IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 July 2024, accepted 21 August 2024, date of publication 27 August 2024, date of current version 6 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3450671

RESEARCH ARTICLE

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

YUNJI KIM[1], JIHYEON KIM[2], AND DONGHO KIM[3]
[1]Department of Artificial Intelligence, Dongguk University, Seoul 04620, South Korea
[2]Department of Computer Science and Engineering, Dongguk University, Seoul 04620, South Korea
[3]Software Education Institute, Dongguk University, Seoul 04620, South Korea

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

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), Ministry of Education under Grant S-2021-A0496-00167; in part by the Ministry of Science and Information and Communication Technology (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program under Grant IITP-2024-2020-0-01789; and in part by the Artificial Intelligence Convergence Innovation Human Resources Development Supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2024-RS-2023-00254592.

ABSTRACT There is a growing need for systems that can be used to effectively detect and classify intrusions in extensive network data exchanges. To this end, we propose ***Hi-MLIC***, a hierarchical multilayer lightweight intrusion classification model that has been designed to address various intrusion types. This study highlights the challenges involved in classifying intrusions due to data imbalance across different types of intrusion data along with the complex nature of consolidating multiple benchmark datasets into cohesive datasets for real-time detection. To address these issues, we consolidated packet capture data from two widely used benchmark datasets, CIC-IDS2017 and UNSW-NB15, into two newer and more comprehensive datasets, CM-CIC-IDS2017 and CM-UNSW-NB15, respectively. This consolidation enables the identification and classification of a broader range of intrusion types. Our hierarchical approach achieves improved classification accuracy by effectively addressing the class imbalance that is inherent in non-hierarchical models. Layer-1 separates network traffic into benign and malicious categories. Layer-2 further classifies malicious traffic into four groups, while Layer-3 identifies 23 specific intrusion types. We reduced the model complexity and processing time by performing misclassification analysis and eliminating unnecessary features. Our model ultimately achieved a recall metric of up to 98.8%, thus demonstrating its effectiveness and efficiency in intrusion detection and classification. Altogether, the proposed ***Hi-MLIC*** represents a significant advancement in addressing the challenges of real-time network intrusion detection.

INDEX TERMS Network intrusion detection, hierarchical classification, lightweight model, data format conversion, data consolidation, machine learning, feature selection.

I. INTRODUCTION

Intrusion detection systems (IDS) play a critical role in cybersecurity, specifically through 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]. In this context, the present study aims to create a lightweight classification model for real-time detection that is capable of both 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 use in real-time detection due to processing time and storage capacity constraints. Therefore, we used the CIC-IDS2017 [3] and UNSW-NB15 [4] datasets to extract useful information from PCAP data and provide more kinds of

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Ni.

recent intrusion types. A data format conversion process was used to consolidate these two datasets to ensure that the data were in a suitable format for intrusion detection and to enable the detection of various intrusion types.

While the consolidated dataset could handle more kinds of intrusion types, overlapping intrusion types were discovered while merging the two datasets, and these overlaps resulted in classification errors. Class imbalance issues were also identified, either where malicious intrusions were underrepresented compared to benign data or where certain intrusion types were overly prevalent. This imbalance in the data made it difficult to effectively classify less frequent intrusion types. To solve these problems, we introduced a hierarchical multi-layer approach Hi-MLIC to solve data imbalance and design an effective classification model. Further, various feature selection techniques were considered to streamline information gathering with the ultimate aim of enhancing real-time detection.

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

Through this process, 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 converted the raw PCAP data from two popular benchmark datasets into their respective formats and then merged the converted data within each format, resulting in two new datasets. This approach facilitates the detection and classification of more kinds of intrusion types by preprocessing and integrating PCAP data from both datasets. Secondly, we implemented a hierarchical multilayer approach to enhance the accurate classification of specific intrusion types. This approach aims to mitigate misclassification rates stemming from the imbalance between benign and malicious data, which us primarily caused by the prevalence of benign traffic. Altogether, 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 II discusses previous studies on NIDS systems along with prior studies involving CIC-IDS2017 and UNSW-NB15 datasets. Section III explains the dataset creation and preprocessing, hierarchical multilayer approach, and feature selection methods. Section IV presents experimental results demonstrating the advantages of the hierarchical multilayer approach and analyzes the selected features. Section V offers a discussion of future directions for further research. Finally, Section VI 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 to train network intrusion detection models were compared and analyzed. We focused on investigating datasets that effectively reduce the size of large-scale PCAP data while also providing various intrusion types, thus emphasizing datasets that are particularly suitable for IDS applications. The prior study examined datasets such as CIC-IDS2017, DARPA, KDD CUP 99, NSLKDD, and UNSW-NB15. In the current study, we evaluated various datasets such as those mentioned, and we ultimately decided to use the CIC-IDS2017 and UNSW-NB15 datasets. These datasets are sufficient for modern cybersecurity, they have been extensively employed in various study experiments, and they offer detailed insights into packet data and transformation processes.

Reference [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. 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.

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

CIC-IDS2017 is the dataset related to flow time-related statistics and contains information about statistics measured for a fixed time of 1 second. UNSW-NB15 contains both connection-related information and flow sequence-related information. Section III-A1 provides more detailed information. Our objective in the current study 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 both intrusion detection and diverse intrusion type classification.

B. REAL-TIME SUITABILITY OF MACHINE LEARNING ON SIGNATURE-BASED DETECTION

Reference [6] proposed a machine learning-based data-driven system that enables active data monitoring. In [7], [8], and [9], Deep Belief Networks, Random Forest, and Deep Feedforward Neural Network specification models were selected, respectively, and a study was conducted to compare the performance of such models with those of several other detection models. These studies trained models

based on network traffic analysis. They provided implications for network traffic analysis, with their results ultimately suggesting the need for an artificial intelligence model.

In [10], [11], [12], and [13], AI frameworks that are capable of performing intrusion detection were introduced. These studies classified intrusion detection system methodologies into Anomaly-based Detection (AD) and Signature-based Detection (SD). In the AD methodology, Deep Learning (DL) models are primarily used to detect malicious traffic exhibiting anomalies in benign traffic, while the SD methodology uses Machine Learning (ML) models to identify well-known intrusion types.

Since our aim in the present work is to develop a classification model for a real-time intrusion detection system, the computational resources and inference time of the model are crucial considerations. Reference [14] demonstrated that ML requires relatively fewer computational resources and less time than DL, while it is also more conducive to interpreting internal operations. This phenomenon can also be seen in [15]. Although they achieved the best classification performance using the DL model, it had longer training and inference times than ML models. Given the trade-off between classification performance and computational efficiency, we opt for ML models that strike a balance between decent classification accuracy and lightweight, fast operation. This allows us to prioritize real-time processing without sacrificing performance.

Considering the signature-based nature of selected benchmark datasets and our desired purpose of ultimately building a model for use with a real-time system, we confirmed the suitability of our proposed system using ML models.

C. EFFECTIVENESS OF HIERARCHICAL APPROACH IN DETECTION AND CLASSIFICATION

In [16], [17], and [15], various intrusion types were classified using a hierarchical intrusion detection system. In these systems, specific intrusion types are classified after identifying abnormal traffic. [16] achieved about 96% accuracy by reducing misclassification rates through an extension part following anomaly detection and attack classification. Meanwhile, [17] achieved both high efficiency and improved detection accuracy.

Reference [15] proposed the CSK-CNN model and evaluated it on CIC-IDS2017 and UNSW-NB15, with the results showing that it achieved over 98% accuracy. When the number of samples in one class is much higher or lower than the corresponding numbers in the other classes, there is a problem of class imbalance, which leads to poor model performance. To address this problem, the researchers used the CKS sampling technique to adjust the dataset by randomly increasing or decreasing the number of samples in each class until achieving a consistent ratio. They then used this modified dataset to train their model.

Due to our use of an integrated dataset, we had to classify a wider range of intrusion types than has been done in previous studies. Classifying numerous types of intrusions in a single model is difficult to class imbalance issues. 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 performance improvements. Therefore, in Hi-MLIC, we adopted a hierarchical approach to classify various intrusion types within the integrated dataset.

D. FEATURE SELECTION METHODS FOR LIGHTWEIGHT MODEL

Existing studies focusing on feature importance include [18], which considered complex features that represent sophisticated intrusions to improve the accuracy of traffic anomaly detection. Information gain was used in that study to rank relevant features and group them by weight. The authors of that study ultimately proposed a method for determining a feature set that is effective for a specific intrusion by substituting grouped features into a classifier algorithm and analyzing the results. Reference [3] extracted 80 features from the PCAP file using CICFlowMeter, which extracted flow-based features for the CIC-IDS2017 dataset and then classified them using the RF class. Through this process, the authors of that study proposed a method with which 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.

Reference [19] proposed a flow-based classification method for characterizing encrypted traffic and VPN traffic using only time-related features. Two scenarios were used to characterize traffic: one scenario that classified traffic into VPN and non-VPN traffic, and another scenario that described traffic without distinguishing between VPN and non-VPN traffic. Using the C4.5 and KNN algorithms, time-related features were demonstrated to be effective classifiers. In [20], XGBoost was used for feature selection, which improved the accuracy of IDS. The authors of that study selected 19 optimal features from the UNSW-NB15 dataset and combined them using various ML techniques. As a result, the reduced feature vector led to a reduction in model complexity and an increase in accuracy. Reference [21] 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, while also striving to reduce false alarms. [22] 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 depicted in Figure 1. To elaborate, in III-A, we first describe how we performed format conversion of packet capture data to procure more

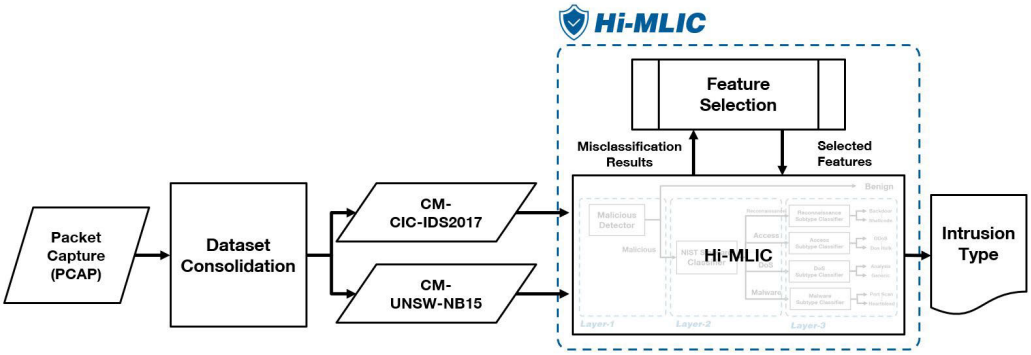

FIGURE 1. Hierarchical multilayer lightweight intrusion classification architecture with dataset consolidation.

data, obtain more intrusion types, and lighten the dataset for rapid inference. Next, in III-B, we explain how we designed a hierarchical multilayer approach to accurately detect various intrusions accurately in the imbalanced dataset. A detailed figure is shown in Figure 4. In III-C, we elucidate the feature selection process that we used to reduce the information to be transformed and ultimately 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. First, the collected PCAP data is consolidated and merged into CM-CIC-IDS2017 and CM-UNSW-NB15 datasets. Next, intrusion data from the generated datasets is classified into various intrusion types using the Hi-MLIC. During the data classification process, Hi-MLIC uses a feature selection process to streamline the lightweight data processing.

A. CONSOLIDATION OF DATASETS 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, as they each reflect a variety of intrusion detection scenarios on the network. While these datasets both provide detailed network traffic information, they can be distinguished by their unique collection and processing methods.

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 [23]	49 features were extracted from Argus and Bro-IDS tools [4]

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 seven primary intrusion types and 14 detailed intrusion types, as well as benign traffic. Meanwhile, UNSW-NB15 was developed by the Australian Centre for Cyber Security, and it aims to capture a variety of intrusion patterns over a shorter collection period. This dataset provides a mix of network traffic, including nine 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 depicts the process of combining UNSW-NB15 and CIC-IDS2017 PCAP files using the CICFlowMeter [23] 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, and 84 features in total are represented. The source and destination information of the initially observed packet is used as the directional key for the corresponding flow.

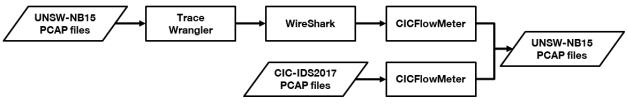

FIGURE 2. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-CIC-IDS2017.

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 must be replaced with the Ethernet header in the process of converting UNSW-NB15 PCAP data for use with the CICFlowMeter. This transformation can be achieved using TraceWrangler software. Subsequently, Wireshark Split-cap is employed to split and remove incorrect packets, while 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 finally 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 were obtained by dividing Bytes and packet count by the flow duration. In cases where packets only existed in one direction, the numerator became 0, resulting in NaN values. NaN values have been replaced with 0. There were also cases where the duration was extremely short and returned as 0 by the CICFlowmeter, thus causing the denominator to be 0 and consequently resulting in infinite values; these were replaced with the maximum value in their respective columns.

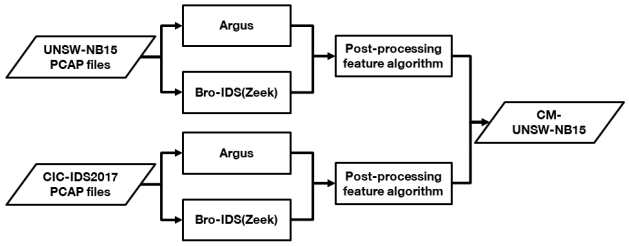

FIGURE 3. Consolidation of UNSW-NB15 and CIC-IDS2017 PCAP data into CM-UNSW-NB15.

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

The Argus tool processes network packet data in PCAP format to create bidirectional flow data while 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, which is 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. Integrating these log files allows for the desired features to be 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], with 11 supplementary features generated as a result. In total, 49 features are extracted from the UNSW-NB15 and CIC-IDS2017 PCAP data, and these are 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 that was 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. Missing values for sTtl, dTtl, SrcTCPBase, and DstTCPBase were also replaced with 0.

3) DATA SIZE REDUCTION FOR LIGHTWEIGHT PROCESSING

PCAP data captures information about every packet that is sent on a network, thus 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, the data must be converted into another format. This conversion process can reduce the size of the data while improving the efficiency of analysis and processing. Table 2 presents the results of converting PCAP data to other data formats. The size of the data after conversion and the corresponding 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 amounted to 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 were combined and converted, the total size amounted 1,040.40GB. Then, depending on the conversion method, the sizes were reduced between 34.17GB and 10.40 GB for the respective formats.

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

4) PREPROCESSING FOR MACHINE LEARNING: ENCODING AND SCALING

In building an ML model, it is crucial to encode character-based data into numerical values and adjust numeric-based data distribution through scaling. It is also necessary to remove unnecessary features for training and inference. 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.

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**

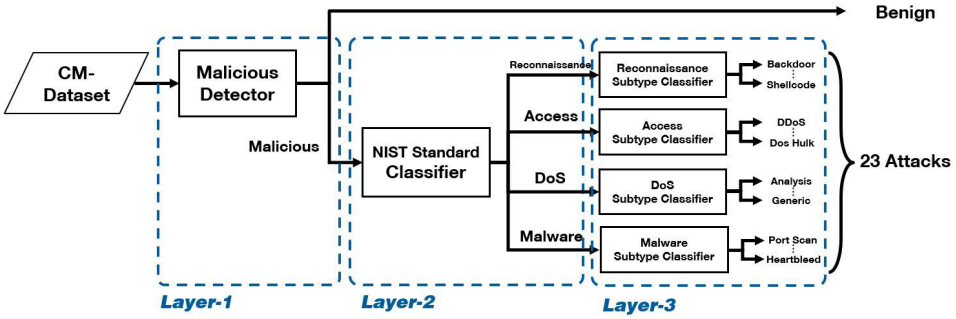

FIGURE 4. Hierarchical multilayer intrusion machine learning classifier framework/pipeline.

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 of the values 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

In the process of combining the CIC-IDS2017 dataset and the UNSW-NB15 dataset, we obtained 23 intrusion types. However, certain class imbalance issues emerged, such as issues involving an abundance of benign traffic or certain intrusion types dominating the data. To address these challenges and classify diverse intrusion types in a proficient manner, we implemented a hierarchical approach. Figure 4 illustrates a detailed framework of a hierarchical multilayer intrusion classifier, as was presented in Figure 1. In this hierarchy, Layer-1 detects malicious activities, Layer-2 classifies malicious traffic into four categories, and 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 that detects whether the traffic is malicious. The model predominantly learns to differentiate between malicious and benign traffic. Layer-2 operates as a NIST standard classifier that categorizes malicious traffic into four categories: Access, DoS, Malware, and Reconnaissance, according to [24].

This classification adheres to the NIST standard [24] outlined in Table 3, which initially sorts similar intrusion types among the 23 intrusion types. In Layer-3, from 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 further 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, which brings them closer to signature-based network intrusions. ML models are widely used in various fields, particularly for data-driven pattern recognition and classification tasks, where they showcase excellent performance. Therefore, it can be highly effective to use such models to classify intrusion types of signature-based network intrusions. They have relatively simple structures requiring less computation and shorter training time than deep learning models. They are known for their interpretability, thus enabling various explanations for why specific classification results are obtained.

We selected the best model among nine popular ML classification models - Decision Tree (DT) [25], Random Forests (RF) [26] Gaussian Naive Bayes (NB) [27], Linear Discriminant Analysis (LDA) [28], Quadratic Discriminant Analysis (QDA) [29], Logistic Regression (LR) [30], AdaBoost Classifier (Aboost) [31], K-Nearest Neighbor Classifier (KNN) [32], and Multilayer Perceptron Classifier (MLP) [33]. Table 4 lists detailed explanations for each model.

3) HYPERPARAMETER OPTIMIZATION

Table 5 presents the important parameters that are needed for ML model optimization. We fine-tuned the performance of our ML model using Grid Search Cross Validation (GridSearchCV) [34]. Exploring a pre-defined hyperparameter

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 with the aim of 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 category can include intrusions such as viruses, worms, and Trojan horses.

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

grid for each layer of the model allowed us to find the most effective hyperparameter combination and optimize the model's performance.

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]

C. REMOVING UNNECESSARY FEATURES FOR LIGHTWEIGHT MODEL

The inclusion of a large number of features in the data can increase the complexity of the model, require longer processing times, and ultimately lead to confused predictions due to the presence of unnecessary information. In network security situations, if the analysis speed is slower than the incoming packet speed, any delays occurring in threat detection and response can be critical to security. It is therefore essential to select suitable features for an ML model in the ML model design process. This paper proposes an error analysis feature selection method that can be used to filter out unnecessary features. First, as described in Section III-C1, 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 III-C2 and impact analysis in Section III-C3. This can improve the performance of the model while also increasing 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 [35]. These characteristics help mitigate the risk of overfitting and improve the prediction accuracy of the model, and these advantages explain why this algorithm has been utilized by many feature selection studies.

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 that are important for classifying into four intrusion groups, and Layer-3 emphasized features that are important for distinguishing between 23 more granular intrusion types. We ultimately ranked feature importance by considering all the feature importance scores that had been calculated in each layer.

2) FEATURE SIMILARITIES BETWEEN MALICIOUS INTRUSION TYPES

After analyzing classification errors, we found that a significant number of intrusions of DoS, Fuzzers, Reconnaissance, Analysis, and Backdoor were misclassified as Exploits, thus indicating a tendency for the model's 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. To this end, we aimed to remove features that did not show significant numerical differences across different types of intrusions: We devised a method to calculate the distance between the average characteristics of each intrusion type, used this distance as a measure of similarity between intrusions, and removed features that exhibited high similarity scores. We used the following formula to calculate the Feature Intrusion Similarity (FIS):

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

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 difference between any two types of intrusion is calculated. This difference ranges from 0 to 1, where a smaller difference indicates more 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 finally 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 such that the normalized values range from 0 to 1. This value can indicate the extent to which a feature exhibits a high degree of similarity between intrusions.

3) FEATURE IMPACT ANALYSIS COMPARING CORRECT AND INCORRECT RESULTS

In our simulations, the intrusion types of Fuzzers, Analysis, Backdoor, DoS, Exploits, Reconnaissance, Shellcode, and Worms all exhibited misclassification rates exceeding 50%. To address this problem, we adopted the analytical strategy 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 close to the CG as possible. This involves 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 are identified as potential causes for the discrepancies between the two groups that might contribute to the high misclassification rate. Therefore, to enhance classification accuracy, features with pronounced differences, which are believed to be the main contributors to misclassification, were excluded.

Algorithm 1 Feature Impact Analysis (CG Vs. IG)

Input: Data instances with features (X), predicted type (y_{pred}), true type (y_{true})
Output: Average feature gap between correct group (CG) and incorrect group (IG)
1: Get X, y_{pred}, and y_{true}
2: Declare an empty DataFrame df to accumulate feature gaps (F_gaps)
3: Get intrusion types with recall below 50%
$low\_recall\_IT \leftarrow []$
for each $intrusion_type$ in $intrusion\_types$ **do**
  if $recall(intrusion_type) < 0.5$ **then**
    $low\_recall\_intrusions.append(i)$
  end if
end for
4: Filter instances by low_recall_IT
$filtered\_inst \leftarrow \text{Filter}(X, y_{\text{pred}}, y_{\text{true}} = low\_recall\_IT)$
5: Create CG and IG groups
$CG \leftarrow \text{Filter}(filtered\_inst, y_{\text{pred}} = y_{\text{true}})$
$IG \leftarrow \text{Filter}(filtered\_inst, y_{\text{pred}} \neq y_{\text{true}})$
6: Accumulate feature gap between IG and CG
$F\_gaps \leftarrow \{\}$
for each feature **do**
  $avg\_CG \leftarrow \text{Average}(CG[\text{feature}])$
  $avg\_IG \leftarrow \text{Average}(IG[\text{feature}])$
  $F\_gaps[\text{feature}] \leftarrow \text{abs}(avg\_CG - avg\_IG)$
end for
7: df with feature gaps
for (feature, gap) in F_gaps **do**
  $df[\text{feature}] \leftarrow \text{gap}$
end for

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

IV. EXPERIMENTS AND EVALUATION

This section details the experimental setup and evaluation procedures we used in this study. In particular, we provide an overview of the consolidated dataset and evaluation configuration, followed by a detailed review of the Hi-MLIC performance. We also review the feature selection process used herein.

A. CONSOLIDATED DATASET

1) TWO CM DATASETS: CONFIGURATION

The dataset comprises 23 distinct types of intrusions that are categorized into four groups according to the NIST

TABLE 6. Distribution of intrusion types in two CM datasets. The column 'Ratio' represents the percentage of instances, while the table is sorted by the number of instances for each NIST category.

Attack type		CM-CIC-IDS2017			CM-UNSW-NB15		
NIST Category	Intrusion Type	Train	Test	Ratio	Train	Test	Ratio
Benign		**4,031,522**	**1,727,796**	**89.093%**	**2,910,333**	**1,247,286**	**84.306%**
Reconnaissance	PortScan	111,363	47,679	2.459%	111,254	47,681	3.223%
	Reconnaissance	10,642	4,561	0.235%	9,791	4,196	0.284%
	Web attack-Brute force	1,055	452	0.023%	956	409	0.028%
	Web attack-XSS	456	196	0.010%	454	194	0.013%
	Web attack-sql injection	15	6	0.000%	8	4	0.000%
	Heartbleed	8	3	0.000%	15	6	0.000%
Total Reconnaissance		**123,539**	**52,897**	**2.730%**	**122,478**	**52,490**	**3.548%**
Access	Exploits	41,392	17,739	0.915%	31,167	13,358	0.903%
	Fuzzers	17,935	7,686	0.396%	16,972	7,274	0.492%
	FTP-Patator	5,557	2,381	0.123%	2,794	1,197	0.081%
	SSH-Patator	4,128	1,769	0.091%	2,089	895	0.060%
	Backdoor	2,849	1,221	0.063%	1,630	699	0.047%
	Bot	1,376	590	0.030%	861	369	0.025%
	Shellcode	605	260	0.013%	1,058	453	0.031%
	Worms	74	31	0.002%	122	52	0.004%
	Infiltration	25	11	0.001%	41	18	0.001%
Total Access		**73,941**	**31,688**	**1.634%**	**56,734**	**24,315**	**1.643%**
DoS	DoS Slowhttptest	161,751	69,322	3.575%	112,768	48,329	3.267%
	DDoS	89,619	38,408	1.981%	67,219	28,808	1.947%
	DoS	15,745	6,748	0.348%	11,447	4,906	0.332%
	DoS GoldenEye	7,205	3,088	0.159%	6,146	2,634	0.178%
	DoS Hulk	4,057	1,739	0.090%	6,094	2,612	0.177%
	DoS slowloris	3,849	1,650	0.085%	6,184	2,651	0.179%
Total DoS		**282,226**	**120,955**	**6.237%**	**209,858**	**89,940**	**6.079%**
Malware	Generic	12,670	5,430	0.280%	150,837	64,644	4.369%
	Analysis	1,186	509	0.026%	1,874	803	0.054%
Total Malware		**13,856**	**5,939**	**0.306%**	**152,711**	**65,447**	**4.424%**
Grand Total		**4,525,084**	**1,939,275**	**100.000%**	**3,452,114**	**1,479,478**	**100.000%**

standard [24]: Reconnaissance, Access, Denial of Service (DoS), and Malware. To facilitate model training and evaluation, the dataset is divided into a training set and a test set in a 7:3 ratio, respectively. Table 6 presents the 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, which are classified based on the shape of the training and test sets according to NIST categories.

As can be seen from the table, the dataset initially exhibited a significant class imbalance among the 23 intrusion types. However, we successfully mitigated this imbalance by stratifying the data into NIST categories. The distribution within each NIST category provides a more balanced representation of intrusion types, which ultimately enhances 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 reflected the significance of accurate intrusion detection, and particularly the importance of minimizing false negative (FN) rates, as it is critical to not miss any malicious incidents. Recall measures the proportion of correctly predicted positive (intrusion) instances among all true positive (TP) instances, thus 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 can be seen 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 emerged as the appropriate criterion with which to select the optimal model at each layer.

For the multi-class classification scenario, we leveraged the Python scikit-learn library [34] while 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 compensates for the skewed distribution of instances by assigning appropriate weights to each label. 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 the innovative hierarchical approach we introduced in Section III-B1. The 1-step architecture is non-hierarchical, as it only uses Layer-3, thus allowing a single ML model to be used to classify data into the benign type and 23 intrusion types, thus comprising 24 classes in total. 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. As occurs in 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, ultimately allowing for a more detailed sequential classification process.

TABLE 7. Optimal models and recall performance for each layer of the 2-step architecture.

	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. Optimal models and recall performance for each layer of the 3-step architecture.

		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

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 the optimal hyperparameters and selected the best model, as explained in Section III-B3. We performed comparative experiments to identify the optimal model using the nine ML models that were introduced in Section III-B2 in addition to the hyperparameter grids for each model, which are provided in Table 5. Following the approach outlined in Section IV-A2, 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.

The process of distinguishing between benign and malicious instances remains consistent across both the 2-step and 3-step procedures through Layer-1. In the CM-CIC-IDS2017 dataset, RF demonstrated superior performance, while in the CM-UNSW-NB15 dataset, KNN outperformed other models. This observation supports the notion that the CM-CIC-IDS2017 format is more suitable for distinguishing between benign and malicious instances. Upon analyzing the frequently used models for each dataset, tree-based ML models such as DT and RF were found to be suitable for classification in the CM-CIC-IDS2017 format, while MLP and KNN were found to be effective for the CM-UNSW-NB15 format.

As presented in Table 8, the Access category in Layer-3 showed relatively low classification performance. This was mainly attributable to confusion with other types of intrusions, particularly the "Exploit" intrusion type. A subsequent analysis of the Malware category, which also exhibited low classification performance, revealed issues with the "Generic" intrusion type. To determine whether these issues were specific to the 3-step architecture, the confusion matrices for 1- and 2-step results were analyzed, and they 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 IV-B1.

Tables 10 and 11 show the performance improvement of the 3-step architecture against the 1- and 2-step architecture for each CM dataset. On CM-CIC-IDS2017, performance improved as the number of steps increased, with all metrics peaking at 3-step.

For CM-UNSW-NB15, an overall performance improvement was also noted with increasing steps. In all metrics aside from the F1-Score, the 3-step architecture exhibited the best performance, particularly showing a significant difference in precision values. This suggests that the hierarchical approach is effective when desiring enhanced performance.

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

Table 9 displays the recall values for each intrusion type, revealing three patterns. First, intrusion types with a larger number of samples tend to achieve higher recall, while those with fewer samples exhibit lower recall. This is caused by the class imbalance in the data, and we addressed this problem using our hierarchical approach. In the 1-step approach, low-frequent intrusion types are properly classified due to the vast ratio of benign samples. We increased the number of layers and adjusted the data distribution for each model's learning. As a result, we achieved better recall as the number of steps increased, particularly in intrusion types such as Shellcode, Bot, and Infiltration.

Secondly, as described in Section IV-B1, many intrusion types were misclassified as Generic or Exploits. Generic contains a general intrusion that does not belong to a specific type, while Exploits denotes the exploitation of vulnerabilities in systems or software. They can often be confused with other types of intrusions as they encompass a broad range of situations.

Moreover, due to the absence of specific characteristics, these two intrusion types were often classified as benign. Our Layer-1 classifier addressed this problem. After classifying intrusions as malicious and benign, it became easier to accurately identify Generic and Exploits. In the case of

TABLE 9. Intrusion type-specific Recall Performance on CM-CIC-IDS2017 and CM-UNSW-NB15 for 1-,2- and 3-step Architecture. The column 'Instances' indicates the number of instances, while the column 'Ratio' shows the percentage of instances used for classification. The percentage values for recall are displayed in the '1-, 2-, and 3-step' columns. The table is sorted based on the number of instances in descending order for CM-UNSW-NB15.

Intrusion Type	CM-CIC-IDS2017					CM-UNSW-NB15				
	Instances	Ratio	1-step	2-step	3-step	Instances	Ratio	1-step	2-step	3-step
Benign	1,727,796	89.093%	99.326	99.079	99.079	1,247,286	84.306%	99.863	99.731	99.737
Generic	5,430	0.280%	72.007	80.941	80.220	64,644	4.369%	97.752	98.059	99.262
DoS Slowhttptest	69,322	3.575%	93.995	99.998	99.997	48,329	3.267%	99.975	99.975	99.973
PortScan	47,679	2.459%	99.742	99.952	99.954	47,681	3.223%	99.992	99.975	99.975
DDoS	38,408	1.981%	99.695	100.000	99.984	28,808	1.947%	99.997	99.983	99.983
Exploits	17,739	0.915%	75.636	99.337	98.729	13,358	0.903%	85.447	87.745	88.636
Fuzzers	7,686	0.396%	1.106	12.225	19.778	7,274	0.492%	33.379	51.251	50.976
DoS	6,748	0.348%	0.133	0.063	0.438	4,906	0.332%	2.752	7.644	4.280
Reconnaissance	4,561	0.235%	1.951	35.172	34.223	4,196	0.284%	76.335	73.427	76.740
DoS slowloris	1,650	0.085%	77.636	99.720	99.720	2,651	0.179%	90.079	97.359	97.548
DoS GoldenEye	3,088	0.159%	88.731	99.782	99.746	2,634	0.178%	96.659	99.506	99.431
Dos Hulk	1,739	0.090%	77.861	99.633	99.695	2,612	0.177%	99.732	99.694	97.320
FTP-Patator	2,381	0.129%	76.732	99.958	100.000	1,197	0.091%	99.916	99.916	99.916
SSH-Patator	1,769	0.096%	46.693	99.880	100.000	895	0.068%	100.000	100.000	100.000
Analysis	509	0.026%	0.000	0.000	0.000	803	0.054%	0.000	3.611	2.740
Backdoor	1,221	0.066%	0.000	0.000	0.000	699	0.053%	2.289	3.577	6.295
Shellcode	260	0.014%	0.000	0.000	15.517	453	0.034%	5.519	10.596	22.075
Web Attack-Brute Force	452	0.023%	0.000	99.459	98.649	409	0.028%	99.022	98.778	98.778
Bot	590	0.032%	0.169	100.000	95.808	369	0.028%	59.350	72.629	72.629
Web Attack-XSS	196	0.010%	0.000	0.535	3.209	194	0.013%	4.639	4.639	4.639
Worms	31	0.002%	0.000	0.000	0.000	52	0.004%	0.000	13.462	11.538
Infiltration	11	0.001%	0.000	50.000	50.000	18	0.001%	61.111	72.222	72.222
Heartbleed	3	0.000%	0.000	100.000	100.000	6	0.000%	83.333	83.333	83.333
Web Attack-Sql Injection	6	0.000%	0.000	0.000	0.000	4	0.000%	0.000	75.000	0.000

TABLE 10. Performance of 1-, 2-, and 3-step architectures on CM-CIC-IDS2017 dataset.

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 11. Performance of 1-, 2-, and 3-step architectures on CM-UNSW-NB15 dataset.

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

CM-CIC-IDS2017, the recall of Exploits increased from 75.64% at 1-step to 98.73% at 3-step, while the recall of Generic from 72.00% to 80.22%, respectively. Meanwhile, in the case of CM-UNSW-NB15, the recall of Exploits went from 85.45% to 88.64% and the recall of Generic went from 97.75% to 99.26% when moving from 1-step to 3-step, respectively. Intrusion types such as Fuzzers, Analysis, and Backdoor that were merely classified as Generic and Exploits also improved.

Lastly, using the CM-CIC-IDS2017 dataset, it was difficult to classify intrusion types from the UNSW-NB15 dataset. This indicates that the conversion method proposed by CIC-IDS2017 may not perform well for different intrusion types, in which case it would lack generalizability. Adding hierarchical steps proved to be helpful in addressing these issues. As the number of steps increased from 1 to 3, there were improvements in the scores for Fuzzers, DoS, and Shellcode. Fuzzers saw a rise in recall from 1.106 to 19.778, DoS saw a corresponding rise from 0.133 to 0.438, and Shellcode saw a corresponding rise from 0 to 15.517. This indicates that our approach is effective for intrusion types with low accuracy.

These findings are consistent with those of other studies. We conducted a comparison with other studies that is described in Section IV-D2.

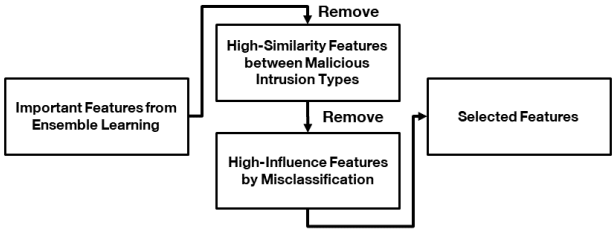

FIGURE 5. Feature selection using three different approaches.

C. FEATURE SELECTION EVALUATION

1) SELECTED FEATURES

Figure 5 presents the feature selection process that was performed using three different approaches while following the methodology described in Section III-C. First, we used the XGBoost model-based information acquisition method to measure the importance of each feature, after which we selected 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, we used error analysis methods to evaluate the impact of individual features on the misclassification rate and excluded features that had a significant negative impact. Finally, from the list of excluded features, we reintroduced

those that had high importance in the first approach into the dataset, thus resulting in the selection of the ultimate dataset.

TABLE 12. 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

Table 12 presents the final feature set for each dataset selected through the three approaches. As listed in the table, 17 features were selected as final features for CM-CIC-IDS2017, while 40 features were selected as final features for CM-UNSW-NB15. The results for features specific to each approach are presented in Table 13.

2) PERFORMANCE AND EXECUTION TIME EVALUATION

To reduce the model's complexity and improve its computational efficiency, feature selection was consistently applied to all Hi-MLIC layers. We named the model Hi-MLIC-Heavy when feature selection was not utilized and Hi-MLIC-Lightweight when selected features were employed.

When 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 specific feature selection methods used were more suitable for the CM-CIC-IDS2017 dataset. More details about each indicator can be seen in Table 14.

Feature selection also had a positive impact on the learning and inference time of the model. We conducted experiments to evaluate the necessary computational resources and time for training and testing of the Hi-MLIC-Heavy and Hi-MLIC-Lightweight models on the CM-CIC-IDS2017 and CM-UNSW-NB15 datasets. These experiments were performed on a system equipped with Intel(R) Xeon(R) Gold 6426Y x 2EA processors.

Since the Heavy model had more features, it exhibited a higher computational cost regarding time and memory usage. For the CM-CIC-IDS2017 dataset, the Heavy model required 207.541 seconds and 81.533 MB with a memory usage of 0.615 MB, while the Lightweight model only required 49.342 seconds, 44.905 MB, and 0.277 MB of memory. Similarly, for the CM-UNSW-NB15 dataset, the Heavy model required 354.486 seconds and 796.625 MB with a memory usage of 2.514 MB, while the Lightweight model only needed 64.658 seconds, 174.369 MB, and 0.723 MB of memory. These findings indicate that the Lightweight model, while achieving slightly lower accuracy, uses significantly reduced computational resources, thereby making it more suitable for real-time applications where resource efficiency represents a critical consideration.

By removing unnecessary features, the amount of information to be processed was reduced, which ultimately allowed the model to learn and process data quickly. This reduced the response time of the model and improved the efficiency of data processing in resource-constrained environments. The reduction of features led to a lighter model that saved computational resources and storage space.

D. COMPARISON WITH PREVIOUS STUDIES

1) FINAL CLASSIFICATION RESULT COMPARISON

Using Hi-MLIC, we achieved high-performance classification across 24 types, thereby surpassing existing studies that were limited to only seven or 10 types. Our classification method was validated using the CIC-IDS2017 and UNSW-NB15 datasets as well as compared with other studies in Table 15.

Hi-MLIC exhibited excellent recall performance, which was our priority evaluation metric. While its accuracy was relatively lower than that presented in [15], it was difficult to make a direct comparison, since their research did not provide recall metrics.

Table 16 compares the key characteristics of Hi-MLIC with previous studies, and this comparison allows us to summarize the strengths and weaknesses of our approach. Hi-MLIC emphasizes real-time processing while managing a broader variety of intrusion types compared to existing studies. Unlike previous studies that have typically focused on a single dataset, Hi-MLIC utilizes consolidated datasets such as CM-CIC-IDS2017, and CM- UNSW-NB15, covering 24 types of intrusions. Despite the increased variety of intrusion types it considers, Hi-MLIC demonstrated an improved recall for each type of intrusion, particularly for those with smaller sample sizes. This indicates that Hi-MLIC has a superior ability to effectively manage and classify diverse intrusion scenarios.

Hi-MLIC outperforms existing studies in terms of classification, as presented in Table 15. The key performance metrics for Hi-MLIC include accuracy reaching up to 98.81%, precision reaching up to 98.79%, recall reaching up to 98.81%, and F1-Score reaching up to 98.68%. These metrics indicate that Hi-MLIC can effectively detect and classify various intrusion types with high reliability. Further, Hi-MLIC's hierarchical multilayer lightweight model balances complexity and performance, thus providing a stable and robust solution that consistently performs above average while ultimately ensuring that dependable results will be obtained across different datasets.

While Hi-MLIC's advanced feature selection methods and hierarchical approach significantly enhance performance, their use of multiple models leads to increased execution

TABLE 13. Weight values of the features from 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

TABLE 14. Evaluation results of Hi-MLIC-Heavy and Hi-MLIC-Lightweight on CM-CIC-IDS2017 and CM-UNSW-NB15. The features column shows the number of features selected. Metrics include Accuracy; Precision; Recall; F1-Score; train and test computation time in seconds; and model size and memory usage in megabytes.

Dataset	Model	Features	Accuracy	Precision	Recall	F1-Score	Train(s)	Test(s)	Size(MB)	Memory(MB)
CM-CIC-IDS2017	Heavy	66	97.721	97.193	97.721	97.325	29,381.169	207.541	81.533	0.615
	Lightweight	14	97.580	97.206	97.380	97.307	5,135.342	49.342	44.905	0.277
CM-UNSW-NB15	Heavy	208	98.805	98.742	98.386	98.580	41,554.486	354.486	796.625	2.514
	Lightweight	40	98.791	98.605	98.791	98.516	7,084.658	64.658	174.369	0.723

TABLE 15. Comparison with previous studies. The Intrusions column shows the number of benign and various intrusion types. The sections marked with '-' are left blank as the corresponding study does not provide such information. Hi-MLIC-'Heavy' describes a model without feature selection, while 'Lightweight' demonstrates a model with feature selection.

Dataset	Study	Model	Hierarchical	Steps	Intrusions	Accuracy	Precision	**Recall**	F1-Score
CIC-IDS2017	Song et al. [15]	CKS-CNN	✓	2	7	**99.91**	-	-	-
	Henry et al. [36]	CNN-GRU	✗	1		98.73	73.85	69.10	71.39
	Verkerken et al. [16]	OC-SVM/RF	✓	2		98.34	99.26	98.34	98.75
	Hi-MLIC	Heavy	✓	3		99.88	**99.88**	**99.88**	**99.88**
		Lightweight				98.47	98.52	98.47	98.47
UNSW-NB15	Song et al. [15]	CKS-CNN	✓	2	10	**98.77**	-	-	-
	Yang et al. [7]	MDPCA-DBN	✗	1		90.21	87.30	96.22	91.54
	Cao et al. [37]	CNN-GRU	✗	1		86.25	86.92	86.25	86.59
	Alazzam et al. [21]	Sigmoid-PIO	✗	1		91.30	**99.18**	89.70	90.40
	Hi-MLIC	Heavy	✓	3		97.75	97.79	**97.75**	**97.72**
		Lightweight				97.38	97.95	97.38	97.48
CM-CIC-IDS2017	Hi-MLIC	Heavy	✓	3	24	97.72	97.19	97.72	97.33
		Lightweight				97.80	97.21	97.80	97.31
CM-UNSW-NB15		Heavy				**98.81**	**98.79**	**98.81**	**98.68**
		Lightweight				98.79	98.61	98.79	98.52

time. This can be a disadvantage when applied to real-time systems. To mitigate this issue, the model can be configured to initially use only the component that differentiates between benign and malicious activities, thus prioritizing

TABLE 16. Comparison of Hi-MLIC and previous studies.

Criterion	Hi-MLIC	Previous Studies
Dataset Coverage	CIC-IDS2017, UNSW-NB15, CM-CIC-IDS2017, CM-UNSW-NB15	Typically limited to a single dataset, e.g., CIC-IDS2017 or UNSW-NB15
Number of Intrusions	24 intrusion types	Limited to 7-10 intrusion types
Specific Intrusion Detection	Improved recall for many intrusion types, especially with small sample sizes	Lower recall for specific intrusion types, often due to class imbalance and dataset limitations
Accuracy	Up to 98.81%	Up to 99.91% in specific models, but lower when recall is considered
Precision	Up to 98.79%	Ranges from 73.85% to 99.26% depending on the model and dataset
Recall	Up to 98.81%, prioritized in evaluation	Often not reported or lower
F1-Score	Up to 98.68%, prioritized in evaluation	Ranges from 71.39% to 98.75% depending on the model and dataset
Feature Selection	Advanced feature selection methods improving lightweight performance	Feature selection not always emphasized; when used, simpler methods are applied
Class Imbalance Handling	Hierarchical steps to mitigate imbalance	Techniques like CKS sampling (random adjustments of sample numbers) used in some models
Model Complexity	Hierarchical Multilayer Lightweight Model	Single-layer or two-layer models

the prevention of malicious intrusions. Detailed solutions for implementing this approach in real-time systems are elaborated upon in Section V-D of the manuscript.

2) RESULT COMPARISON BY SPECIFIC INTRUSION TYPE

We addressed the class imbalance issue that commonly occurs in intrusion detection datasets. Our Hi-MLIC effectively mitigates this problem, thus improving performance across various intrusion types. Class imbalance can cause models to be biased toward majority classes, ultimately resulting in lower detection rates for minority classes. To address this potential issue, we adopted a hierarchical classification strategy that divides the data into more balanced subsets before performing detailed classification.

The experimental results presented in Tables 17 and 18 demonstrate the effectiveness of our approach, as it exhibits increased classification performance for specific intrusion types. Using the CIC-IDS2017 dataset, out of the 15 types, nine show better classification performance compared to [38]. With the UNSW-NB15 dataset, out of the 10 intrusion types, seven exhibit better classification performance compared to [7].

This table is helpful for understanding how our hierarchical approach addresses class imbalance issues. By examining the ratio of each intrusion type, we can see that our method improves the detection rates of less frequent intrusion types.

Specifically, in the Access category of the CIC-IDS2017 dataset, there was a very low number of instances for each specific intrusion relative to the entire dataset, which resulted in low data ratios and reduced recall. By hierarchically classifying the data, we increased the proportion of each intrusion type within a single model. This approach resolved the class imbalance issue and effectively improved accuracy, with SSH-Patator improving from 97 to 100, Bot from 79 to 96, and Infiltration from 0 to 50. Similar improvements were observed in the DoS category of the CIC-IDS2017 dataset and the Access category of the UNSW-NB15 dataset, where there were significant increases in previously low data ratios of specific intrusion types, ultimately leading to enhanced performance.

Through this experiment, we verified that our hierarchical approach addresses class imbalance and enhances classification performance for various intrusion types.

V. DISCUSSION

A. EXPECTED EFFECTS OF THE PROPOSED CONSOLIDATED DATASET

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

In a future study, we aim to 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, which would in turn allow us to design strengthened security systems and prepare for new intrusion types. Saving data storage space and reducing dataset transmission costs also enables efficient data management, while improving data processing speed enhances the model training speed for the development of real-time security systems. These benefits are expected to support more realistic and effective security research 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, thus contributing to the system's overall efficiency. Each layer's model operates as an independent module, while individually retraining to provide an adaptive solution for new environments and network intrusions. The independent performance improvements achieved at each layer accumulate to lead to an overall enhancement in the system's performance.

TABLE 17. Intrusion type distribution and improvement in classification performance for CIC-IDS2017 and UNSW-NB15 Datasets. This table shows the number of instances and the ratio of each intrusion type within the respective dataset and NIST category. The column 'Ratio in NIST' represents the percentage of each intrusion type within its NIST category. The 'Improvement' column highlights the enhancement in classification recall compared to previous studies.

Dataset	NIST Category	Intrusion Type	Instances	Ratio	Ratio in NIST	Improvement
CIC-IDS2017	Benign		2271320	80.320%		
	Reconnaissance	PortScan	158804	5.620%	98.637%	99 to 100
		Web Attack-Brute Force	1507	0.053%	0.936%	78 to 99
		Web Attack-XSS	652	0.023%	0.405%	-
		Web Attack-SQL Injection	21	0.001%	0.013%	-
		Heartbleed	11	0.000%	0.007%	-
	Total Reconnaissance		160995	5.690%		
	Access	FTP-Patator	7935	0.280%	50.150%	-
		SSH-Patator	5897	0.210%	37.270%	97 to 100
		Bot	1956	0.069%	12.353%	79 to 96
		Infiltration	36	0.001%	0.228%	0 to 50
	Total Access		15824	0.560%		
	DoS	DoS Hulk	230124	8.140%	60.614%	-
		DDoS	128025	4.530%	33.706%	98 to 100
		DoS GoldenEye	10293	0.360%	2.711%	99 to 100
		DoS slowloris	5796	0.210%	1.526%	99 to 100
		DoS Slowhttptest	5499	0.190%	1.448%	96 to 100
	Total DoS		379737	13.430%		
	Total Intrusion		470365	16.630%		
	Total		2827876	100.000%		
UNSW-NB15	Benign		93000	36.092%		
	Reconnaissance	Reconnaissance	13987	5.428%	1.000%	-
	Total Reconnaissance		13987	5.428%		
	Access	Exploits	44525	17.280%	61.173%	84 to 89
		Fuzzers	24246	9.410%	33.312%	44 to 51
		Backdoor	2329	0.904%	3.200%	1 to 6
		Shellcode	1511	0.586%	2.076%	-
		Worms	174	0.068%	0.239%	11 to 12
	Total Access		72785	28.247%		
	DoS	DoS	16353	6.346%	1.000%	-
	Total DoS		16353	6.346%		
	Malware	Generic	58871	22.847%	95.651%	97 to 99
		Analysis	2677	1.039%	4.349%	0 to 3
	Total Malware		61548	23.886%		
	Total Intrusion		164673	63.908%		
	Total		257673	100.000%		

TABLE 18. Intrusion-specific recall performance compared with previous studies. Columns 'CM-C' and 'CM-U' represent the classification performance of the 3-step architecture on the CM-CIC-IDS2017 and the CM-UNSW-NB15 datasets. Column '[38]' represents Nguyen et al.'s classification performance on the CIC-IDS2017 dataset, while column '[7]' represents Yang et al.'s performance on the UNSW-NB15 dataset. This table demonstrates that certain intrusion types exhibit low recall rates not only in our study but in other similar research as well.

Intrusion Type	CM-C	CM-U	[38]	[7]
Benign	99	**100**	**100**	83
DoS Slowhttptest	**100**	**100**	96	
PortScan	**100**	**100**	99	
DDoS	**100**	**100**	98	
DoS slowloris	**100**	98	99	
DoS GoldenEye	**100**	99	99	
Dos Hulk	**100**	97	**100**	
FTP-Patator	**100**	**100**	**100**	
SSH-Patator	**100**	**100**	97	
Web Attack-Brute Force	**99**	**99**	78	
Bot	**96**	73	79	
Web Attack-XSS	3	5	**33**	
Infiltration	50	**72**	0	
Heartbleed	**100**	83	**100**	
Web Attack-Sql Injection	0	0	0	
Generic	80	**99**		97
Exploits	**99**	89		84
Fuzzers	20	**51**		44
DoS	0	4		**24**
Reconnaissance	34	**77**		**77**
Analysis	0	**3**		0
Backdoor	0	**6**		1
Shellcode	16	22		**39**
Worms	0	**12**		11

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 described herein contrasts with the non-hierarchical approach of batch classification by a 1-step model for all network traffic, which results in an overall increase in system efficiency. Filtering benign data at Layer-1 reduces the amount of data that subsequent layers need to process, instead allowing subsequent layers 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, thus 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 various types of intelligent intrusions.

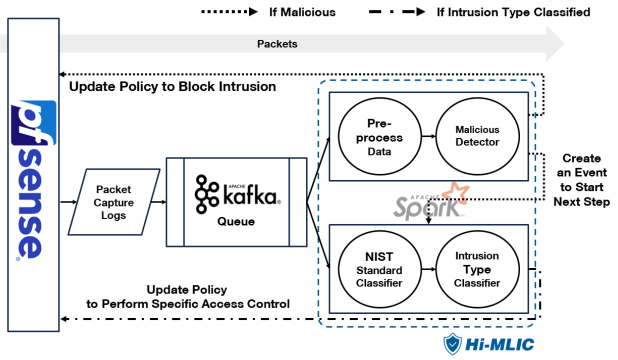

FIGURE 6. Proposed solution for integrating Hi-MLIC into a real-time system.

C. CONSIDERING DATA CHARACTERISTICS FOR FEATURE SELECTION

The results of the feature selection method presented in Section III-C are summarized in Section IV-C. Although 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, while the CM-UNSW-NB15 dataset includes categorical features. Some features do not follow a normal distribution that is centered around a mean value. However, we select the mean value as the representative value for features in Sections III-C2 and III-C3. Without considering their distributions, they may inadequately capture the characteristics of the features.

For future research, an advanced feature selection methodology should be developed 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 could lead to the design of a model that is capable of rapid classification using fewer 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.

D. APPLICATION OF HI-MLIC IN REAL-TIME SYSTEMS

We propose a solution to integrate Hi-MLIC into real-time systems while focusing on efficient data handling and rapid intrusion response. As shown in Figure 6, this solution is structured into three main stages: data capture, hierarchical intrusion detection, and automated access control.

Network traffic data is extensive and requires real-time processing. Intrusions can generate particularly large volumes of traffic and therefore cause disruptions. This poses challenges for systems that rely on event-based architectures and asynchronous processing. To address this, we propose the use of pfSense [39], an open-source firewall that is suitable for providing automated access control, to capture network packets in the first stage of the intrusion prevention system. We also suggest using Apache Kafka [40], which is a distributed streaming platform that allows efficient processing of large-scale data streams, to preprocess the traffic, thus making it possible to manage large-scale traffic.

Hi-MLIC employs a 3-step hierarchical approach to enhance classification accuracy and processing speed, thereby effectively managing the balance between benign and malicious data. Layer-1 differentiates between benign and malicious traffic by quickly detecting and blocking malicious traffic to prevent immediate threats. Layer-2 further classifies the blocked malicious traffic into four broad categories, while Layer-3 identifies 23 specific intrusion types within these broader categories. This detailed classification allows for the implementation of appropriate access policies based on the specific nature of the threat. This structured approach improves detection accuracy and speed, making it suitable for use in real-time systems where quick and precise classification is essential. To serve our model, we suggest using Spark [41], which supports machine learning libraries and is efficiently designed to implement Hi-MLIC, where multiple models are integrated. Its compatibility with Apache Kafka also ensures seamless data preprocessing and model utilization.

Upon detecting malicious traffic, we propose the automatic application of specific access control policies using pfSense. These policies, which can be either host-based or network-based, ensure that identified threats will be mitigated quickly and effectively. Host-based access control operates at the application level, managing security for individual systems, while network-based access control analyzes and controls access at the network and transport layers. This automated response mechanism minimizes the potential for human error and delays in response, thereby enhancing overall system security.

By implementing these three stages, it is possible for Hi-MLIC to be effectively deployed in real-time environments. Such integration is expected to facilitate robust and adaptive security measures, thus protecting against a wide range of network intrusions.

VI. CONCLUSION

In this study, we have proposed Hi-MLIC, which is a hierarchical multi-layer 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 have been constructed by consolidating two benchmark datasets. Following this data integration, experiments have been conducted to determine the suitable data format

for network intrusion detection, which also results in the acquisition of more types of intrusion scenarios.

As the two datasets are consolidated, a problem of data imbalance emerges. The hierarchical multilayer approach has been introduced to reduce misclassification rates resulting from data imbalance. This approach shows strong performance over the non-hierarchical approach, as it achieves a recall rate of up to 98.81%. Through this comparison, we confirm that the model can be trained to address data imbalance by hierarchically adding the layers, thereby enhancing the ease of classification.

Feature selection is a necessary aspect of creating 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 has allowed the model to be lighter while maintaining high performance. In the CM-CIC-IDS2017 dataset in particular, which consists of numeric features, a significant performance improvement is observed. Overall, our model achieves excellent accuracy of 98.81%, precision of 98.79%, recall of 98.81%, and F1 score of 98.68%, as presented in Table 15. Ultimately, these results show that the Hi-MLIC model can effectively detect and classify various intrusion types, thus showcasing its potential to respond effectively to intrusions within a network, even when diverse response strategies come to be applied in the future.

ABBREVIATIONS

The following abbreviations are used in this manuscript:

Aboost	AdaBoost Classifier.
AD	Anomaly-based Detection.
CG	Correct Group.
CM	Consolidated and Merged.
DL	Deep Learning.
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.

ACKNOWLEDGMENT

(Yunji Kim and Jihyeon Kim are co-first authors.)
Dataset access: https://github.com/CSID-DGU/Hi-MLIC

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," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 5, no. 3, pp. 72–81, 2020.

[2] R. Hofstede, P. Celeda, 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 Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 4th Quart., 2014.

[3] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 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)," in *Proc. Military Commun. Inf. Syst. Conf.*, 2015, pp. 1–6.

[5] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019.

[6] K. Aslansefat, "Safeml: Safety monitoring of machine learning classifiers through statistical difference measures," in *Proc. Int. Symp. Model-Based Saf. Assessment*, 2020, pp. 1–20.

[7] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Appl. Sci.*, vol. 9, no. 2, p. 238, Jan. 2019.

[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," *Proc. Inst. Syst. Program. 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. 154156–154171, 2021.

[10] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.

[11] S. Bhatia, A. Jain, P. Li, R. Kumar, and B. Hooi, "MStream: Fast anomaly detection in multi-aspect streams," in *Proc. Web Conf.*, Apr. 2021, pp. 1–26.

[12] 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, p. 867, Mar. 2022.

[13] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Appl. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106301.

[14] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electron. Markets*, vol. 31, no. 3, pp. 685–695, Sep. 2021, doi: 10.1007/S12525-021-00475-2.

[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, Feb. 2023.

[16] M. Verkerken, L. D'hooge, D. Sudyana, Y.-D. Lin, T. Wauters, B. Volckaert, and F. D. Turck, "A novel multi-stage approach for hierarchical intrusion detection," *IEEE Trans. Netw. Service Manage.*, vol. 1, no. 1, pp. 1–16, Jun. 2023.

[17] H. Ezzat Ibrahim, S. M. Badr, and M. A. Shaheen, "Adaptive layered approach using machine learning techniques with gain ratio for intrusion detection systems," 2012, *arXiv:1210.7650*.

[18] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2017.
[19] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.
[20] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *J. Big Data*, vol. 7, no. 1, pp. 1–20, Dec. 2020.
[21] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Syst. Appl.*, vol. 148, Jun. 2020, Art. no. 113249.
[22] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *Proc. IEEE 26th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2017, pp. 1881–1886.
[23] A. H. Lashkari. (2017). *Cicflowmeter*. Accessed: Aug. 10, 2021. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt
[24] R. Blank and P. Gallagher, "Nist special publication 800–30 revision 1 guide for conducting risk assessments," *Nat. Inst. Standards Technol.*, vol. 1, no. 1, pp. 1–16, 2012.
[25] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
[26] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[27] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Phil. Trans. R. Soc.*, vol. 53, pp. 370–418, Aug. 1763.
[28] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.
[29] S. Srivastava, M. R. Gupta, and B. A. Frigyik, "Bayesian quadratic discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, no. 6, pp. 1–20, 2007.
[30] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Stat. Soc. Ser. B, Stat. Methodology*, vol. 21, no. 1, pp. 238–240, Jan. 1959.
[31] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[32] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vols. IT-13, no. 1, pp. 21–27, Jul. 1967.
[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[34] F. Pedregosa and G. Varoquaux, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.
[35] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[36] 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, Jan. 2023.
[37] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, "Network intrusion detection model based on CNN and GRU," *Appl. Sci.*, vol. 12, no. 9, p. 4184, Apr. 2022.
[38] N. Nguyen Thi Thanh and Q. H. Nguyen, "Detection of abnormal network traffic using bidirectional long short-term memory," *Comput. Syst. Sci. Eng.*, vol. 46, no. 1, pp. 491–504, 2023.
[39] Netgate. (2023). *PFSense: Open Source Security*. Accessed: Jul. 10, 2024. [Online]. Available: https://www.pfsense.org/
[40] Apache Softw. Found. (2023). *Apache Kafka: A Distrib. Streaming Platform*. Accessed: Jul. 10, 2024. [Online]. Available: https://kafka.apache.org/
[41] (2023). *Apache Spark: Unified Analytics Engine for Big Data*. Accessed: Jul. 10, 2024. [Online]. Available: https://spark.apache.org/

YUNJI KIM received the B.S. degree in information and communication engineering from Dongguk University, South Korea, in 2023, where she is currently pursuing the master's degree in artificial intelligence. Her research interests include artificial intelligence, machine learning, nework security, and reinforcement-learning.

JIHYEON KIM received the B.S. degree in computer science and engineering from Dongguk University, in 2024. Her research interests include networks, security, and artificial intelligence.

DONGHO KIM received the B.S. degree in computer engineering from Seoul National University, South Korea, in 1990, and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, California, USA, in 1992 and 2002, respectively. He is currently working as a Professor with the Software Education Institute, Dongguk University, South Korea. His research interests include artificial intelligence, distributed systems, networks, and security.