IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 3, 2022, accepted January 31, 2022, date of publication February 7, 2022, date of current version February 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149295

Machine Learning With Variational AutoEncoder for Imbalanced Datasets in Intrusion Detection

YING-DAR LIN[1], (Fellow, IEEE), ZI-QIANG LIU[1], REN-HUNG HWANG[1,2], (Senior Member, IEEE), VAN-LINH NGUYEN[2,3], (Member, IEEE), PO-CHING LIN[2], (Member, IEEE), AND YUAN-CHENG LAI[4]

[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu City 300, Taiwan
[2]Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi County 621, Taiwan
[3]Department of Information Technology, University of Information and Communication Technology, Thai Nguyen 25000, Vietnam
[4]Department of Information Management, National Taiwan University of Science and Technology, Taipei City 106, Taiwan

Corresponding author: Ren-Hung Hwang (rhhwang@cs.ccu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under Grant 110-2811-E-194-501-MY2, Grant MOST 109-2221-E-194-025-MY2, Grant 108-2221-E-194-022-MY3, and Grant 108-2221-E-194-019-MY3; and in part by the Advanced Institute of Manufacturing with High-Tech Innovations (AIM-HI) through the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

ABSTRACT As a result of the explosion of security attacks and the complexity of modern networks, machine learning (ML) has recently become the favored approach for intrusion detection systems (IDS). However, the ML approach usually faces three challenges: massive attack variants, imbalanced data issues, and appropriate data segmentation. Improper handling of the issues will significantly degrade ML performance, e.g., resulting in high false-negative and low recall rates. Despite many efforts have done in the literature, detecting security attacks in a complicated network environment with imperfect data collection is still an open issue. This work proposes a *machine learning* framework with a combination of a *variational autoencoder* and *multilayer perceptron* model to deal with imbalanced datasets and detect the explosion of attack variants on the Internet. The detection engine also includes an efficient *range-based sequential search* algorithm to address the segmentation challenge in data pre-processing from multiple sources (network packets, system/statistic logs) effectively. Our work is the first attempt to demonstrate the effect of using an appropriate combination of ML models for boosting IDS detection capability in a heterogeneous environment, where data collection imperfection is common. Experimental results on a public system log dataset (e.g., HDFS) show that our method gains approximately as much as 97% on F1 score and 98% on recall rate, a promising result compared to the same measurement of other solutions. Even better, we found that the proposed treatment of imbalanced datasets can improve up to 35% on the F1 score and 27% on recall rate. The testing results also indicate that our model can detect new attack variants.

INDEX TERMS Imbalanced dataset, machine learning, variational autoencoder, intrusion detection.

I. INTRODUCTION

Zero-day vulnerabilities have been the headache of security protection systems for decades, particularly in susceptible networks. Exploiting such flaws, an attacker can intrude the enterprise organizations to steal trade secrets, sabotage the infrastructure, and even disrupt the operating services [1]. When the Internet gets more complicated, network intrusion becomes a cannot-be-ignored threat, for example, by attacking weak nodes or exploiting software bugs [2], [3]. To address the challenges, a common approach is to use firewalls and intrusion detection systems (IDS). Many traditional IDSs have heavily relied on signature-based detection approaches, such as using a fixed list of pre-defined pattern rules. If an access behavior matches the rules, the configured IDS will mark it as an attack and block the access. However, if the attacks come from a zero-day vulnerability, this kind of IDS will likely fail to detect them. Another common solution is to use anomaly detection with various baselines [4]. For example, the detection engines can find out the abnormal behavior by checking whether the traffic pattern is far from a defined ''normal'' profile. However, this approach comes with a cost of high false-positive rate.

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

To address the challenges of the negative performance in the abnormal-based intrusion detection, many researchers have aggressively pursued a way of applying Machine Learning (ML) and Deep Learning (DL) techniques [1], [5], [6]. Essentially, there are two common models for ML/DL-based IDS: unsupervised learning and supervised learning. Of the two methods, the unsupervised learning-based approach promises to be the key player for future IDSs due to its strength in addressing the explosion of security attack variants [7]. For example, the authors in [8] propose to map the original data to a hyper-sphere and then calculate the distance between each point to find an abnormal point. On the other hand, Garchery *et al.* [9] adopted multiple unsupervised learning models to assign a score to every data point. Any activity is then marked as an anomaly if its score is higher than a set threshold. Tang *et al.* [10] proposed a similar method but used a decoder-encoder network to calculate scores. A typical advantage of the decoder-encoder approach is that it does not rely on labeling the dataset and thus is able to cope with unknown data, such as zero-day attacks and new attack variants. On another work, the authors in [11] propose six machine-learning-based IDSs by using K Nearest Neighbor, Random Forest, Gradient Boosting, Adaboost, Decision Tree, and Linear Discriminant Analysis algorithms. Their system are tested on CSE-CIC-IDS2018, an imbalanced dataset. According to the authors' claim, the imbalance ratio can be reduced by using a synthetic data generation model called Synthetic Minority Oversampling TEchnique (SMOTE). However, the algorithms are mostly based on conventional machine learning models. Mezina *et al.* [12] also propose a LSTM-and-CNN based IDS that can achieve accuracy 92% and 97% on the KDD99 and the CSE-CIC-IDS2018, respectively.

However, dataset issues are the Achilles heel of ML-based IDS. Datasets need to be cleaned and validated before training with ML; otherwise, ML-based models' detection performance will be even worse than conventional approaches. In general, data sources can be classified into two types, system logs and network traffic. Most state-of-the-art solutions [1], [13]–[16] are in favor of using system logs for host-based IDS while using packet traffic for network-based IDS. Unfortunately, the features extracted from packet headers cannot reveal information for certain attacks, such as malware spreading. System logs can be more meaningful in some scenarios than using network traffic. For example, crypto-mining attacks can be traced by monitoring the abnormal CPU usage (in normal PC) in the resource usage logs or a massive number of communications with other peer networks. Moreover, given the popularity of secure networks and webs (e.g., using HTTPS), it is a challenge to analyze the encrypted traffic payloads [17].

This work presents a prospective method to address three issues of the dataset collection and pre-processing procedures for IDS in the literature: massive attack variants, imbalanced data issues, and appropriate data segmentation [18], [19]. Most existing studies focused on a single dataset, e.g., network traffic. By contrast, our system can deal with a combination of multiple data sources (i.e., system logs and network traffic). Also, imbalanced datasets and improper data segmentation are the main factors contributing to the loss of detection accuracy, particularly for DL-based IDSs. Several state-of-the-art methods [20], [21] have attempted to use two separate architectures in dealing with the imbalanced dataset problem and abnormal detection, respectively. Our system utilizes a variational autoencoder to address these two issues simultaneously.

A. CONTRIBUTIONS

The main contributions of this work are summarized as follows.

- We propose a machine learning framework to detect security variants in heterogeneous networks. The framework is equipped with Range-based Sequential Search (RSS), an efficient algorithm to deal with the segmentation challenge of data pre-processing from multiple sources. RSS can determine the best format (sequence length) for data segmentation.
- We build a combination of two learning models (variational autoencoder, multilayer perceptron) to address imbalanced datasets and attack detection simultaneously instead of separating them as most existing architectures did. The evaluation results indicate that our system architecture gives competitive detection performance compared with the state-of-the-art studies.
- Our work is the first attempt to demonstrate the effect of using an appropriate combination of ML models to boost the IDS detection capability on a multi-source dataset. We release the source codes for further research in the community.[1]

B. ORGANIZATION OF THE PAPER

The remainder of this paper is organized as follows. Section II introduces the background and the related work for anomaly intrusion detection. In Section III describes the key phases of our ML-based intrusion detection and the model architecture design. Sections IV and Section V present the implementation steps and experiment results. Finally, Section VI gives our conclusion and hints for future work.

II. BACKGROUND AND RELATED WORK

This section discusses the related work on ML-based IDS methods and several typical approaches to overcome the data imbalanced issues in training and evaluation. In the end, various comparisons on the features of our method with that of previous works are also presented.

A. MULTIPLE DATA SOURCES & IMBALANCED DATASET ISSUES

Machine learning requires understandable input data for training and testing. Unfortunately, each data source favors

[1]The datasets and source codes are available at https://bit.ly/hybridvaeids

TABLE 1. Machine learning host-based anomaly/intrusion detection approaches.

Paper	Dataset	Sequence Length	Solution for Imbalanced Dataset			Capability of Similar Unknown Abnormal Detection	Machine learning model	Reuse Model Architecture
			Under/Over Sampling	Generate Synthetic Sampling	Cost-Sensitive Classification			
[24]	SINET4	Network system log length: 1 day	X	X	X	N/A	Conditional Variational Autoencoder	V
[25]	HDFS, Openstack	fixed length (default: 10)	X	X	X	Limited	LSTM	X
[26]	HDFS, Yahoo network traffic	N/A	X	X	X	Yes	LSTM and autoencoder	X
[27]	HDFS, BGL	fixed length (default: 10)	X	X	X	Limited	Stacked LSTM	X
[20]	HDFS, BGL	fixed length (default: 3)	X	X	X	Yes	GAN networks	X
[21]	BGL, Openstack	N/A	X	V	X	Yes	Sequence GAN	X
Ours	HDFS	fixed length 5~15	X	V	V	Yes	Variational autoencoder & MLP	V

X: Not supported; V: Supported

a different format and structure for storage. System logs have a more diverse structure compared to network packets. The challenge is to support the learning of these multiple sources simultaneously due to the difference in their data type and format. In this work, our IDS supports data from both system logs and network packets during the training. The data are grouped into batches by frequency or time. For network traffic formatting, grouping is done by using log sequence length and time window. Similarly, a fixed-length is used in structuring system logs.

Imbalanced datasets are another challenge for ML/DL-based IDS. A dataset is imbalanced if there is a significant disproportion among the classes or samples inside the dataset. For example, data collection usually consists of much data from legitimate activities but a few from the attacks. Similarly, datasets that include the attack data only are not uncommon [22]. Relying on such imbalanced datasets, an ML model can fail to accurately predict the minority class [23]. In the literature, generally, there are three common approaches to deal with the imbalanced dataset problem. The first one is to use the undersampling/oversampling technique, i.e., by deleting a proportion of majority data or duplicate minority data to balance each class's data points. The second method is to use synthetic sampling, which will clone data points of the minority class based on the original minority data. The final technique is to use the cost-sensitive classification, which calculates the ratio with the number of positive and negative data points and changes the weight for labels based on the ratio.

B. ML/DL-BASED INTRUSION DETECTION SYSTEMS

Currently, the state-of-the-art ML/DL IDS model can be categorized into two approaches: host-based and network-based. Host-based IDS often runs on the workstation or dedicated servers to monitor the internals of a computing system or the network packets on its network interfaces to detect security attacks and anomalies. The comparison of several typical methods on the first approach is detailed in Table 1. Generally, a host-based ML IDS can use one of two following learning techniques:

- Learning from legitimate activities *or* attack behaviors: the IDS will focus on learning from a single type of activity, and then any unknown object detected will be considered from the attacks. This method is suitable for simple environments that have few changes of services over time. Some notable studies of this approach are DeepLog [25], nLSALog [27], and unlearning techniques [26]. However, since the ML models are trained with benign traffic only, the imbalanced dataset issues are not often considered. When a new service is launched, the IDS system must run the training again; otherwise, it will incorrectly mark the traffic of the new service as abnormal.
- Learning from both legitimate *and* attack behaviors: this method is suitable for complicated environments. The difficulty is to build a good learning model. Since the system accepts both legitimate and attack data, its learning model needs to be able to deal with the imbalanced dataset issue. Farzad *et al.* [21] proposed to use a generative adversarial network (GAN) to carry out synthetic sampling, with the goal of enhancing data modeling. Several kinds of research of this approach are [20] and [24]. However, the remaining challenge is to integrate the imbalanced datasets solution (as presented in Section II-A) and the learning model as a hybrid process. Our learning model is designed to overcome this challenge.

For network-based ML IDS, Table 2 shows the difference between our IDS and various existing studies. Important lessons learned from the comparison are as follows. First, most of the early works relied on training the attack traffic only. For example, Zhao *et al.* [28] extracted network traffic and trained it for detecting a botnet attack by setting a time window of 300 seconds. Kirubavathi *et al.* [29] also trained their ML model using network traffic and suggested that a time window of 180 seconds could produce the best result. In another works, the authors in [31] and [11] proposed SMOTE, a DL-based IDS, to carry out synthetic attack samples to enrich the network traffic dataset, and thus, improve the detection accuracy. Second, previous studies did not

TABLE 2. Machine learning network-based anomaly/intrusion detection approaches.

Paper	Dataset	Sequence Length	Solution for Imbalanced Dataset			Capability of Similar Unknown Abnormal Detection	Machine learning model	Reuse Model Architecture
			Under/Over Sampling	Generate Synthetic Sampling	Cost-Sensitive Classification			
[28]	LBNL	300 seconds	X	X	X	Only for botnet detection	Nearest Neighbor & SVM	X
[29]	ISOT	60~300 seconds (best: 180 seconds)	X	X	X	Only for botnet detection	Naive Bayesian & SVM	X
[30]	CIDDS-001	Full flow	V	X	X	Yes	Variational autoencoder Stacking	X
[31]	CSE-CIC-IDS2018	Full flow	V	V	V	Yes	LSTM	X
[32]	UNSW-NB15 CICIDS2018	Full flow	V	V	V	Yes	Multi-Task Learning	X
[33]	IDS2017 ISIC2019	Full flow	V	V	V	Yes	1-D CNN	X
[34]	CSE-CIC-IDS2018	Full flow	V	V	V	Yes	Classical ML & SMOTE	X
Ours	III TTP 2020	15~90 seconds, (15 seconds per unit)	X	V	V	Yes	Variational autoencoder & MLP	V

X: Not supported; V: Supported

specifically address the impact of subflow sequence length and imbalanced data on training performance. The authors in [15], [16], [35] presented a variational autoencoder-based IDS. However, the system was trained using a single data-source dataset only. In this work, we provide a general and flexible solution to select the best sequence length so that users can define a custom profile according to their needs for the system inputs. Furthermore, our IDS solution works well with a multi-source dataset.

III. KEY PHASES OF OUR ML-BASED INTRUSION DETECTION DETECTION

A machine learning framework consists of three phases: input data pre-processing, ML model training, and ML model evaluation. Of these three, pre-processing is the first step for building well-structured data for training in the following phases. Its importance will increase with the complexity of multiple data sources, i.e., the challenge of accurately representing attack/abnormal behavior. This section overviews the study's notations and the details of our ML framework.

A. NOTATIONS

Table 3 defines the notations used in our work. A data point means a sequence of log records or packets. A dataset denotes a collection of data points with specified sequence length and data type. Specifically, a dataset is determined by the sequence length l and the data type k; k could be system log or network traffic. l is based on the log counts (for log dataset) or the time window (for traffic dataset). A data point consists of four attributes: sequence id i, information K_i^l (generated by the sequence of data), a corresponding ground truth label t_i^l, and a prediction result p_i^l from ML model. Class weight, class count, and ML algorithm are also important parameters to be defined. The class weight is vital for cost-sensitive classification. Suppose that w_p, w_n, $w_{p'}$ denotes the training weights for positive data, negative data and generated positive data, respectively. The class count consists of the number of positive (c_p), negative (c_n), and generated positive data ($c_{p'}$) in the training dataset. An ML algorithm ml denotes an unsupervised learning model M_u or a supervised learning model M_s.

B. KEY PHASES OF OUR MACHINE LEARNING BASED IDS

Figure 1 shows our ML-based IDS architecture with its relevant components. The first stage is to perform data pre-processing and data labeling. The outputs of this stage are multiple versions of labelled dataset based on different sequence lengths. After obtaining the datasets, the system runs a range-based sequential search (Section III-D) to find the best dataset instance along with the ML parameters.

C. DATA PRE-PROCESSING

To minimize false-negative rate in the prediction result, i.e., (p_i^l, t_i^l), the data preprocessing plays a crucial role. We separate this process into two tasks: data segmentation and imbalanced data fixing. A data segment denotes a data point or a sequence of log records/packets, i.e., l. For different data sources, an evaluation is required to determine the best value of l that contributes the best performance for the ML model. With regard to the imbalanced dataset problem, there are two possible cases. The first one is that the number of abnormal data segments is much less than the number of legitimate segments. The other is the attack data segments dominate the legitimate ones. Given the existence of multiple labeled datasets with the same log type k but various sequence length l, the goal is to determine the best configuration of the sequence length l for the architecture M^l. Labeling data in the datasets accurately is another challenge. The complexity soars if the system must process a mixed dataset collected from many nodes. The pre-processing and labeling flow for each data type are as follows.

1) DATA PRE-PROCESSING FOR SYSTEM LOGS

After collection, the system logs are sliced by the length sequence l. Figure 2 shows an example of the cutting method with sequence length $l = 4$. As shown in the figure, each

TABLE 3. Notations used in this work.

Category	Name	Notation	Note
Datasets	Sequence Length	l	length for each data point
	Data Type	k	log: log template, traffic: packet
Data Points	Data Sequence ID	i	
	Data Sequence Information	$K_i^l = [k_i^1, k_i^2, k_i^3, ..., k_i^l]$	generated by the sequence of data
	Ground Truth	t_i^l	positive: 1, negative: 0
	Prediction	p_i^l	positive: 1, negative: 0
Machine Learning	Class Weight	$w_p, w_n, w_{p'}$	positive, negative, generated positive
	Class Count	$c_p, c_n, c_{p'}$	positive, negative, generated positive
	ML Algorithm	ml	Variational AutoEncoder
	ML Model	$M^l = ml(l) = M_u^l + M_s^l$	M_u: unsupervised, M_s: supervised

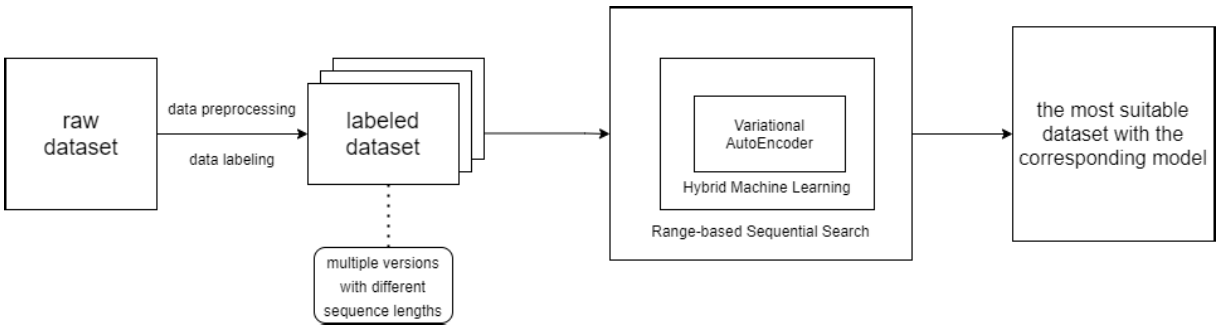

FIGURE 1. The illustration of our intrusion detection system architecture and relevant components.

FIGURE 2. An example of the cutting method for system logs.

square is a log, and the number inside the square is the log index. The rule defines the label for each sequence as follows. If there is one or more log within the sequence which is abnormal behavior, the sequence will be labeled as abnormal; otherwise, it is marked as normal. The cutting procedure on the dataset with several values of l aims to create multiple copies of the dataset but with different distributions or attack/benign data proportions.

2) DATA PRE-PROCESSING FOR NETWORK TRAFFIC

For network traffic, a bi-directional flow is identified by using five tuples (source IP, destination IP, source port, destination port, and protocol id). The sequence length of a flow is defined as a time window. A flow can be split into sub-flows by the same time frame. Multiple traffic datasets are obtained by setting a variant of l. For labeling, if a flow is marked as abnormal, its sub-flows have the same type.

D. RANGE-BASED SEQUENTIAL SEARCH

Finding the best value of l is essential for determining the best configuration for an ML model. After splitting the raw dataset into multiple instances with various configurations of the sequence length l, each instance of the dataset with a specific value of l will be trained and tested. The goal is to figure out which sequence length gives the best performance. We name this search procedure as "Range-based Sequential Search" (RSS). RSS is performed by increasing the sequence length sequentially in a selected range, e.g., [4 . . . 10]. The proposed ML is trained and tested sequentially within this range for each kind of dataset. After evaluation, the sequence length with the highest F1 score is the selected.

Figure 3 is the flow chart of the RSS. In this example, the RSS starts from each dataset with $l = 5$. The dataset (K_i^l, t_i^l) is feed into the ML model with variational autoencoder (see the next section). During the training, the predicted output p_i^l and the ground truth t_i^l are used to calculate the loss and update the parameters of the ML model. An F1 score is obtained from the testing dataset for each value of l. The RSS records the dataset that yields the best F1 score and repeats the procedure until the maximum sequence length is reached. Finally, the RSS returns the dataset with the sequence length that yielded the best F1 score.

E. MACHINE LEARNING MODEL FOR INTRUSION DETECTION

The learning model is the core of an IDS system. Figure 4 shows the architecture of our learning model. The learning process is a hybrid process which combines an unsupervised learning (variational autoencoder) M_u with a supervised learning (multiplelayer perception) model M_s. A dataset (K_i^l, t_i^l) is marked as imbalanced if most of data in K_i^l are legitimate or attacks only. As a countermeasure, the model M_u takes the lead to generate positive data points (minority class) and then append them to the original dataset to balance the data. The generated synthetic dataset is then trained with

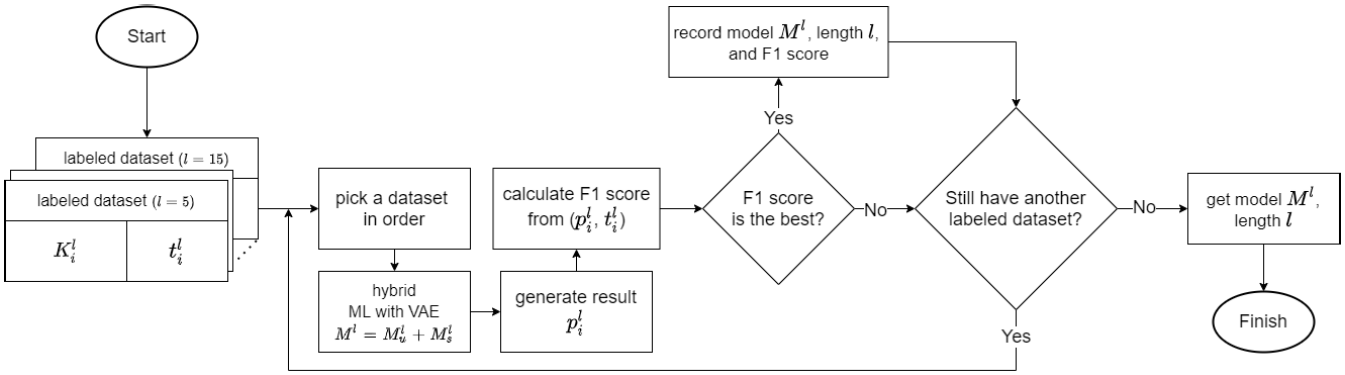

FIGURE 3. The illustration of the Range-based Sequential Search to figure out the best value of the sequence length *l* for a ML model M^l.

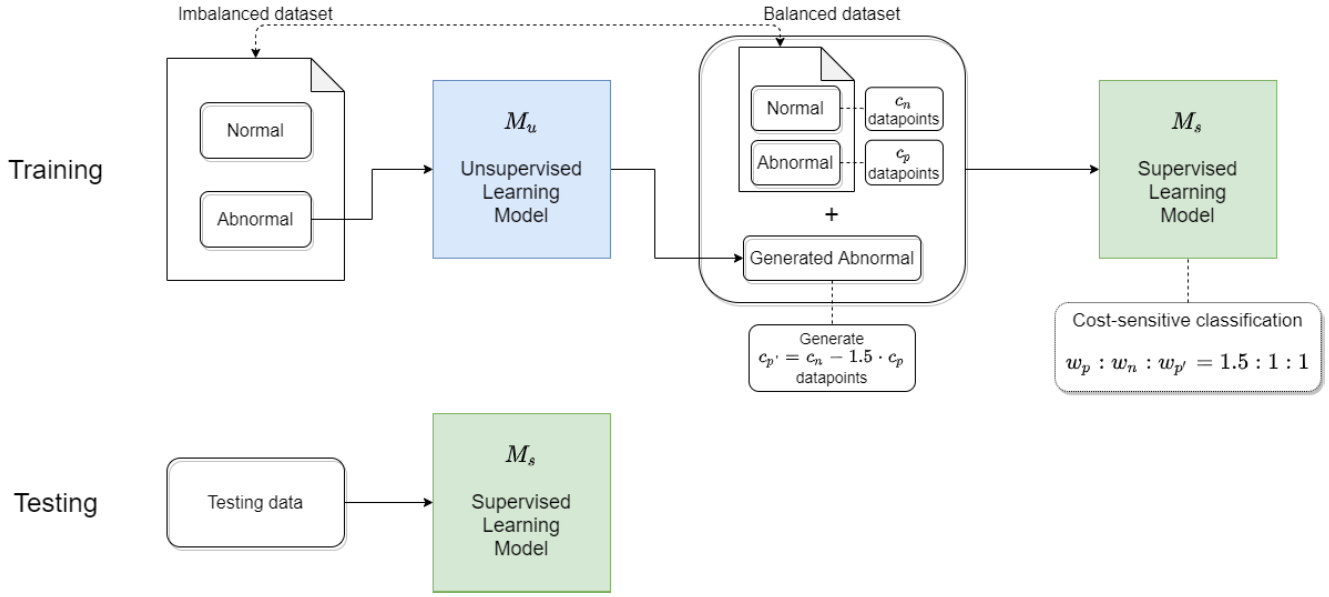

FIGURE 4. The illustration of our proposed machine learning architecture for intrusion detection in the training and testing stage.

the M_s model. However, due to the uncertainty of the quality of the generated data points, the weights of the synthetic data are allocated a lower value than those of the original data [31], e.g., training weight ratio could be $w_p : w_n : w_{p'} = 1.5 : 1 : 1$ where $w_{p'}$ is the weight for generated positive data points. To overcome the imbalanced dataset issue, a key issue is to determine the additional number of positive data points that need to be generated. For this problem, we propose to use the estimation $c_{p'} = c_n - 1.5 * c_p$. This estimation is to make sure the total weight for each class in the training phase (positive/negative) is the same.

1) UNSUPERVISED LEARNING WITH VARIATIONAL AUTOENCODER

The unsupervised learning model M_u used in this work is a variational autoencoder, which was first invented by [36]. Figure 5 shows the architecture of this model. An advantage of using the variational autoencoder is that it can generate positive data points. Also, an important feature of using variational autoencoder is the data distribution can be changed after applying the model. Original data with normal distribution can be transformed into Gaussian distribution after variational autoencoder training. As a result, we can generate a number $c_{p'}$ of the positive data points by feeding a number $c_{p'}$ of Gaussian noise samples. The generated positive data points are then merged into the original ones.

2) SUPERVISED LEARNING WITH MULTILAYER PERCEPTRON

Figure 6 shows the architecture of the supervised learning model M_s. In this architecture, the encoder from the unsupervised learning model is integrated into the supervised learning model to support transfer learning. Since the encoder learns an abnormal representation from M_u, it can help the system to classify the normal and abnormal behavior more accurately. Another advantage of the integration is that the system does not need to train again in the encoder part of M_s again.

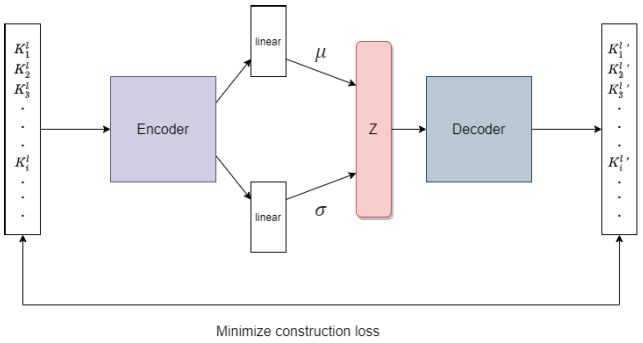

FIGURE 5. The illustration of our unsupervised learning model with variational autoencoder.

We can then finish the evaluation by testing the processed data (from M_u) on M_s only.

IV. IMPLEMENTATION

This section introduces the tools used for implementation and the procedure for building a framework for evaluation. Furthermore, the log/traffic dataset visualization and the lessons learned from the results are also justified.

A. TOOLS USED FOR THE IMPLEMENTATION

The tools for system implementation are listed in Table 4. Spell [37] is used to parse the unstructured system logs to the given format, by including timestamp, log template, and parameter value. After parsing, the logs are grouped according to the range of the sequence length l. For network traffic, we use the tool *argus* for parsing the *pcap* file. *Argus* has two parts: argus server (argus) and argus client (ra). Argus server can translate the *pcap* file into the particular subflow format and save it to a file. By contrast, the *argus* client helps to transfer the features from the file to a CSV file. Note that we can specify how frequently the *argus* server reports and writes the sub-flow information to the file. Theoretically, the duration of each sub-flow data point can be set in the *argus* options. Spell and argus are the tools to create the datasets for system logs and network traffic with different values for the sequence length l. For the machine learning module, we use Keras processing and validating the input data, and for outputs, we use Scikit-learn.[2]

B. ML MODEL IMPLEMENTATION

Figure 7 illustrates the detail of the core layers of the ML models, including the encoder, the decoder, and the MLP model. These layers are designed to balance between the system performance and the complexity, as a result of using many hidden layers. For training, we use the original imbalanced dataset and only select the positive data points (e.g., attack behaviors) as the input of variational autoencoder (as shown in Figure 7a) and Figure 7b). After training, the decoder module is used to generate an amount of $c_{p'}$ of synthetic abnormal samples and then append them to the original dataset. The target is to balance the data proportion of the minority class. In all these procedures, the choice of the activation function is critical. Since the encoder and decoder only learn positive values on data points, we choose LeakyReLU as the activation function instead of ReLU.

The final step is to use the encoder and a Multilayer perceptron (MLP) model (Figure 7c) to run the intrusion detection. The models are trained with the balanced dataset from the previous step with ground truth labels. After the training, the model is evaluated with the testing dataset. The evaluation metrics include F1 score, precision, and recall. The training and testing processes are performed on all the generated datasets with various values of l. Recall that the RSS algorithm outputs the sequence length and dataset set that yields the highest F1 score.

C. TESTBED

We used an HDFS dataset [38] and the TTP dataset from an industry institute [39] as the system log dataset and network traffic dataset, respectively.

1) INPUT DATASET FOR SYSTEM LOGS

The HDFS dataset is a public dataset made by [38], and the complete raw dataset is available from Loghub [40]. This dataset is generated in a private cloud environment. The authors labeled the dataset by block. Figure 8 shows an example of the features of a log sequence. If we consider a log sequence with a length of 5, each number in the tuple (4, 2, 2, 35, 36) means a log template id, which we can get after using Spell to parse the raw logs. As illustrated in Figure 8, there are four main sets of information. The first part is ''frequency'', which denotes how many times a log template id appears in the sequence. The second is the order of log template id, i.e., the sequence in five log templates. The third is the time delta in a log sequence, which is the time difference between two log templates, and the final part is ''label''. Since logs are grouped with the same block id, we can label each log sequence with that index. If a block id belongs to abnormal, we label it with the value ''1''; otherwise, the value is set to ''0''. There are a total of 11 dataset instances created from different values of the sequence length l ($l = [5 \ldots 15]$).

2) INPUT DATASET FOR NETWORK TRAFFIC

The TTP dataset is a private dataset with labeled information from an industry institute [39]. The dataset is collected by applying three kinds of attack scenarios (cyber, kill, chain) based on MITRE's ''ATT&CK Matrix for Enterprise''[3] to simulate an enterprise environment. During the attack time, network traffic was collected using the Wireshark tool, which then produced multiple *pcap* files. The dataset also included the ground truth information of the attacker and victim, such as IP address. As noted above, argus is used to extract flows

[2]https://scikit-learn.org/

[3]https://attack.mitre.org/

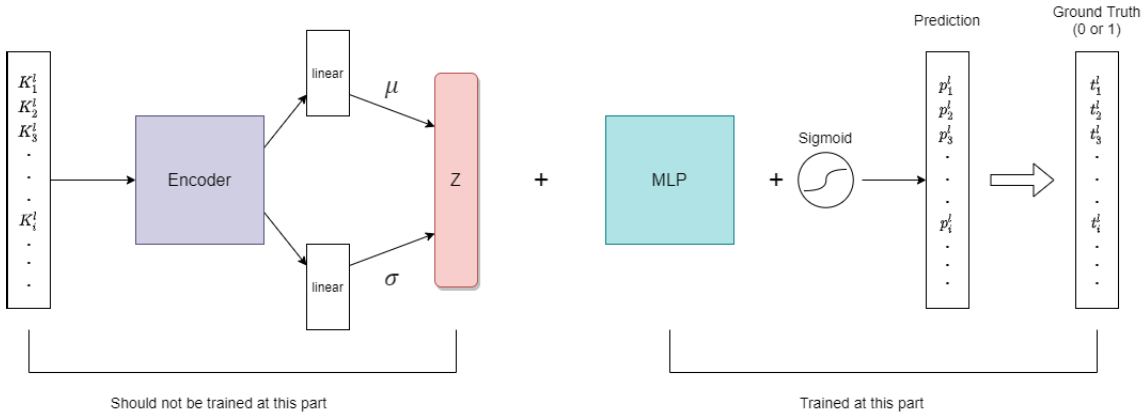

FIGURE 6. The illustration of the supervised learning model with multilayer perceptron.

TABLE 4. The experiment tools are used in this work.

Category	Name	Functionality
Data Extraction	Spell	Log parsing tool
	Argus	Pcap files extracting tool
Machine Learning	Keras	Build deep learning model
Preprocessing & Validation	Scikit-Learn	Split into training & testing dataset
		Data normalization
		Metrics calculation

and sub-flows from *pcap* files based on the 5-tuple information. Note that the collected traffic of a flow is bi-directional. Features of a flow (and sub-flow) are shown in Figure 9. From the findings of [41], the recommended duration for TCP and UDP flow in extracting features is 57.32 seconds and 10.72 seconds, respectively. Scaling the range, we use six-time different window sizes (15, 30, 45, 60, 75, 90 seconds).

3) DATA PRE-PROCESSING

Data cleaning and pre-processing are required before testing. Unlike prior work, our system aims to work with multiple data sources. Notably, the importance of a data source may be different in different attack scenarios. For example, if the attacker runs a malware intrusion, the network traffic is unlikely to have many clues about the activities of the malware. By contrast, the evidence of abnormal behaviors of the malware in a well-logged system is overwhelming. However, the same log sequence can be labeled inconsistently. We refer to this issue as the grey area phenomenon. Inconsistent labeling often makes it very hard for an ML model to learn the decision rules from the input data. In order to solve this problem, all of those data points are cut off from auto labeling and then post-checked by excluding the logs of the pre-defined benign software.

For data normalization, the minmax-scaler, a built-in function in the Scikit-learn package, is used to translate each column's value into a fixed range, e.g., [0, 30] for HDFS dataset and [0, 20] for the TTP dataset. Finally, the function ''train_test_split'' (inside the Scikit-learn package) is used to split each dataset into training and testing part.

V. EXPERIMENT RESULTS

This section describes the parameter settings for training two ML models, followed by performance evaluation and comparisons with previous works. Furthermore, the system performance of the proposal is highlighted by examining the proposed solutions of three main issues: imbalanced datasets, attack variant detection, and data segmentation.

A. PARAMETER CONFIGURATION

The experiments were performed on a computer with CPU Intel i7-4790K and GPU Nvidia RTX 2060. Table 5 lists the parameters of the system for the training. Note that this setting is suitable for both HDFS and TTP datasets. The encoder was set to non-trainable, i.e., to learn directly from the variational autoencoder-based model's output. Table 6 shows the configuration of the log/traffic data sets (described in Section IV-C).

B. PERFORMANCE COMPARISON WITH THE STATE-OF-THE-ART WORKS

Table 7 shows the system performance comparison of our system and several works on the HDFS dataset. Performance of DeepLog and LogGAN were reproduced from [25] and [20] respectively. The results in Table 7 indicate that our proposed solution outperforms most of the state-of-the-art works except the recall metric of ''Invariant Mining''. However, the ''Invariant Mining'' assessment only relied on abnormal data points. Furthermore, the results of ''DeepLog'' must rely on domain expert feedback that

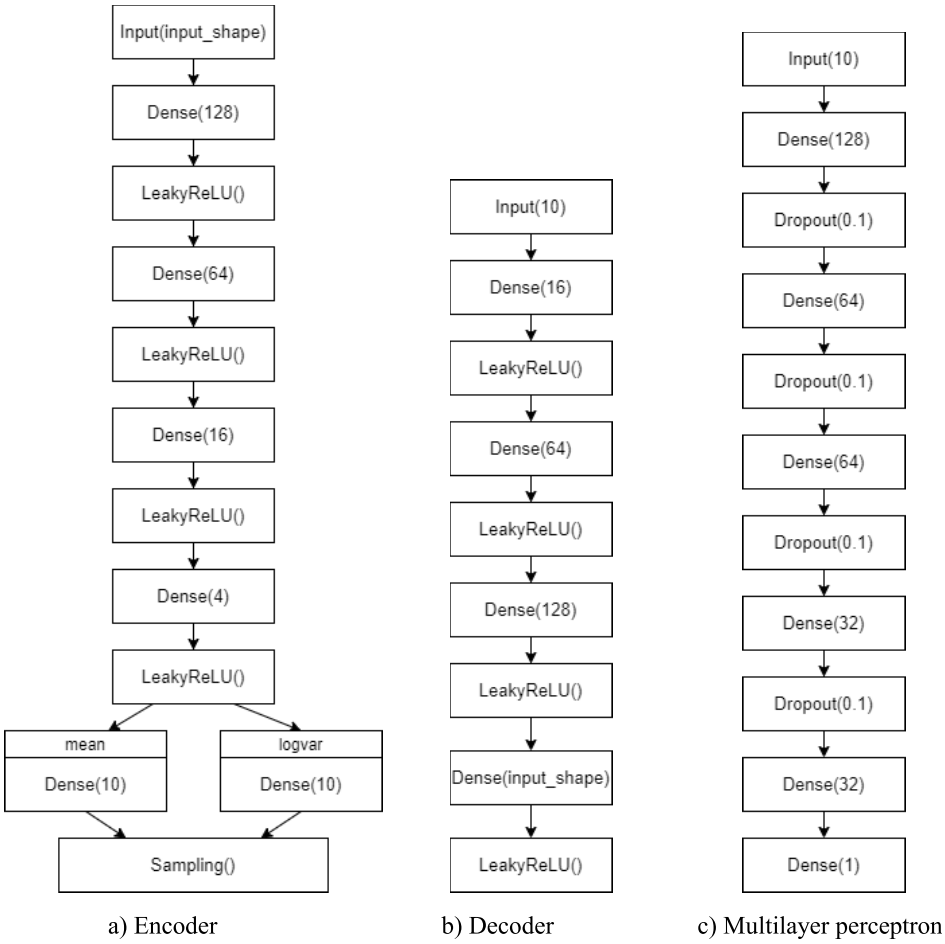

FIGURE 7. The illustration of the core layers and their configuration in our ML model.

Frequency							Order(Log Event Template)					Order(Timestamp_delta)					Label
1	2	3	4	⋯	35	36	1	2	3	4	5	1	2	3	4	5	Anomaly
0	2	0	1	⋯	1	1	4	2	2	35	36	ts_delta	ts_delta	ts_delta	ts_delta	ts_delta	0 or 1

FIGURE 8. HDFS dataset format.

5-tuple					Flow Information	Label
Protocol	SrcIP	SrcPort	DstIP	DstPort	rank stime flgs pkts bytes state ltime seq dur mean stddev	Anomaly
					sum min max spkts dpkts sbytes dbytes rate srate drate	0 or 1

FIGURE 9. TTP dataset format.

TABLE 5. The configuration used for our learning models.

Method	Architecture	Batch Size	Epochs	Sample Weight	Loss Function	Optimizer
Unsupervised	VAE	512	1000	N/A	KL divergence loss + Reconstruction loss(mse)	Adam(lr=0.01)
Supervised	(encoder)+MLP	256	100	$w_p : w_n : w_{p'} = 1.5 : 1 : 1$	Binary crossentropy	Adam(lr=0.003)

practically required intensive human interactions and labor. On the other hand, our solution yields a much higher recall rate than most of the previous works did. We also noted that several authors proposed to tackle the imbalanced data

TABLE 6. The training/testing data proportion for system log and network traffic datasets.

Dataset	Type	Sequence Length	Data Normalization	Train/Test Split
HDFS	log sequence	5~15 logs in sequence	0~30	90% training 10% testing
TTP	traffic subflow	15,30,45,60, 75,90 seconds	0~20	80% training 20% testing

TABLE 7. The system performance comparison of several intrusion detection systems in HDFS dataset.

Method	F1 score	Precision	Recall
Invariant Mining	0.154	0.084	1
PCA	0.465	0.707	0.346
DeepLog (from the result in LogGAN)	0.032	0.939	0.016
DeepLog (user feedback update)*	0.96*	0.95*	0.96*
iForest	0.482	1	0.318
LogClustering	0.532	1	0.362
LogGAN-sess	0.525	1	0.356
Ours	**0.8021**	**0.9263**	**0.7073**

*Domain expert required.

TABLE 8. Quantity of each class in training datasets.

Dataset	Normal	Abnormal (No balance)	Generated abnormal	Abnormal (Balanced)
HDFS	280852	12559	262013	274572
TTP1	46333	317	45857	46174
TTP2	46760	400	46160	46560
TTP3	66838	1296	64894	66190

problem, such as [11], [21], [32]–[34]. Unfortunately, none of these methods supports learning on multiple data sources. In summary, our proposed solution can achieve 0.80 F1 score and 0.71 on recall which outperforms the same measurements in the works noted.

C. THE INFLUENCE OF THE IMBALANCE IN DATASETS TO THE DETECTION PERFORMANCE

This section presents our system's performance in dealing with the imbalanced data problem. Table 8 shows the number of attack data points for each dataset classified by the variational autoencoder module. Each dataset is highly imbalanced; for example, the ratio of legitimate to attack data points in the HDFS dataset is 22:1, while that of the TTP dataset is up to 146:1. The highly imbalanced ratio impacts the performance of ML models significantly.

As shown in Table 9, our learning model performs well to detect the attack behavior, i.e., a higher recall and F1 score, even with the few appearances of the minority class. With the recall rate of the imbalanced dataset (45%∼61%) and that of the balanced dataset generated by VAE (70%∼97%), the results indicate that most of the attack data points have been predicted correctly. Note that, without a correct learning strategy, a class of data points that dominates in the dataset will assimilate the prediction results with its characteristics. Therefore, balancing a dataset plays a key role in enhancing the system performance of the ML-based IDS.

By comparing the recall rate of different datasets (shown in Table 9), we found that the system gives the lowest recall rate with the HDFS dataset. The rationale can be the result of the system logs having fewer features in training than those of network traffic. As shown in Figures 8 and Figure 9, the number of features of system logs is about four times less than that of network traffic. Due to the lack of relevant data representation (features) in system logs, the detection performance for various attack types becomes degraded. For example, the HDFS dataset yielded a higher False Negative (FN) rate, as shown in the last column of Table 9, given the same attack records in network traffic.

The limited number of features can potentially cause a grey area phenomenon. Table 10 shows the number of data points in the grey area in each dataset. In the HDFS dataset, more than 50% of data points fell into the grey area, which meant that the ground truth was inconsistent, even when the features were the same. Since these data points were deleted, the ML model has fewer data points for training, leading to poor detection than the TTP dataset.

D. ATTACK VARIANT DETECTION

Attack variants mean the attacks share major characteristics but are not entirely the same as each other. For example, two attacks come from the same tools and exploit the same techniques but use different parameter configurations to penetrate the victim. Detecting such attack variants is a challenge, even with state-of-the-art work [1]. So, if there are two kinds of attacks that belong to the same ATT&CK technique, it is important to know whether the proposed solution could detect either of the attacks by training the system with one of them only. In our experiments, the TTP datasets are especially suitable for testing the attack variants detection because they contain a broad range of attack variants. Since existing state-of-the-art works do not consider attack variant detection with a multi-source dataset, revising their architectures for evaluation is non-trivial. We leave this substantial task for future work.

Figure 10 shows the evaluation results of our attack variants detection for datasets generated by two groups; each group consisted of two similar attack types. The first group consisted of "Dark comet poison ivy" and "private backdoor script," while the second group consisted of "collect system information with meterpreter" and "ARP scanning." For each group, the first attack type was used for training, and the second attack type was used for testing. As shown in Figure 10, our proposed ML detects 100% of the attack variants for the first group and 82% for the second group, where the MLP could not detect any attack variants (0%). The super performance came from the fact that the encoder in our VAE module could automatically learn the attack behavior, even with few changes. Furthermore, synthetic samples generated by VAE also contributed significantly to the detection ability.

E. IMPACT OF DATA SEGMENTATION ON SYSTEM PERFORMANCE

Data segmentation, i.e., sequence length selection, is crucial for detection performance. Figure 11 shows the system

TABLE 9. Training datasets performance in comparison: Imbalanced dataset vs. Balanced dataset.

Dataset	Performance on Imbalanced Dataset							Performance on Balanced Dataset						
	F1 score	Precision	Recall	TP	TN	FP	FN	F1 score	Precision	Recall	TP	TN	FP	FN
HDFS dataset	0.6094	0.9429	0.4502	628	31168	38	767	0.8020	0.9267	0.7068	986	31128	78	409
TTP1 dataset	0.7189	0.8729	0.6110	3558	5322	518	2265	0.9462	0.9263	0.9670	5631	5392	448	192
TTP2 dataset	0.6283	0.9073	0.4805	2830	5611	289	3060	0.9678	0.9575	0.9783	5762	5644	256	128
TTP3 dataset	0.6290	0.9667	0.4662	3973	10301	137	4549	0.8855	0.8282	0.9514	8108	8756	1682	414

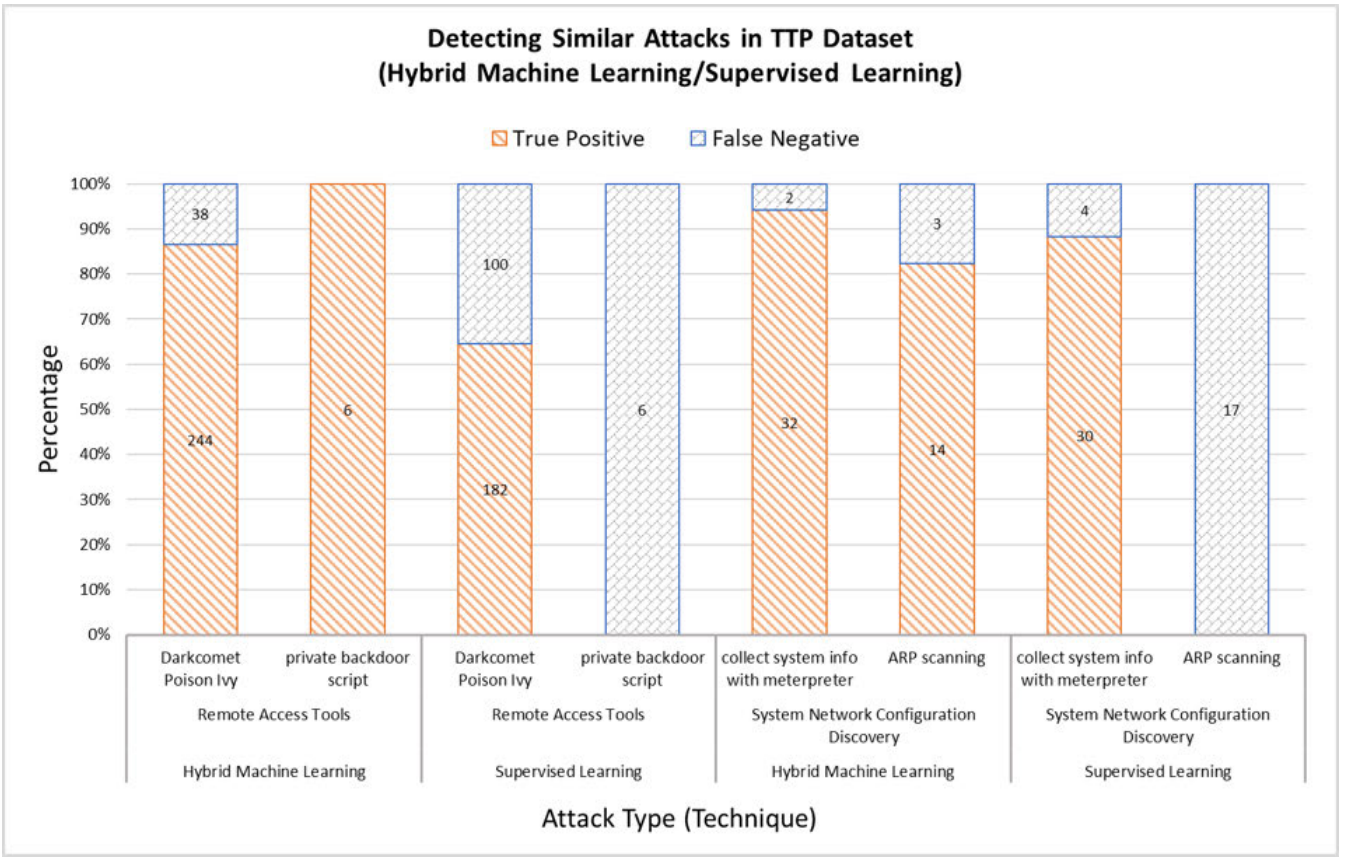

FIGURE 10. Detecting attack variants of the same ATT&CK technique.

TABLE 10. Grey area data in datasets.

DatasetDatapoints	All	Grey area	Available for ML
HDFS dataset	756805	430792	326013
TTP1 dataset	58313	0	58313
TTP2 dataset	58975	25	58950
TTP3 dataset	85308	140	85168

performance on the HDFS dataset under different sequence lengths. Figure 12 shows the corresponding results on the TTP datasets. The model can yield the best performance (0.8 on F1 score) when the sequence length is 10 ($l = 10$) on the HDFS dataset, i.e., ten continuous system log templates are grouped into a sequence. This result indicates that the recall score becomes irregular when the sequence length increases. The degradation hints that *a longer sequence length can introduce more noise, which is probably the characteristic of the other attacks*. Even better, similar to [42], our DL-based system can predict the abnormal behavior accurately by inspecting a small piece of data. For example, considering the "pass the ticket" attack in the TTP3 dataset, the attack activity may last for more than 10 hours; however, the attack can be detected by examining the data sequence of the first 30 seconds. By contrast, *a too-short value of l can make the system fail to conclude whether an activity is from an attack due to the lack of evidence*. Note that, in an imbalanced dataset, if the majority class is legitimate, the absence of attack patterns can lead to the wrong prediction

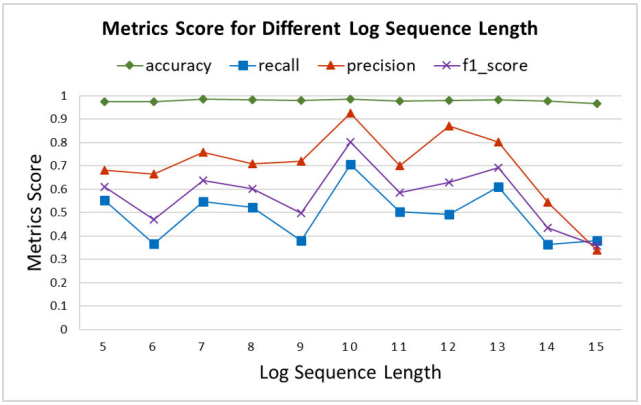

FIGURE 11. Impact of data segmentation on the system performance on the HDFS dataset.

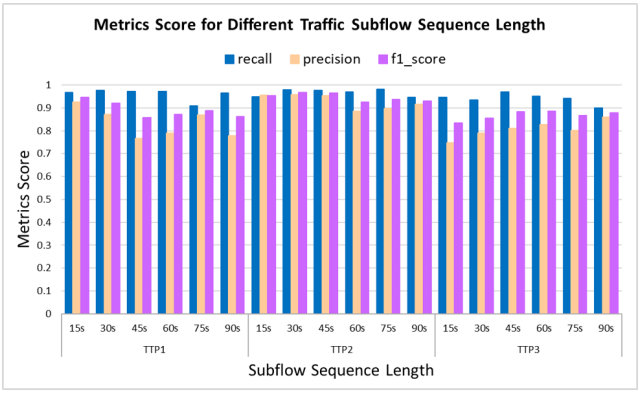

FIGURE 12. Impact of data segmentation on the system performance on the TTP dataset.

overall. Consequently, the precision gets worse when the sequence length becomes longer.

Figure 12 shows the impact of sequence length on the system performance on the TTP datasets, which consists of three attack scenarios. Each attack scenario contributes different attack characteristics. The best sequence length is thus different for each scenario. The best sequence length is 15 seconds for TTP1 (F1-score: 0.94), 30 seconds for TTP2 (F1-score: 0.96), and 60 seconds for TTP3 (F1-score: 0.88). This result again suggests that a longer sequence length does not always result in a good performance.

VI. CONCLUSION AND FUTURE WORK

In this work, we present a prospective ML-based framework to detect security attacks and their variants, even with imbalanced datasets in training. Our work aims to deal with three challenges: massive attack variants, imbalanced data issues, and effective data segmentation. The results demonstrate the advantages of the proposed machine learning model in enhancing the attack detection performance for the first and second goal, even with a limited number of samples of minority class or the existence of massive variants. Specifically, the experiment results on the HDFS dataset indicate that the system performance improvement can be up to 35% on the recall rate and 27% on the F1-score, compared with the state-of-the-art solutions. Besides, we found that the effectiveness of ranking datasets for ML-based IDS relies much on selecting correct segmentation. At best, data segmentation should be dynamically adjusted according to the specific environment. For this, we have proposed RSS, a range-based sequential search algorithm, to find out the best value of the sequence length l for data segmentation. Finally, our results also indicate that our model can detect new attack variants.

Despite the excellent performance in detecting security attacks and variants, there are several potential areas for further enhancement. First, a broad range of evaluating the sequence length for common datasets, regardless of types, can provide a better overview of their ranking in terms of effectiveness. The statistic can help the readers quickly identify a proper dataset and thereby contribute to security defense research. Second, the labeling for system log datasets is still an open topic. Since it is a challenge to deal with the grey area phenomenon in ML, a hybrid processing model, such as a combination with signature-based detection, may improve the system performance significantly. Although the system presents promising results in detecting attacks in heterogeneous networks, the training and testing still require substantial time. A compression or novel approach to shorten the time for inter-learning is also worthy. Finally, causal ML-based models to explain where and what causes the attacks are the potential targets, given the benefits of learning on multiple data sources.

REFERENCES

[1] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Syst. J.*, vol. 15, no. 2, pp. 1717–1731, Jun. 2021.

[2] S. Gatlan. (2021). *Mirai Botnet Variants Targeting New Processors and Architectures*. Accessed: Dec. 25, 2021. [Online]. Available: https://www.bleepingcomputer.com/news/security/mirai-botnet-variants-targeting-new-processors-and-architectures

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[4] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 122, pp. 1–35, 2019.

[5] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.

[6] G. Abdelmoumin, D. B. Rawat, and A. Rahman, "On the performance of machine learning models for anomaly-based intelligent intrusion detection systems for the Internet of Things," *IEEE Internet Things J.*, early access, Aug. 10, 2021, doi: 10.1109/JIOT.2021.3103829.

[7] H. Wu, H. Han, X. Wang, and S. Sun, "Research on artificial intelligence enhancing Internet of Things security: A survey," *IEEE Access*, vol. 8, pp. 153826–153848, 2020.

[8] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2022–2030.

[9] M. Garchery and M. Granitzer, "Identifying and clustering users for unsupervised intrusion detection in corporate audit sessions," in *Proc. IEEE Int. Conf. Cognit. Comput. (ICCC)*, Jul. 2019, pp. 19–27.

[10] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu, and Y. Liu, "ZeroWall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2479–2488.

[11] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.

[12] A. Mezina, R. Burget, and C. M. Travieso-González, "Network anomaly detection with temporal convolutional network and U-Net model," *IEEE Access*, vol. 9, pp. 143608–143622, 2021.

[13] Z. Shi, J. Li, C. Wu, and J. Li, "DeepWindow: An efficient method for online network traffic anomaly detection," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 2403–2408.

[14] L. Zhang, Z. Lv, X. Zhang, C. Chen, N. Li, Y. Li, and W. Wang, "A novel approach for traffic anomaly detection in power distributed control system and substation system," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2019, pp. 408–417.

[15] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Variational data generative model for intrusion detection," *Knowl. Inf. Syst.*, vol. 60, no. 1, pp. 569–590, Jul. 2019.

[16] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42169–42184, 2020.

[17] J. Xing and C. Wu, "Detecting anomalies in encrypted traffic via deep dictionary learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 734–739.

[18] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1646–1685, 3rd Quart., 2020.

[19] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 3rd Quart., 2020.

[20] B. Xia, J. Yin, J. Xu, and Y. Li, "LogGAN: A sequence-based generative adversarial network for anomaly detection based on system logs," in *Proc. Int. Conf. Sci. Cyber Secur.* Cham, Switzerland: Springer, 2019, pp. 61–76.

[21] A. Farzad and T. A. Gulliver, "Oversampling log messages using a sequence generative adversarial network for anomaly detection and classification," 2019, *arXiv:1912.04747*.

[22] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," pp. 1–7, Feb. 2018, *arXiv:1802.10135v1*.

[23] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[24] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "Latent variable based anomaly detection in network system logs," *IEICE Trans. Inf. Syst.*, vol. E102.D, no. 9, pp. 1644–1652, 2019.

[25] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1285–1298.

[26] M. Du, Z. Chen, C. Liu, R. Oak, and D. Song, "Lifelong anomaly detection through unlearning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1283–1297.

[27] R. Yang, D. Qu, Y. Gao, Y. Qian, and Y. Tang, "NLSALog: An anomaly detection framework for log sequence in security management," *IEEE Access*, vol. 7, pp. 181152–181164, 2019.

[28] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.

[29] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Comput. Electr. Eng.*, vol. 50, pp. 91–101, Feb. 2016.

[30] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Lett.*, vol. 3, no. 1, pp. 1–4, Jan. 2019.

[31] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Proc. Int. Conf. Cloud Comput.* Cham, Switzerland: Springer, 2019, pp. 161–176.

[32] L. Sun, Y. Zhou, Y. Wang, C. Zhu, and W. Zhang, "The effective methods for intrusion detection with limited network attack data: Multi-task learning and oversampling," *IEEE Access*, vol. 8, pp. 185384–185398, 2020.

[33] K. R. M. Fernando and C. P. Tsokos, "Dynamically weighted balanced loss: Class imbalanced learning and confidence calibration of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 14, 2021, doi: 10.1109/TNNLS.2020.3047335.

[34] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

[35] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset," *IEEE Access*, vol. 9, pp. 140136–140146, 2021.

[36] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[37] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 859–864.

[38] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM SIGOPS 22nd Symp. Oper. Syst. Princ. (SOSP)*, 2009, pp. 117–132.

[39] H.-K. Bui, Y.-D. Lin, R.-H. Hwang, P.-C. Lin, V.-L. Nguyen, and Y.-C. Lai, "CREME: A toolchain of automatic dataset collection for machine learning in intrusion detection," *J. Netw. Comput. Appl.*, vol. 193, Nov. 2021, Art. no. 103212.

[40] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," 2020, *arXiv:2008.06448*.

[41] M. Kim, Y. J. Won, H. Lee, J. Hong, and R. Boutaba, "Flow-based characteristic analysis of internet application traffic," in *Proc. IEEE Int. Workshop End-End Monit. Techn. Services*, San Diego, CA, USA, Oct. 2004, pp. 62–67.

[42] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

YING-DAR LIN (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), in 1993. He was the Founder and the Director of the Network Benchmarking Laboratory (NBL), during 2002–2018, which reviewed network products with real traffic and automated tools, and has been an approved test lab of the Open Networking Foundation (ONF). He was a Visiting Scholar at Cisco Systems in San Jose during 2007–2008, the CEO at Telecom Technology Center, Taiwan, during 2010–2011, and the Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He also cofounded L7 Networks Inc., in 2002, later acquired by D-Link Corporation, and O'Prueba Inc., a spin-off from NBL, in 2018. He is currently a Chair Professor of computer science at the National Yang Ming Chiao Tung University, Taiwan. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He published a textbook, *Computer Networks: An Open Source Approach*, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011). His research interests include network security, wireless communications, network softwarization, and machine learning for communications. He received the 2017 Research Excellence Award and the K. T. Li Breakthrough Award. He has served or is serving on the editorial boards of several IEEE journals and magazines, including the Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (COMST) from January 2017 to December 2020. He was an IEEE Distinguished Lecturer (2014–2017) and an ONF Research Associate (2014–2018).

ZI-QIANG LIU received the master's degree in computer science in network engineering from the National Yang Ming Chiao Tung University, Taiwan, in 2021. His research interests include machine learning, network security, and intrusion detection.

REN-HUNG HWANG (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Massachusetts, Amherst. He joined the Department of Computer Science and Information Engineering, National Chung Cheng University, in 1993, where he is currently a Distinguished Professor and the Chief Information Technology Officer. He worked as the Dean of the College of Engineering, during 2014–2017. He has published more than 250 international journals and conference papers. His current research interests include the Internet of Things, network security, cloud/edge/fog computing, and 5G V2X networks. He received the IEEE Best Paper Award from IoV 2019, IEEE Ubi-Media 2018, IEEE SC2 2017, and IEEE IUCC 2014.

VAN-LINH NGUYEN (Member, IEEE) received the Ph.D. degree in computer science and information engineering from the National Chung Cheng University (CCU), Taiwan, in 2019. He is currently an Assistant Professor with the Department of Information Technology, Thai Nguyen University of Information and Communication Technology (ICTU), Vietnam. His current research interests include cybersecurity, network/edge intelligence, autonomous driving, and vehicular networks.

PO-CHING LIN (Member, IEEE) received the Ph.D. degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2008. He joined the Faculty of the Department of Computer Science and Information Engineering, National Chung Cheng University, in August 2009. He is currently a Professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.

YUAN-CHENG LAI received the Ph.D. degree from the Department of Computer and Information Science, National Chiao Tung University, in 1997. He joined the Faculty of the Department of Information Management, National Taiwan University of Science and Technology, in August 2001, where he has been a Distinguished Professor, since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and the IoT security.