies trained models based on network traffic analysis. They provided implications for network traffic analysis, suggesting the need for an artificial intelligence model.

In [10], [11], [12], AI frameworks capable of performing intrusion detection were introduced. These studies demonstrated the classification of intrusion detection system methodologies into Anomaly-based Detection (AD) and Signature-based Detection (SD). In AD, deep learning models were primarily utilized to detect malicious traffic with anomalies in benign traffic, while SD employed ML models to identify well-known intrusion types. In [13], it was demonstrated that ML requires relatively fewer computational resources and time compared to deep learning, and it is more conducive to interpreting internal operations. Our study primarily aims to rapidly distinguish intrusion types, and considering that the selected benchmark datasets are signature-based, we confirmed the suitability of utilizing ML models.

In [14], [15], and [16], various intrusion types were classified using a hierarchical intrusion detection system. Specific intrusion types are classified after abnormal traffic is identified. [14] achieves about 96% accuracy by reducing misclassification rates through an extension part following anomaly detection and attack classification. [15] proposes the CSK-CNN model which evaluates it on CIC-IDS2017 and UNSW-NB15, achieving over 98% accuracy. Having verified the effectiveness of hierarchical classification in previous studies, particularly in scenarios with increased data volume and diversified intrusion types, we determined that this approach could lead to significant performance improvements. There-

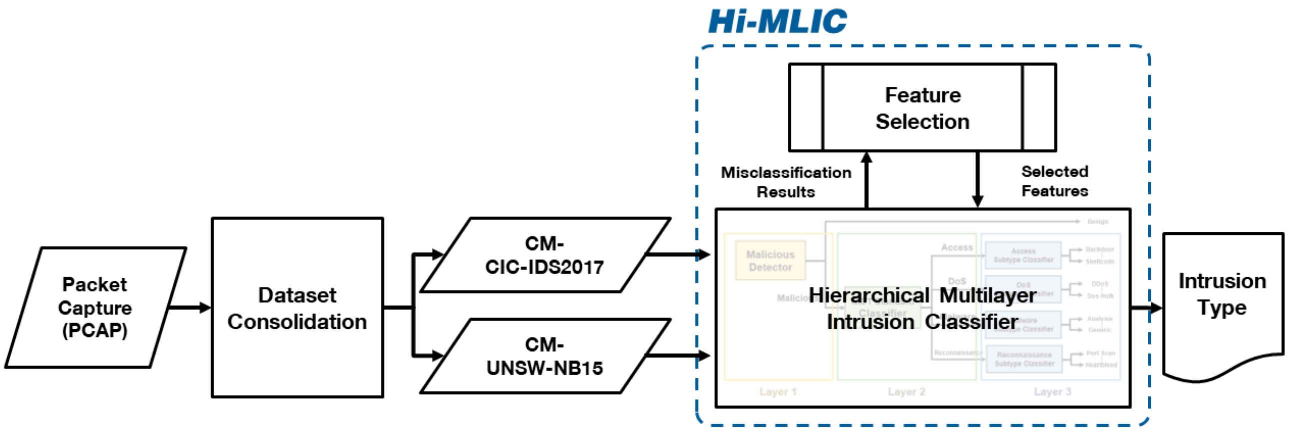

FIGURE 1. **Hierarchical Multilayer Lightweight Intrusion Classification Architecture with Dataset Consolidation**

fore, in Hi-MLIC, we adopted a hierarchical approach to classify various intrusion types within the integrated dataset.

C. FEATURE SELECTION METHODS FOR LIGHTWEIGHT MODEL

Studies focusing on feature importance include [17], considered complex features representing sophisticated intrusions to improve the accuracy of traffic anomaly detection. Information gain was used to rank relevant features and group them by weight. They proposed a method for determining a feature set effective for a specific intrusion by substituting grouped features into a classifier algorithm and analyzing the results. [3] extracted 80 features from the PCAP file using CICFlowMeter, which extracted flow-based features for the CIC-IDS2017 dataset and classified them using the RF class. In this process, They proposed a method to extract the best feature set for each intrusion type detection by calculating each feature's importance, the average standardized mean value of each feature, and the corresponding feature importance in each class.

[18] proposed a flow-based classification method for characterizing encrypted traffic and VPN traffic using only time-related features. Two scenarios were employed for characterizing traffic: one that classifies traffic into VPN and non-VPN, and another that describes traffic without distinguishing VPN and non-VPN. Using the C4.5 and KNN algorithms, it was demonstrated that time-related features are effective classifiers. In [19], XGBoost was used for feature selection, which improved the accuracy of IDS. They selected 19 optimal features from the UNSW-NB15 dataset and combined them with various ML techniques. As a result, the reduced feature vector reduced model complexity and increased accuracy. [20] implemented the feature selection method for IDS using PIO with the primary goal of reducing the number of features while maintaining detection rate and accuracy, and reducing false alarms. [21] used Weka tools and various feature selection algorithms to find the optimal feature combinations in UNSW-NB15.

III. METHODOLOGY FOR EFFICIENT LIGHTWEIGHT INTRUSION CLASSIFICATION

In this section, we elaborate on how we devised Hi-MLIC to be accurate and lightweight, as shown in Figure 1. First, in 3.1, we describe how we performed format conversion of packet capture data to procure more data, obtain more intrusion types, and lighten the dataset for rapid inference. Next, in 3.2, we explain how we designed a hierarchical multilayer approach to detect various intrusions accurately in the imbalanced dataset. A detailed figure is shown in Figure 4. In 3.3, we elucidate the feature selection process to reduce the information to be transformed and formulate a lighter classification model. Finally, through a feature selection process, the Hi-MLIC model, a hierarchical multilayer lightweight intrusion classification model, was devised.

Figure 1 illustrates the overall process of Hi-MLIC. The collected PCAP data is consolidated and merged into CM-CIC-IDS2017 and CM-UNSW-NB15 datasets. Intrusion data from the generated datasets is classified into various intrusion types using the Hi-MLIC. During the data classification process, Hi-MLIC streamlines the lightweight data processing through a feature selection process.

A. CONSOLIDATING DATASET FOR MORE INTRUSION TYPES AND EFFICIENT SIZE REDUCTION

1) CIC-IDS2017 and UNSW-NB15 Data Format

The CIC-IDS2017 and UNSW-NB15 datasets are recognized as key benchmarks in modern network security studies, each reflecting a variety of intrusion detection scenarios on the network. While these datasets provide detailed network traffic information, they are distinguished by their unique collection and processing methods.

CIC-IDS2017 is provided by the Canadian Institute for Cybersecurity and is designed to capture a variety of intrusion patterns under complex network conditions. This dataset includes 7 primary intrusion types and 14 detailed intrusion types, as well as benign traffic. UNSW-NB15, on the other hand, was developed by the Australian Centre for Cyber Security and aims to capture a variety of intrusion patterns

TABLE 1. Organization of the Datasets

Dataset Name	CIC-IDS2017	UNSW-NB15
Provider	Canadian Institute for Cybersecurity	Australian Centre for Cyber Security
Dataset Contents	Mixture of benign traffic, seven primary intrusion types and 14 detailed intrusion types	Mixture of benign traffic and nine intrusion types
Data Collection Period	Five days	Two days
Data Size	51.1 GB of raw data in PCAP format	100 GB of captured raw traffic
Features	84 flow-based features were extracted using CICFlowMeter [22]	49 features were extracted from Argus and Bro-IDS tools [4]

over a shorter collection period. This dataset provides a mix of network traffic, including 9 primary intrusion types as well as benign traffic. Both datasets utilize unique attribute extraction tools to extract valuable information from network traffic. CIC-IDS2017 used CICFlowMeter and UNSW-NB15 used Argus and Bro-IDS tools to extract features. Table 1 summarizes the key features of the CIC-IDS2017 and UNSW-NB15 datasets.

2) Dataset Consolidation Process

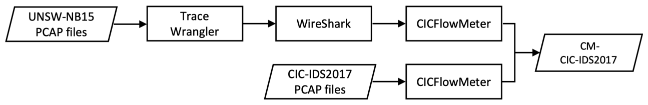

FIGURE 2. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-CIC-IDS2017

Figure 2 depicts the process of combining UNSW-NB15 and CIC-IDS2017 PCAP files using CICFlowMeter [22] to generate CIC-IDS2017 formatted data.CICFlowMeter utilizes the network traffic log data of each dataset to create bidirectional flows. For each generated flow, information is extracted, and statistics are calculated, representing a total of 84 features. The source and destination information of the initially observed packet is used as the directional key for the corresponding flow.

CICFlowMeter only processes the Ethernet header of the packets. As the link layer protocol header in UNSW-NB15 is Linux cooked-mode capture (SLL), it needs to be replaced with the Ethernet header to convert UNSW-NB15 PCAP data for CICFlowMeter. This transformation can be achieved using TraceWrangler software. Subsequently, Wireshark Splitcap is employed to split and remove incorrect packets, and Wireshark Mergecap is used to combine the PCAP files. The processed UNSW-NB15 data and CIC-IDS2017 PCAP data are then transformed into CIC-IDS2017 formatted data using CICIFlowMeter and returned as CSV output. The integrated dataset is named ***CM*-CIC-IDS2017**.

During the feature generation process in CIC-IDS2017 format, missing values and infinite values were identified in the Flow Byts/s and Flow Pkts/s columns. These features are obtained by dividing Bytes and packet count by the flow duration. In cases where packets exist in only one direction, the numerator becomes 0, resulting in NaN values. NaN values have been replaced with 0. Additionally, in cases where the duration is extremely short and returned as 0 by CICFlowmeter, causing the denominator to be 0 and resulting in infinite values, these were replaced with the maximum value in their respective columns.

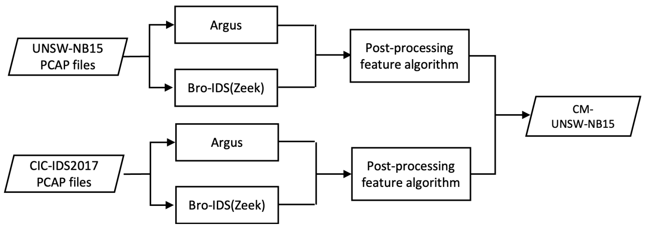

FIGURE 3. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-UNSW-NB15

Figure 3 illustrates the process of combining UNSW-NB15 and CIC-IDS2017 PCAP files through three transformation stages to generate data in the UNSW-NB15 format. Throughout this process, the model employs the Argus tool to generate main features, utilizes Bro-IDS to generate additional features, combines the two datasets using Flow ID, and subsequently generates the remaining features through post-processing [4].

The Argus tool processes network packet data in PCAP format to create bidirectional flow data, employing both argus-server and argus-client components. The former records input PCAP files in binary format within the argus file format, while the latter extracts features from these argus files. Bro-IDS, an open-source network traffic monitor, generates features through three distinct logs: the conn log records essential connection information, the HTTP log captures request and reply details, and the FTP log encompasses all FTP protocol-related activities. By integrating these log files, the desired features are extracted. The features generated by these two tools are merged using the Flow 5-tuple feature, with the Argus feature containing flow-based information and the Bro feature consisting of packet-based features. This consolidated data undergoes a post-processing algorithm [4], generating 11 supplementary features. Forty-nine features are extracted from the UNSW-NB15 and CIC-IDS2017 PCAP data, subsequently provided as UNSW-NB15 format data in CSV output format. We name this consolidated dataset ***CM*-UNSW-NB15**.

During the feature generation in the UNSW-NB15 format, missing values were identified for some CIC-IDS2017 PCAP data not generated by the Argus tool. Assuming that values not generated by Argus imply a value of 0, missing values were imputed as 0. For protocol features with values arp, igm, and sctp, missing values in the S/Dport feature were replaced with 0. Similarly, missing values for sTtl, dTtl, SrcTCPBase, and DstTCPBase were replaced with 0.

TABLE 2. Data Size Reduction after Format Consolidation

	CIC-IDS2017		UNSW-NB15		**Merged**	
PCAP Dataset(GB)	47.90		50.20		**98.10**	
	Size(GB)	Reduction(%)	Size(GB)	Reduction(%)	Size(GB)	Reduction(%)
CICIFlowmeter	1.30	97.29	1.60	96.81	**3.00**	**96.64**
Argus with Bro	0.48	99.01	0.48	99.05	**0.95**	**99.03**

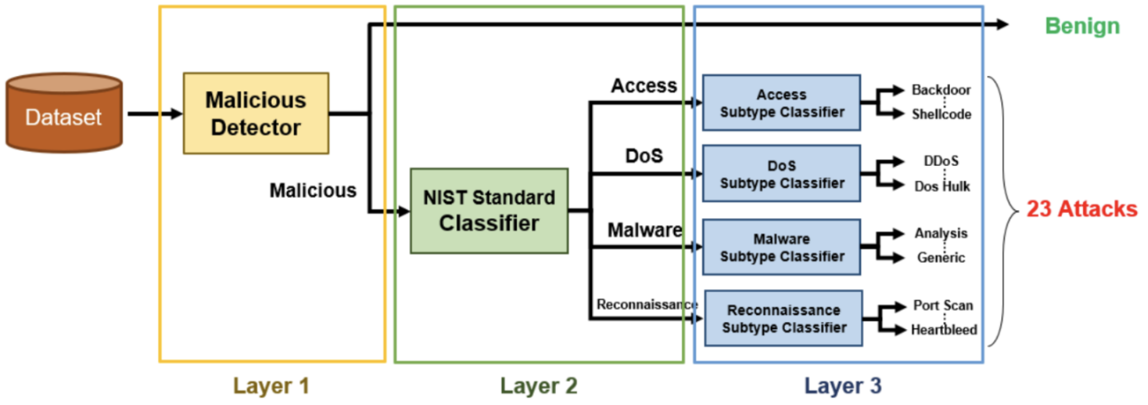

FIGURE 4. Hierarchical Multilayer Intrusion Machine Learning Classifier Framework/Pipeline

3) Data Size Reduction for Lightweight Processing

PCAP data captures information about every packet sent on a network, providing a detailed snapshot of network activity. Because of the depth of detail in this data, direct analysis or processing of PCAP files requires large computing resources and memory. To efficiently manage and process the high volume and detail of PCAP data, it is essential to convert the data to another format. This conversion process can reduce the size of the data while improving the efficiency of analysis and processing. Table 2 shows the results of converting PCAP data to other data formats. The size of the data after conversion and the resulting percentage reduction in size provide a clear picture of the importance and effectiveness of data conversion.

Assuming a data collection period of 30 days with five days of actual data collection, the CIC-IDS2017 dataset amounted to 287.40GB. After conversion into each dataset format, it was reduced to 8.08GB for the CIC-IDS2017 format and 2.86GB for the UNSW-NB15 format. Similarly, assuming a data collection period of 30 days with two days of actual data collection, the UNSW-NB15 dataset was 753.00GB. After conversion into each dataset format, it was reduced to 24.73GB for the CIC-IDS2017 format and 7.45GB for the UNSW-NB15 format. When both datasets collected over 30 days are combined and converted, the total size reaches 1,040.40GB. Depending on the conversion method, the sizes are reduced between 34.17GB and 10.40 GB.

The results demonstrate that converting PCAP data into the CIC-IDS2017 or UNSW-NB15 format significantly reduces storage requirements. This conversion not only improves the ML model's performance but also aids in efficient storage management.

4) Preprocessing for Machine Learning: Encoding and Scaling

For building an ML model, it's crucial to encode character-based data into numerical values and adjust numeric-based data distribution through scaling. Additionally, unnecessary features for training and inference need to be removed. In integrating two benchmark datasets, the information to be excluded includes the timing of intrusion, IP and port values representing the host information involved in the intrusion, and unique IDs for each traffic. All features containing such information were deleted.

Categorical values were encoded using one-hot encoding, and numeric values were scaled using a quantile transformer. While the CM-CIC-IDS2017 dataset consists entirely of numeric values, the CM-UNSW-NB15 dataset includes categorical values such as 'proto,' 'state,' and 'service,' with the rest being numeric. During the one-hot encoding process, the features of the CM-UNSW-NB15 dataset expanded from three categorical features to 178 features.

B. HIERARCHICAL MULTILAYER APPROACH FOR INTRUSION CLASSIFICATION USING MACHINE LEARNING

While combining the CIC-IDS2017 dataset and the UNSW-NB15 dataset, we obtained 23 intrusion types. However, class imbalance issues arose, such as an abundance of benign traffic or certain intrusion types dominating the data. To tackle these challenges and proficiently classify diverse intrusion types, we implemented a hierarchical approach. Figure 4 illustrates a detailed framework of a hierarchical multilayer intrusion classifier, as presented in Figure 1. The Layer-1 detects malicious activities, the Layer-2 classifies malicious traffic into four categories, and the Layer-3 further categorizes them into 23 specific intrusion types. Each layer involves nine ML models with a hyperparameter tuning process.

1) Hierarchical Approach: Three Layers

As shown in Figure 4, Layer-1 functions as a malicious detector, detecting whether the traffic is malicious. The model predominantly learns to differentiate between malicious and benign traffic. Layer-2 operates as a NIST standard classifier, categorizing malicious traffic into four categories: Access, DoS, Malware, and Reconnaissance according to [23].

TABLE 3. Layer-2 Intrusion Groups Categorized by NIST Standard

NIST Category	Intrusion Type	Description
Reconnaissance	PortScan, Web Attack-Brute force, Web Attack-XSS, Web Attack-sql injection, Heartbleed, Reconnaissance	These intrusions involve the gathering of information about a computer system or network to identify vulnerabilities that can be exploited in a future intrusion. This can include intrusions such as port scanning, network mapping, and OS fingerprinting.
Access	FTP-Patator, SSH-Patator, Bot, Infiltration, Backdoor, Shellcode, Exploits, Fuzzers, Worms	These intrusions are designed to gain unauthorized computer system or network access. This can include intrusions such as password guessing, social engineering, and exploiting vulnerabilities in software or hardware.
DoS	DoS, DoS Hulk, DDoS, DoS GoldenEye, DoS slowloris, DoS Slowhttptest	These intrusions are designed to overwhelm a network or computer system with traffic or requests, making it unavailable to legitimate users. This can include intrusions such as flooding a network with traffic or exploiting vulnerabilities in network infrastructure.
Malware	Generic, Analysis	These intrusions involve using malicious software to compromise a computer system or network. This can include intrusions such as viruses, worms, and Trojan horses.

This classification adheres to the NIST standard [23] outlined in Table 3, sorting similar intrusion types initially among the 23 intrusion types. Layer-3, within the previously classified four intrusion categories, a further subdivision into 23 specific intrusion types takes place. The model was trained to discern subtle differences among similar intrusions in more detail.

2) Machine Learning Models for Signature-Based Network Intrusion Classification

The CM-CIC-IDS2017 and CM-UNSW-NB15 datasets primarily focus on commonly known types of intrusions, making them closer to signature-based network intrusions. ML models are widely used in various fields, especially for data-driven pattern recognition and classification tasks, showcasing excellent performance. Thus, using them to classify intrusion types of signature-based network intrusions can be highly effective. They have relatively simple structures requiring less computation and shorter training time than deep learning

TABLE 4. Comparison of Machine Learning Models

Models	Algorithm	Advantages	Disadvantages
DT	Splits feature space to create decision boundaries	Interpretable, easy to explain	Prone to overfitting, sensitive to data
RF	Combines multiple decision trees for stability	Reduced overfitting, high performance, versatile	Interpretability challenges, computationally expensive
NB	Probability calculation based on Bayes' theorem	Simple, efficient, performs well in high dimensions	Strong independence assumption, may not match real-world data
LDA	Creates linear discriminant boundaries between classes	Dimensionality reduction, good performance in high dimensions	Assumes different class covariances, may perform poorly
QDA	Non-linear discriminant boundary creation	Improved performance with differing class covariances	Challenging for high-dimensional data
LR	Predicts probabilities using linear combinations	Interpretable, adaptable updates	Limited for high-dimensional or non-linear problems
Aboost	Combines weak learners to form a strong model	Robust to outliers, high performance	Sensitive to noise, computationally expensive
KNN	Predicts based on majority of k-nearest neighbors	Simple, intuitive, applicable to diverse data	Computationally expensive for large datasets
MLP	Builds complex non-linear models through neural networks	Applicable to various problems, high performance	Requires extensive data, parameter tuning

models. They are known for their interpretability, enabling explanations for why classification results are obtained.

We selected the best model among nine popular ML classification models - Decision Tree (DT) [24], Random Forests (RF) [25] Gaussian Naive Bayes (NB) [26], Linear Discriminant Analysis (LDA) [27], Quadratic Discriminant Analysis (QDA) [28], Logistic Regression (LR) [29], AdaBoost Classifier (Aboost) [30], K-Nearest Neighbor Classifier (KNN) [31], Multilayer Perceptron Classifier (MLP) [32]. Detailed explanations for each model are in Table 4.

3) Hyperparameter Optimization

TABLE 5. Hyperparameter Grid Definition

Models	Hyperparameter Grid
DF	'max_depth': [5, 10], 'min_samples_split':[2,3]
RF	'n_estimators': [10, 50, 100], 'max_depth': [5, 10]
NB	'var_smoothing' : [1e-11, 1e-10, 1e-9]
LDA	'solver': ["svd","lsqr"]
QDA	'reg_param':[0.1,0.2,0.3,0.4,0.5]
LR	'C': [0.001, 0.01, 0.1, 1.0]
Aboost	'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 1.0]
KNN	'n_neighbors': [3,5,7], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan', 'minkowski']
MLP	'hidden_layer_sizes': [(50,), (100,), (150,)], 'activation': ['relu', 'tanh'], 'solver':['adam'], 'learning_rate':['constant'], 'power_t':[0.5], 'alpha':[0.0001], 'max_iter':[10000], 'early_stopping':[False], 'warm_start':[False]

Table 5 presents the important parameters needed for ML model optimization. We fine-tuned the performance of our ML model using Grid Search Cross Validation (GridSearchCV) [33]. By exploring a pre-defined hyperparameter grid for each layer of the model, we could find the most effective hyperparameter combination and optimize the model's performance.

C. REMOVING UNNECESSARY FEATURES FOR LIGHTWEIGHT MODEL

When a large number of features are included in the data, it can increase the complexity of the model, require longer processing times, and confuse predictions with unnecessary information. In network security situations, if the analysis speed is slower than the incoming packet speed, delays in threat detection and response can be critical to security. Therefore, selecting suitable features for an ML model is essential in the ML model design process.

This paper proposes an error analysis feature selection method to filter out unnecessary features. First, in section 3.3.1, we rank important features using the XGBoost algorithm in all layers of the hierarchical multilayer classifier. Based on this evaluation, the final feature selection is determined by two misclassification rate reduction methods: similarity analysis in section 3.3.2 and impact analysis in section 3.3.3. This can improve the performance of the model and increase the analysis speed.

1) Feature Importance Ranking by Ensemble Learning

To calculate the importance of features in high-dimensional and complex datasets, we used the XGBoost algorithm. XGBoost is an ensemble learning method that effectively controls the complexity of the model by limiting the depth of decision trees and adjusting the learning speed [34]. These characteristics help to mitigate the risk of overfitting and improve the prediction accuracy of the model, which is why many feature selection studies have utilized this algorithm.

XGBoost allows for a score representation of feature importance in trained models. In Hi-MLIC, where separate learning models exist for each layer, we had to calculate feature importance for each layer. Layer-1 identified important features for distinguishing between positive and negative instances. Layer-2 highlighted features important for classifying into four intrusion groups, and Layer-3 emphasized features important for distinguishing between 23 more granular intrusion types. Ultimately, we ranked the importance of features by considering all the feature importance scores calculated in each layer.

2) Feature Similarities between Malicious Intrusion Types

Analyzing classification errors, we found that a significant number of intrusions of DoS, Fuzzers, Reconnaissance, Analysis, and Backdoor were misclassified as Exploits, indicating a tendency for predictions to cluster into a particular category of intrusion. We decided to focus on features that may be causing similar intrusion types to be indistinguishable from one another. In other words, we aimed to remove features that do not show significant numerical differences across different types of intrusions. To this end, we devised a method to calculate the distance between the average characteristics of each intrusion type, using this distance as a measure of similarity between intrusions and removing features that exhibit high similarity scores. The formula to calculate the Feature Intrusion Similarity (FIS) is as follows.

$$FIS = \frac{\sum_i^M \sum_j^M (1 - |\mu_i - \mu_j|)}{M^2} \tag{1}$$

Here, FIS represents the Feature Intrusion Similarity, M is the number of intrusion types (23 in our CM datasets), and i, j denotes intrusion type indices. The term μ_i signifies the average of each feature data belonging to the i-th intrusion type.

For each intrusion, the formula calculates the difference between any two types of intrusion. The difference ranges from 0 to 1, where a smaller difference indicates similarity. We then subtract this difference value from 1 to represent this similarity.

The resulting similarity values are summed across all combinations of intrusions to create a similarity matrix for the intrusion combinations, and a total sum is calculated. If all intrusions were identical, the sum of the diagonal elements of the matrix would be M^2.

Therefore, the obtained total sum is normalized by dividing it by M^2 to range from 0 to 1. This value can indicate how much a feature exhibits a high degree of similarity between intrusions.

3) Feature Impact Analysis Comparing Correct and Incorrect Results

From our simulations, the intrusion types, including Fuzzers, Analysis, Backdoor, DoS, Exploits, Reconnaissance, Shellcode, and Worms exhibited a misclassification rate exceeding 50%. To address this, we adopted the following analytical strategy as shown in Algorithm 1.

First, we partitioned the data of these problematic intrusion types into the Correct Group(CG) and Incorrect Group(IG). The CG consists of instances where the predicted type matches the actual type. Conversely, the IG comprises instances where the predicted type does not align with the actual type.

We aim to identify features that reflect the discrepancy group as closely as possible to the CG. This involved calculating the mean value for each feature within both groups and subsequently discerning disparities in these mean values between the groups. Features exhibiting significant mean differences were identified as potential causes for the discrepancies between the two groups, thereby contributing to the high misclassification rate. Therefore, features with pronounced differences, believed to be the main contributors to misclassification, were excluded to enhance classification accuracy.

Algorithm 1 Feature Impact Analysis (CG vs. IG)

```
1: Input: Data instances with features ($X$), predicted type ($y_{\text{pred}}$), true type ($y_{\text{true}}$)
2: Output: Average feature gap between correct group (CG) and incorrect group (IG)
3: 1: Get $X$, $y_{\text{pred}}$, and $y_{\text{true}}$
4: 2: Declare an empty DataFrame $df$ to accumulate feature gaps ($F\_gaps$)
5: 3: Get intrusion types with recall below 50%
6: $low\_recall\_IT \leftarrow []$
7: for each $intrusion_type$ in $intrusion\_types$ do
8:     if $recall(intrusion_type) < 0.5$ then
9:         $low\_recall\_intrusions.append(i)$
10:    end if
11: end for
12: 4: Filter instances by $low\_recall\_IT$
13: $filtered\_inst \leftarrow \text{Filter}(X, y_{\text{pred}}, y_{\text{true}} = low\_recall\_IT)$
14: 5: Create CG and IG groups
15: $CG \leftarrow \text{Filter}(filtered\_inst, y_{\text{pred}} = y_{\text{true}})$
16: $IG \leftarrow \text{Filter}(filtered\_inst, y_{\text{pred}} \neq y_{\text{true}})$
17: 6: Accumulate feature gap between $IG$ and $CG$
18: $F\_gaps \leftarrow \{\}$
19: for each feature do
20:         $avg\_CG \leftarrow \text{Average}(CG[\text{feature}])$
21:         $avg\_IG \leftarrow \text{Average}(IG[\text{feature}])$
22:         $F\_gaps[\text{feature}] \leftarrow \text{abs}(avg\_CG - avg\_IG)$
23: end for
24: 7: $df$ with feature gaps
25: for (feature, gap) in $F\_gaps$ do
26:         $df[\text{feature}] \leftarrow \text{gap}$
27: end for
```

In this section, we explained the Hierarchical Multilayer Lightweight Intrusion Classifier, ***Hi-MLAC***, to enable accurate and rapid classification of various intrusion types in consolidated datasets. This is achieved by a hierarchical approach and feature removal.

IV. EXPERIMENTS AND EVALUATION

This section provides the experimental setup and evaluation procedures. We provide an overview of the consolidated dataset and evaluation configuration, followed by a detailed review of Hi-MLIC performance. We also review the feature selection process.

A. CONSOLIDATED DATASET

1) Two CM Datasets: Configuration

The dataset comprises 23 distinct types of intrusions categorized into four groups according to the NIST standard [23]: Reconnaissance, Access, Denial of Service (DoS), and Malware. To facilitate model training and evaluation, the dataset was divided into a training set and a test set in a 7:3 ratio. Table 6 presents an overall distribution of intrusion types for the CM-CIC-IDS2017 and CM-UNSW-NB15 datasets. The table includes the instance counts for each intrusion type, classified based on the shape of the training and test sets according to NIST categories.

As evident from the table, the dataset initially exhibited a significant class imbalance among the 23 intrusion types. However, by stratifying the data into NIST categories, we successfully mitigated this imbalance. The distribution within each NIST category provides a more balanced representation of intrusion types, enhancing the dataset's suitability for effective model training and evaluation.

2) Evaluation Criteria: False Negative Rates

We selected the recall metric as our primary evaluation criterion. This choice was rooted in the significance of accurate intrusion detection, particularly the importance of minimizing false negative (FN) rates, since not missing malicious incidents is critical. Recall measures the proportion of correctly predicted positive(intrusion) instances among all true positive (TP) instances, providing a clear assessment of the model's ability to identify intrusions. It can be expressed as follows:

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

As evident from the formula, enhancing recall entails reducing FN rates. This aligns well with the imperative of minimizing missed detections in intrusion scenarios. Consequently, the recall value became the criterion for selecting the optimal model at each layer.

For the multi-class classification scenario, we leveraged the Python scikit-learn library [35], setting the average option. Given the inherent class imbalance with many benign instances, we adopted the weighted average approach to address this disparity. This methodology assigns appropriate weights to each label, compensating for the skewed distribution of instances. Our study underscores the importance of prioritizing the reduction of FN rates and utilizing suitable evaluation metrics to ensure accurate intrusion detection.

B. ASSESSMENT OF THE HIERARCHICAL MULTILAYER APPROACH

We propose and compare 1- to 3-step architectures to showcase the performance of our innovative hierarchical approach introduced in Section 3.2.1. The 1-step architecture is non-hierarchical, using only Layer-3, allowing a single ML model to classify data into benign and 23 intrusion types, a total of 24 classes. In the 2-step architecture, Layer-1 classifies incoming data into benign or malicious categories, while Layer-3 refines the process by categorizing malicious data into 23 distinct intrusion types. The 3-step architecture extends the hierarchy, utilizing three layers. Like the 2-step architecture, data is initially categorized as benign or malicious in Layer-1. However, Layer-2 proceeds to classify malicious data into four primary intrusion categories. Finally, Layer-3 classifies data into sub-intrusion types within each category, allowing for a more detailed sequential classification process.

TABLE 6. Distribution of Intrusion Type in Two CM Datasets

Attack type		CM-CIC-IDS2017		CM-UNSW-NB15	
NIST Category	**Intrusion Type**	**Train**	**Test**	**Train**	**Test**
Benign		4,031,522	1,727,796	2,910,333	1,247,286
Reconnaissance	PortScan	111,363	47,679	111,254	47,681
	Reconnaissance	10,642	4,561	9,791	4,196
	Web-Attack-Brute force	1,055	452	956	409
	Web attack-XSS	456	196	454	194
	Web attack-sql injection	15	6	8	4
	Heartbleed	8	3	15	6
Access	Exploits	41,392	17,739	31,167	13,358
	Fuzzers	17,935	7,686	16,972	7,274
	FTP-Patator	5,557	2,381	2,794	1,197
	SSH-Patator	4,128	1,769	2,089	895
	Backdoor	2,849	1,221	1,630	699
	Bot	1,376	590	861	369
	Shellcode	605	260	1,058	453
	Worms	74	31	122	52
	Infiltration	25	11	41	18
DoS	DoS Slowhttptest	161,751	69,322	112,768	48,329
	DDoS	89,619	38,408	67,219	28,808
	DoS	15,745	6,748	11,447	4,906
	DoS GoldenEye	7,205	3,088	6,146	2,634
	DoS Hulk	4,057	1,739	6,094	2,612
	DoS slowloris	3,849	1,650	6,184	2,651
Malware	Generic	12,670	5,430	150,837	64,644
	Analysis	1,186	509	1,874	803
Total Reconnaissance		**134,524**	**57,895**	**132,368**	**57,482**
Total Access		**72,692**	**31,472**	**64,727**	**27,493**
Total DoS		**283,226**	**121,935**	**210,858**	**91,360**
Total Malware		**13,856**	**5,939**	**152,711**	**65,447**
Grand Total		**4,326,654**	**1,849,147**	**3,078,660**	**1,320,065**

1) Optimal Model Selection

We conducted a process to enhance the overall performance of intrusion classification by selecting the optimal model at each layer. Utilizing the GridSearchCV function from the Python scikit-learn library, we explored optimal hyperparameters and selected the best model, as explained in Section 3.2.3. We performed comparative experiments to identify the optimal model using nine ML models introduced in Section 3.2.2 and the hyperparameter grids for each model provided in Table 5. Following the approach outlined in Section 4.1.2, we chose the optimal model based on the highest recall value. Tables 7 and 8 present the selected optimal models and their corresponding recall values for each layer and classifier.

TABLE 7. Performance of the optimal model for each layer of the 2-step architecture. The indicated numbers represent the percentage values for recall.

	CM-CIC-IDS2017		CM-UNSW-NB15	
	Model	**Recall**	**Model**	**Recall**
Layer-1	RF	99.110	KNN	98.346
Layer-2	MLP	91.594	MLP	95.083

TABLE 8. Performance of the optimal model for each layer of the 3-step architecture. The indicated numbers represent the percentage values for recall..

		CM-CIC-IDS2017		CM-UNSW-NB15	
		Model	**Recall**	**Model**	**Recall**
Layer-1		RF	99.110	KNN	98.346
Layer-2		DT	94.588	MLP	96.370
Layer-3	Reconnaissance	RF	99.577	KNN	99.627
	Access	RF	81.899	MLP	88.394
	DoS	KNN	99.896	MLP	99.890
	Malware	AdaBoost	94.039	MLP	99.144

The process of distinguishing between benign and malicious instances remains consistent in both the 2-step and 3-step procedures through Layer-1. In the CM-CIC-IDS2017 dataset, RF demonstrated superior performance, while KNN outperformed other models in CM-UNSW-NB15. This observation supports the notion that the CM-CIC-IDS2017 format is more suitable for distinguishing between benign and malicious instances.

Upon analyzing frequently used models for each dataset, it was found that tree-based ML models such as DT and RF were suitable for classification in the CM-CIC-IDS2017 format, whereas MLP and KNN were effective for the CM-UNSW-NB15 format.

According to Table 8, the classification performance for the Access category in Layer-3 was relatively low. This was mainly due to confusion with other types of intrusions, particularly the "Exploit" intrusion type. Subsequently, an analysis of the Malware category, which also exhibited low classification performance, revealed issues with the "Generic" intrusion type. To determine whether these issues are specific to the 3-step architecture, an analysis of confusion matrices for 1- and 2-step results consistently indicated a problem of misclassifying various intrusion types as either Exploit or Generic.

2) Hierarchical Approach Evaluation

To assess the overall performance of the entire architecture, we conducted evaluations using the same test data as utilized in section 4.2.1.

TABLE 9. Performance Improvement of the 3-step against 1- and 2-step Architecture on CM-CIC-IDS2017. The indicated numbers represent the percentage values for each metric.

CM-CIC-IDS2017				
Classifier	Accuracy	Precision	Recall	F1-Score
1-step	97.598	97.002	97.598	97.183
2-step	97.708	97.171	97.708	97.296
3-step	**97.721**	**97.193**	**97.721**	**97.325**

TABLE 10. Performance Improvement of the 3-step against 1- and 2-step Architecture on CM-UNSW-NB15. The indicated numbers represent the percentage values for each metric.

CM-UNSW-NB15				
Classifier	Accuracy	Precision	Recall	F1-Score
1-step	98.766	98.624	98.766	98.567
2-step	98.800	98.766	98.800	**98.691**
3-step	**98.805**	**98.792**	**98.805**	98.680

Tables 9 and 10 show performance improvement of the 3-step against 1- and 2-step architecture for each CM dataset. On the CM-CIC-IDS2017, performance improved as the step increased, with all metrics peaking at 3-step. For CM-UNSW-NB15, an overall performance improvement was also noted with increasing steps. Except for the F1-Score, the 3-step exhibited the best performance across all metrics, particularly showing a significant difference in precision values. This suggests that the hierarchical approach is effective for performance enhancement.

Compared to the 1-step approach, the performance of both datasets notably improved in the 2-step and 3-step approaches. This underscores the significance of introducing Layer-1, the malicious detector. In particular, through a more detailed analysis of FN, instances 'classified as benign but are intrusions,' it was observed that on the CM-CIC-IDS2017 test dataset, the number of instances decreased from 19,652 at 1-step to 12,114 at 2- and 3-step, representing a 38.36% reduction. On the CM-UNSW-NB15 dataset, the corresponding figure dropped from 6,601 to 3,839, indicating a reduction of 41.88%. This highlights the outstanding malicious detection performance of Hi-MLIC.

A significant improvement in recall was also observed for the most confusing types of intrusion analyzed in section 4.2.1, Exploits and Generic. In the case of CM-CIC-IDS2017, the recall of Exploits increased from 75.64% at 1-step to 98.73% at 3-step and the recall of Generic from 72.00% to 80.22%. On the CM-UNSW-NB15, the recall of Exploits went from 85.45% to 88.64% and the recall of Generic from 97.75% to 99.26%. This indicates that Hi-MLIC effectively reduces the confusion rate when classifying specific intrusions.

C. FEATURE SELECTION EVALUATION

1) Selected Features

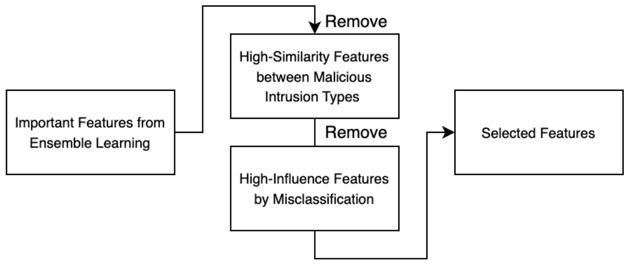
Important Features from Ensemble Learning
Remove
High-Similarity Features between Malicious Intrusion Types
Remove
High-Influence Features by Misclassification
Selected Features

FIGURE 5. Feature Selection Using Three Different Approaches

TABLE 11. Final selected features for each dataset

CM-CIC-IDS2017	CM-UNSW-NB15
bwd_pkts_s, flow_duration, fwd_iat_min, fwd_iat_tot, flow_iat_mean, subflow_bwd_pkts, tot_bwd_pkts, fwd_iat_mean, fwd_pkt_len_min, tot_fwd_pkts, fwd_iat_std, protocol, pkt_len_max, fwd_act_data_pkts, bwd_iat_mean, fwd_pkt_len_std, pkt_size_avg	proto_udp, dmeansz, sintpkt, sloss, dintpkt, state_RST, sbytes, synack, res_bdy_len, service_-, dwin, dttl, swin, ct_state_ttl, service_ssh, dpkts, ackdat, state_REQ, state_CON, state_FIN, service_http, sttl, service_0, dbytes, sload, proto_l2tp, proto_secure-vmtp, proto_vrrp, proto_ddx, proto_wb-mon, proto_ib, proto_trunk-2, proto_fc, proto_srp, proto_etherip, proto_mhrp, service_dhcp, proto_ip, proto_encap, proto_vines

Figure 5 presents the feature selection process performed using three different approaches following the methodology described in Section 3.4. First, we utilized the XGBoost model-based information acquisition method to measure the importance of each feature and select the top 50% of features with the highest importance to generate an initial feature set for the final feature set. Second, we analyzed the feature similarity between intrusion types and removed highly similar features from the feature data set. Thirdly, employing error analysis methods, we evaluated the impact of individual features on the misclassification rate and excluded features that had a significant negative impact. Finally, from the list

TABLE 12. Weight values of the CM-CIC-IDS2017 and CM-UNSW-NB15 selected features for Importance, Similarity, and Misclassification

Dataset	Feature	Feature Importance Score by XGBoost						Similarity	Misclassification
		Layer-1	Layer-2	Layer-3					
				Reconnaissance	Access	DoS	Malware		
CM-CIC-IDS2017	bwd_iat_mean	0.008	0.008	0.000	0.001	0.003	0.003	0.708	0.184
	bwd_pkts_s	0.022	0.022	0.000	0.003	0.000	0.020	0.801	0.045
	flow_duration	0.006	0.006	0.001	0.005	0.042	0.015	0.774	0.036
	flow_iat_mean	0.004	0.004	0.000	0.000	0.007	0.001	0.796	0.024
	fwd_act_data_pkts	0.005	0.005	0.000	0.021	0.054	0.077	0.691	0.339
	fwd_iat_mean	0.009	0.009	0.004	0.001	0.000	0.002	0.797	0.017
	fwd_iat_min	0.021	0.021	0.004	0.010	0.006	0.002	0.743	0.032
	fwd_iat_std	0.012	0.012	0.020	0.002	0.001	0.005	0.777	0.022
	fwd_iat_tot	0.023	0.023	0.001	0.001	0.000	0.001	0.767	0.037
	fwd_pkt_len_std	0.004	0.004	0.000	0.008	0.000	0.068	0.703	0.157
	pkt_len_max	0.007	0.007	0.002	0.012	0.011	0.001	0.708	0.230
	pkt_size_avg	0.140	0.140	0.000	0.002	0.003	0.011	0.711	0.228
	protocol	0.013	0.013	0.002	0.135	0.001	0.025	0.835	0.273
	subflow_bwd_pkts	0.008	0.008	0.000	0.028	0.000	0.003	0.781	0.137
	tot_bwd_pkts	0.005	0.005	0.009	0.031	0.006	0.002	0.780	0.140
	tot_fwd_pkts	0.012	0.012	0.001	0.023	0.012	0.030	0.807	0.051
CM-UNSW-NB15	ackdat	0.006	0.006	0.000	0.001	0.000	0.002	0.627	0.110
	ct_state_ttl	0.273	0.273	0.559	0.012	0.272	0.014	0.546	0.013
	dbytes	0.002	0.002	0.000	0.006	0.012	0.011	0.689	0.060
	dintpkt	0.000	0.000	0.007	0.009	0.009	0.017	0.659	0.092
	dmeansz	0.001	0.001	0.005	0.015	0.011	0.016	0.667	0.034
	dpkts	0.000	0.000	0.000	0.003	0.000	0.001	0.718	0.073
	dttl	0.007	0.007	0.001	0.021	0.004	0.000	0.765	0.110
	dwin	0.000	0.000	0.000	0.000	0.003	0.000	0.621	0.112
	proto_udp	0.001	0.001	0.000	0.056	0.000	0.625	0.833	0.268
	res_bdy_len	0.002	0.002	0.000	0.001	0.004	0.002	0.760	0.031
	sbytes	0.003	0.003	0.000	0.005	0.042	0.164	0.553	0.256
	service_minus	0.001	0.001	0.000	0.004	0.002	0.000	0.002	0.315
	service_0	0.001	0.001	0.000	0.005	0.000	0.000	1.000	1.000
	service_http	0.002	0.002	0.000	0.035	0.004	0.000	0.002	0.913
	service_ssh	0.000	0.000	0.000	0.317	0.000	0.000	0.002	1.000
	sintpkt	0.001	0.001	0.001	0.001	0.009	0.004	0.730	0.913
	sload	0.018	0.018	0.001	0.001	0.004	0.001	0.786	0.904
	sloss	0.002	0.002	0.000	0.003	0.003	0.009	0.786	0.920
	state_CON	0.016	0.016	0.000	0.001	0.020	0.000	1.000	0.988
	state_FIN	0.000	0.000	0.005	0.000	0.045	0.000	0.000	0.889
	state_REQ	0.010	0.010	0.000	0.000	0.199	0.000	0.000	0.991
	state_RST	0.584	0.584	0.403	0.255	0.035	0.000	0.929	1.000
	sttl	0.005	0.005	0.006	0.049	0.006	0.028	0.000	0.992
	swin	0.001	0.001	0.000	0.004	0.006	0.025	0.739	0.888
	synack	0.034	0.034	0.000	0.007	0.000	0.002	0.673	0.898

of excluded features, we reintroduced those that had high importance in the first approach into the dataset, resulting in the selection of the ultimate dataset.

Table 11 represents the final feature set for each dataset selected through three approaches. For each dataset, 17 features were selected as final features for CM-CIC-IDS2017, and 40 features were selected as final features for CM-UNSW-NB15. The results for features specific to each approach are explained in Table 12.

2) Performance and Execution Time Evaluation

The feature selection was consistently applied to all Hi-MLIC layers to reduce the model's complexity and improve computational efficiency.

Using the CM-CIC-IDS2017 dataset, feature selection improved accuracy and recall from 97.721% to 97.800%. However, when the same feature selection method was applied to the CM-UNSW-NB15 dataset, the performance decreased, which could be attributed to the interaction between the characteristics of the dataset and the selected features, as the

TABLE 13. Test results of 3-step on CM-CIC-IDS2017 after applying Feature Selection(FS). The Features column shows the number of features used for classification. The indicated numbers represent the percentage values for each metric.

CM-CIC-IDS2017				
Features	**Accuracy**	**Precision**	**Recall**	**F1-Score**
66	97.721	97.193	97.721	97.325
14	**97.800**	**97.206**	**97.800**	**97.307**

TABLE 14. Performance Improvement of the 3-step against 1- and 2-step Architecture on CM-UNSW-NB15. The indicated numbers represent the percentage values for each metric.

CM-UNSW-NB15				
Features	**Accuracy**	**Precision**	**Recall**	**F1-Score**
208	**98.805**	**98.792**	**98.805**	**98.680**
40	98.791	98.605	98.791	98.516

TABLE 15. Comparison with previous studies. The Intrusions column shows the number of benign and various intrusion types.

Study	Dataset	Hierarchical	Steps	Intrusions	Recall
Verkerken et al. [14]	CIC-IDS2017	✓	2	7	98.34
Song et al. [15]	UNSW-NB15	✓	2	10	98.77
Alazzam et al. [20]	UNSW-NB15	✗	1	10	89.70
Cao et al. [36]	UNSW-NB15	✗	1	10	86.25
Henry et al. [37]	CIC-IDS2017	✗	1	7	98.73
Hi-MLIC	CM-CIC-IDS2017	✓	3	24	97.80
	CM-UNSW-NB15	✓	3	24	**98.79**

feature selection methods were more suitable for the CM-CIC-IDS2017 dataset. More details about each indicator can be found in Tables 13 and 14.

Feature selection also had a positive impact on the learning and inference time of the model. By removing unnecessary features, the amount of information to be processed was reduced, allowing the model to learn and process data quickly. This reduces the response time of the model and improves the efficiency of data processing in resource-constrained environments. The reduction of features led to a lighter model, saving computational resources and storage space.

We compared our intrusion classification results with other studies using the CIC-IDS2017 and UNSW-NB15 datasets in Table 15. Among them, Hi-MLIC using the CM-UNSW-NB15 dataset exhibited the highest recall. While other studies classified fewer than 10 intrusion types, Hi-MLIC successfully handled up to 24 intrusion types, achieving similar or higher levels of recall despite an increased number of attacks. In other words, Hi-MLIC can effectively adapt to a more diverse range of attack types compared to existing models. This indicates that our hierarchical approach has built a more effective classification system compared to previous study.

It is evident that the CM-UNSW-NB15 dataset is a more effective dataset for intrusion classification. Accordingly, we will use the data format of CM-UNSW-NB15 as the format for the dataset to be collected for future intrusion detection system development, with specific details to be discussed in the subsequent discussion section.

V. DISCUSSION

A. EXPECTED EFFECTS OF THE PROPOSED CONSOLIDATED DATASET

The dataset we have proposed offers the advantage of enhancing the model's generalization capabilities by consolidating two benchmark datasets to address more kinds of intrusion types. It also exhibits a significant reduction in size compared to the existing PCAP data, which enhances data management efficiency.

In our future study, we will develop real-time intrusion detection and dynamic access control systems. To explore diverse network environments and intrusion scenarios, we plan to utilize the format of the proposed dataset during the log collection process. Logs collected in the format of the proposed dataset are expected to more accurately reflect our intrusion trends and dynamics, allowing us to strengthen security systems and prepare for new intrusion types. Furthermore, saving data storage space and reducing dataset transmission costs enables efficient data management, while improving data processing speed enhances model training speed for the development of real-time security systems. This is expected to support more realistic and effective security study and system development.

B. HIERARCHICAL MULTILAYER MODEL EFFECTIVE IN DETECTING AND CLASSIFYING VARIOUS INTRUSION TYPES

The proposed hierarchical multilayer approach enhances the accuracy of large-scale datasets by leveraging consolidated benchmark datasets, contributing to the system's overall efficiency. Each layer's model operates as an independent module, individually retraining to provide an adaptive solution for new environments and network intrusions. The independent performance improvements at each layer lead to an overall enhancement in the system's performance.

Using the same feature set across all layers reduces the resources needed for feature processing and storage. This is particularly useful for handling large-scale data analysis, minimizing the computational burden for feature extraction and transformation, and effectively reducing storage space duplication. The hierarchical approach contrasts with the non-hierarchical approach of batch classification by a 1-step model for all network traffic, resulting in an overall increase in system efficiency. Filtering benign data at Layer-1 reduces the amount of data subsequent layers need to process, allowing them to focus on intrusion-type classification and reduce the frequency of FN. This hierarchical multilayer approach overcomes the typical trade-off between efficiency and accuracy by improving both.

Lastly, the characterization of each layer will enhance not only the intrusion detection accuracy of IDS developed in future research, enabling it to detect and classify effectively various intrusion types but also its ability to swiftly and adaptively respond to evolving intrusion scenarios. This approach establishes a crucial foundation for developing advanced systems that can effectively detect and classify intelligent intrusions.

C. CONSIDERING DATA CHARACTERISTICS FOR FEATURE SELECTION

The results of the feature selection method presented in Section 3.3 were summarized in Section 4.3. It was observed that while the feature selection improved performance on the CM-CIC-IDS2017 dataset, it harmed the CM-UNSW-NB15 dataset. This may be attributed to the fact that the CM-CIC-IDS2017 dataset consists entirely of numeric features, whereas the CM-UNSW-NB15 dataset includes categorical features. Some features do not follow a normal distribution centered around the mean value. However, we select the mean value as the representative value for features in sections 3.3.2 and 3.3.3. Without considering their distributions may inadequately capture the characteristics of the features.

For future study, it is suggested to develop an advanced feature selection methodology that takes into account the distribution and characteristics of the features. In particular, for the CM-UNSW-NB15 dataset, the process of encoding categorical features into a one-hot vector format resulted in a significant expansion of the feature space. A more thorough consideration of the encoding process can lead to a model capable of rapid classification using fewer features.

VI. CONCLUSIONS

In this study, we propose Hi-MLIC, a hierarchical multilayer lightweight intrusion classification model. This model is based on machine learning and leverages new consolidated datasets to cover more kinds of intrusions types.

The CM-CIC-IDS2017 and CM-UNSW-NB15 datasets are constructed by consolidating two benchmark datasets. Through data integration, experiments are conducted to determine the suitable data format for network intrusion detection, also resulting in the acquisition of more types of intrusion scenarios.

As consolidating two datasets, data imbalance was revealed. The hierarchical multilayer approach is introduced to reduce misclassification rates resulting from data imbalance. This shows strong performance over the non-hierarchical approach, achieving a recall rate of up to 98.81%. Through this, we confirmed that the model can be trained to address data imbalance by hierarchically adding the layers, enhancing the ease of classification.

Feature selection is necessary to create a lightweight model for real-time applications. We propose new feature selection methods to eliminate features that contribute to misclassification by calculating their scores. This allowed the model to be lighter while maintaining high performance. Especially in the CM-CIC-IDS2017 dataset, which consists of numeric features, a significant performance improvement was observed.

Overall, our model achieved excellent accuracy of 98.81%, precision of 98.79%, recall of 98.81%, and F1 score of 98.68%, as presented in Table 9. This indicates that the Hi-MLIC model can effectively detect and classify various intrusion types, showcasing its potential to respond effectively to intrusions within a network, even when applying diverse response strategies in the future.

ACKNOWLEDGMENT

Dataset access: https://github.com/CSID-DGU/Hi-MLIC

ABBREVIATIONS

The following abbreviations are used in this manuscript:

Aboost	AdaBoost Classifier
AD	Anomaly-based Detection
CG	Correct Group
CM	Consolidated and Merged
DT	Decision Tree
DoS	Denial of Service
FIS	Feature Intrusion Similarity
FN	False Negative
FTP	File Transfer Protocol
GB	Giga Byte
GridSearchCV	Grid Search Cross Validation
Hi-MLIC	Hierarchical Multilayer Lightweight Intrusion Classification
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IG	Incorrect Group
KNN	K-Nearest Neighbor Classifier
LDA	Linear Discriminant Analysis
LR	Logistic Regression
ML	Machine Learning
MLP	Multilayer Perceptron Classifier
NB	Gaussian Naive Bayes
NIDS	Network Intrusion Detection System
PCAP	Packet Capture
QDA	Quadratic Discriminant Analysis
RF	Random Forests
SD	Signature-based Detection
TP	True Positive
XGBoost	Extreme Gradient Boosting

REFERENCES

[1] G. Rekha, S. Malik, A. K. Tyagi, and M. M. Nair, "Intrusion detection in cyber security: role of machine learning and data mining in cyber security," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 3, pp. 72–81, 2020.

[2] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSp*, 2018, pp. 108–116.

[4] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," 2015.

[5] M. Ring *et al.*, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[6] K. Aslansefat *et al.*, "Safeml: safety monitoring of machine learning classifiers through statistical difference measures," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2020.

[7] O. Belarbi *et al.*, "An intrusion detection system based on deep belief networks," *arXiv preprint arXiv:2207.02117*, 2022.

[8] M. N. Goryunov, A. G. Matskevich, and D. A. Rybolovlev, "Synthesis of a machine learning model for detecting computer attacks based on the cicids2017 dataset," *Proceedings of the Institute for System Programming of the RAS*, vol. 32, no. 5, pp. 81–94, 2020.

[9] M. Data and M. Aritsugi, "T-dfnn: An incremental learning algorithm for intrusion detection systems," *IEEE Access*, vol. 9, pp. 154 156–154 171, 2021.
[10] S. Bhatia *et al.*, "Mstream: Fast anomaly detection in multi-aspect streams," in *Proceedings of the Web Conference 2021*, 2021.
[11] H.-Y. Kwon, T. Kim, and M.-K. Lee, "Advanced intrusion detection combining signature-based and behavior-based detection methods," *Electronics*, vol. 11, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/6/867
[12] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Applied Soft Computing*, vol. 92, p. 106301, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620302416
[13] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, September 1 2021. [Online]. Available: https://doi.org/10.1007/s12525-021-00475-2
[14] M. Verkerken, L. D'hooge, D. Sudyana, Y.-D. Lin, T. Wauters, B. Volckaert, and F. De Turck, "A novel multi-stage approach for hierarchical intrusion detection," *IEEE Transactions on Network and Service Management*, 2023.
[15] J. Song, X. Wang, M. He, and L. Jin, "Csk-cnn: Network intrusion detection model based on two-layer convolution neural network for handling imbalanced dataset," *Information*, vol. 14, no. 2, p. 130, 2023.
[16] H. E. Ibrahim, S. M. Badr, and M. A. Shaheen, "Adaptive layered approach using machine learning techniques with gain ratio for intrusion detection systems," *arXiv preprint arXiv:1210.7650*, 2012.
[17] D. Stiawan *et al.*, "Cicids-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.
[18] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.
[19] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, pp. 1–20, 2020.
[20] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert systems with applications*, vol. 148, p. 113249, 2020.
[21] T. Janarthanan and S. Zargari, "Feature selection in unsw-nb15 and kddcup'99 datasets," in *2017 IEEE 26th international symposium on industrial electronics (ISIE)*. IEEE, 2017, pp. 1881–1886.
[22] A. H. Lashkari *et al.*, "Cicflowmeter," https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt, 2017, accessed: 2021-08-10.
[23] R. Blank and P. Gallagher, "Nist special publication 800-30 revision 1 guide for conducting risk assessments," *National Institute of Standards and Technology*, 2012.
[24] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, March 1986.
[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.
[26] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Phil. Trans. R. Soc.*, vol. 53, pp. 370–418, 1763.
[27] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[28] S. Srivastava, M. R. Gupta, and B. A. Frigyik, "Bayesian quadratic discriminant analysis," *Journal of Machine Learning Research*, vol. 8, no. 6, pp. 1–20, 2007.
[29] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.
[30] Y. Freund and R. E. Schapire, "A Decision-Theoretic generalization of On-Line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, August 1997.
[31] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[33] F. Pedregosa, G. Varoquaux *et al.*, "Scikit-learn: Machine learning in python," *Machine Learning in Python*, 2011.
[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html
[36] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, "Network intrusion detection model based on cnn and gru," *Applied Sciences*, vol. 12, no. 9, p. 4184, 2022.
[37] A. Henry, S. Gautam, S. Khanna, K. Rabie, T. Shongwe, P. Bhattacharya, B. Sharma, and S. Chowdhury, "Composition of hybrid deep learning model and feature optimization for intrusion detection system," *Sensors*, vol. 23, no. 2, p. 890, 2023.

YUNJI KIM received a B.S. degree in Information and Communication Engineering from Dongguk University in 2023. She is currently pursuing her Master's degree in Artificial Intelligence at Dongguk University in Korea since 2023. Her research interests include artificial intelligence, machine learning, nework security, and reinforcement-learning.

JIHYEON KIM is currently pursuing her Bachelor's degree in Computer Science and Engineering at Dongguk University in Korea since 2019. Her research interests include networks, security and artificial intelligence.

DONGHO KIM received his BS degree in Computer Engineering from Seoul National University, Korea in 1990 and his MS and Ph.D. degrees in Computer Science from the University of Southern California, Los Angeles, California, USA, in 1992 and 2002, respectively. He is now a professor at the Software Education Institute, Dongguk University, Korea. His research interests include artificial intelligence, distributed systems, networks, and security.