



Article

A Novel Deep Learning Framework for Intrusion Detection Systems in Wireless Network

Khoa Dinh Nguyen Dang ^{1,†} , Peppino Fazio ^{1,2,†} and Miroslav Voznak ^{1,*,†}

¹ Department of Telecommunications, VSB—Technical University of Ostrava, 708 00 Ostrava, Czech Republic; khoa.dinh.nguyen.dang.st@vsb.cz (K.D.N.D.); peppino.fazio@unive.it (P.F.)

² Department of Molecular Sciences and Nanosystems, Ca' Foscari University of Venice, 30123 Venezia, Italy

* Correspondence: voznak@ieee.org

† These authors contributed equally to this work.

Abstract: In modern network security setups, Intrusion Detection Systems (IDS) are crucial elements that play a key role in protecting against unauthorized access, malicious actions, and policy breaches. Despite significant progress in IDS technology, two of the most major obstacles remain: how to avoid false alarms due to imbalanced data and accurately forecast the precise type of attacks before they even happen to minimize the damage caused. To deal with two problems in the most optimized way possible, we propose a two-task regression and classification strategy called Hybrid Regression–Classification (HRC), a deep learning-based strategy for developing an intrusion detection system (IDS) that can minimize the false alarm rate and detect and predict potential cyber-attacks before they occur to help the current wireless network in dealing with the attacks more efficiently and precisely. The experimental results show that our HRC strategy accurately predicts the incoming behavior of the IP data traffic in two different datasets. This can help the IDS to detect potential attacks sooner with high accuracy so that they can have enough reaction time to deal with the attack. Furthermore, our proposed strategy can also deal with imbalanced data. Even when the imbalance is large between categories. This will help significantly reduce the false alarm rate of IDS in practice. These strengths combined will benefit the IDS by making it more active in defense and help deal with the intrusion detection problem more effectively.

Keywords: intrusion detection systems; wireless network; deep learning



Citation: Nguyen Dang, K.D.; Fazio, P.; Voznak, M. A Novel Deep Learning Framework for Intrusion Detection Systems in Wireless Network. *Future Internet* **2024**, *16*, 264. <https://doi.org/10.3390/fi16080264>

Academic Editor: Paolo Bellavista

Received: 21 June 2024

Revised: 13 July 2024

Accepted: 23 July 2024

Published: 25 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Literature Review and Build-Up Ideas

Wireless networks have become the backbone of modern communication, enabling connectivity across a wide range of devices and applications. However, this ubiquity has also made them a prime target for malicious actors, who constantly seek to exploit vulnerabilities for data theft, disruption, or espionage [1,2]. Therefore, Intrusion Detection Systems (IDS) are essential tools for protecting wireless networks. Developing IDS systems has attracted the attention of researchers for many decades since the early age of wireless networks to observe and secure the accessibility to networks. Until now, the design of IDS can be categorized according to the detection technique it employs. There are two main types and one hybrid type [3,4]:

- Signature-based IDS (or knowledge-based detection)—A signature-based IDS solution typically monitors inbound network traffic to find sequences and patterns that match a particular attack signature. Its strength is its low false alarm rates compared to anomaly-based IDS. However, a major limitation of signature-based IDS solutions is their inability to detect unknown attacks. Malicious actors can simply modify their attack sequences in malware or other types of attacks to avoid detection. Some research in this type of IDS is [5–7];

- Anomaly-based IDS (or behavior-based detection, statistical-based detection)—A behavior or anomaly-based IDS solution acts beyond identifying particular attack signatures to detect and analyze malicious or unusual patterns of behavior. This type of system applies Artificial Intelligence (AI) and ML to analyze large quantities of data and network traffic to pinpoint the anomalies. Despite having higher false alarm rates than knowledge-based IDS, anomaly-based IDS can adapt to new, unique, or original attacks, and it is less dependent on identifying specific operating system vulnerabilities. Some contributions in this field are [8–10];
- Hybrid IDS—A combination of the types mentioned above. This type of system can effectively pinpoint the observed attack types and learn the pattern of traffic to track new attack types. It is one of the best solutions and has received the most focus from recent researchers but comes with the cost of high hardware resource consumption and complicated implementation depending on the components that create the system. The contributions in this field are [11–14]

The rise of ML created the opportunity to achieve more excellent IDS. Some frontiers that set the cornerstone for improving the IDS include Chih-Fong Tsai and Yu-Feng (2009) in [15], which investigates the challenges and opportunities of applying machine learning (ML) techniques for network intrusion detection in real-world settings. Or Halqual in [16], who introduced a multi-grade intrusion detection model based on data mining technology. The authors aim to address the shortcomings of traditional IDS, such as high false alarm rates and limited detection capabilities. Numerous more IDS designs based on Linear regression, Support Vector Machine (SVM), Naive Bayes mode, Tree-based family model and clustering models are mentioned in contributions [17–21] and surveys [22–24]. However, despite extensive research and promising results in controlled environments, the adoption of these proposed systems in operational settings remains limited. These contributions with ML and traditional methods often struggle to keep pace with the evolving threat landscape, where the amount of data features required to distinguish between anomaly and normal traffic behavior increases drastically as cyber-attacks become more and more delicate.

The advent of DL, a subset of ML, has opened up new avenues for intrusion detection. DL algorithms, inspired by the structure and function of the human brain, excel at automatically learning intricate patterns and representations from vast amounts of data. This ability to discern subtle anomalies and correlations within network traffic makes DL a promising tool for identifying malicious activity that might elude traditional IDS methods. Some early frontiers in this area are Ghanem and his partners, who performed the research [25], which proposes a novel approach using an enhanced Bat algorithm to train a multilayer perceptron for intrusion detection, highlighting the potential of nature-inspired optimization. The emergence of Graph Neural Networks (GNNs) for IDS has shown promising results due to their ability to model complex network relationships. The authors of [26,27] provide comprehensive surveys on GNNs in IDS, highlighting their adaptability to evolving network structures, although challenges with computational cost and interpretability remain. DL architectures like Convolutional Neural Networks CNNs and Generative Adversarial Networks (GANs) have also been explored. Al-Milli et al., in [28], demonstrated the feasibility of using CNNs and GANs for intrusion detection, but generalization and adversarial robustness remain concerns. Mohammadpour et al., in [29], surveyed CNN-based IDS, emphasizing their capability of automatic feature extraction while noting the need for careful hyperparameter tuning.

In recent years, more advanced DL structures and hybrid models combining different DL architectures have also been investigated. ElSayed et al. [30] proposed a CNN-based model with regularization for SDNs, while Gautam et al. [31] introduced a hybrid Recurrent Neural Network (RNN) with feature optimization. Both show promise but require further validation and generalization. The use of Long Short-Term Memory Networks (LSTM), one of the variants of RNN for IDS, has been explored in various studies. Contributions such as [32,33] were focused on investigating the LSTM-based IDS for host-based and network-based intrusion detection, respectively. These studies demonstrated the effectiveness of

LSTMs in capturing temporal dependencies in network traffic, but the need for large labeled datasets remains a challenge. Further advancements include Bidirectional LSTM (BiLSTM) and hybrid CNN-LSTM models. Chen et al. [34] and Imrana et al. [35] utilized BiLSTMs for intrusion detection, showcasing their ability to capture bidirectional temporal relationships. However, the computational cost associated with training deep BiLSTM models remains a concern. And the most advanced approach is the hybrid between CNN and RNN (LSTM or GRU), which demonstrates their efficacy in capturing temporal dependencies in network traffic in both time and space aspects, as shown in the contributions [36–38]. They offer vastly improved performance and introduce computational complexities but require further research to exploit their full potential in developing IDS.

Almost all of the mentioned contributions and studies have focused on classification approaches. That means the systems only detect and classify the attacks at the moment they occur and, thus, leave the system passive in observation and protection. As a wise idiom said: “An ounce of prevention is worth a pound of cure”. No matter how fast an IDS solves the problem, there is a possibility that the attacks will occur successfully and will more or less damage the wireless system. This is true especially for some dangerous attacks, such as DDoS, because they quickly flood the system with bots and barely give enough time for the system to recognize the attack and decide on a solution to fix the problem. Furthermore, according to their contributions, the IDS can never achieve 100% correctness in prediction, which means there will always be a possibility of wrong prediction. When this happens, the lack of response time can leave the system vulnerable. Therefore, to maximize the protection, it is better to develop a system that can estimate and predict the network traffic status in a short amount of time based on the recent traffic status history. This will allow the IDS to estimate how the traffic will behave in the near future and detect any potential threats that can occur based on past data and the current traffic flow status. This will contribute to making the IDS more active in their observation duty and give them more time to deal with the attacks, thus increasing the efficiency of the protection. One more advantage is that, when actively estimating the traffic for a certain duration, the IDS can be based on not only the current status of the traffic but also the predicted status to decide if there is a potential attack about to occur or not. Hence, the false alarm rate will be reduced in the future.

Regarding this approach, there have been very few frontier studies that have tried to develop the IDS this way. The latest and closest research to this approach is [39], where the authors propose a strategy of using a combination of a CNN, LSTM, and attention models to predict the future T packets. The research shows promising results in which their best model obtained an F1 score (the F1 score is a metric used to evaluate the performance of classification tasks [40]) of 83% for the $T = 1$ packet scenario and reached a 91% F1 score for forecasting an attack in the subsequent $T = 20$ packets. However, there is no imbalanced data consideration in their contribution, which leads to a reduction in the accuracy of the strategy they used, and the combination of three separate models can consume the processing time and resources of the IDS.

Another common limitation of these former contributions that affected the accuracy and precision of prediction is the imbalance in the datasets they used. These include AWID [41,42], CICIDS2017, CSE-CICIDS2018 [28], LITNET, and KDDcup [43]. Some older datasets are still in use, for example, KDD-1999, DARPA 1999, and KDDCup-99 [44,45], and are applied in anomaly, signature, and hybrid-based IDSs and the family of KDD (Knowledge Discovery and Data mining) datasets [46]. No matter what datasets they use and how good they are, they all have a common flaw, which is the imbalance between categories in the datasets. Typically, this flaw comes from the fact that anomaly traffic data, like attacks, is often a rare event compared to the vast amount of normal traffic in a network. As a result, the algorithm may not have enough examples of attack behavior to learn its distinctive features effectively, making it more prone to misclassification. Suppose the dataset used to train the detection algorithm has a disproportionate amount of normal traffic data compared to attack data. In that case, the algorithm may become biased toward

classifying new instances as normal. This bias can lead it to misclassify actual attacks as normal (false negatives) or flag harmless anomalies as attacks (false positives). This is the cause of high false alarming errors in a majority of current IDS.

According to [7], the factor of imbalanced data is unavoidable due to the nature of the cyber security problem, where the “Normal” behavior outnumbers the attacks considerably. As Wilson claimed in [47], there is almost no ultimate technique or method to completely treat the imbalanced data in wireless networks, and depending on the utilized dataset, the researchers will, based on their experience and knowledge, choose the best approaches to deal with their data. These approaches have been studied and applied in numerous research, such as in [48–50], where the authors applied an ML algorithm to reduce the effect of the imbalanced datasets they used; in the contribution [51], the authors introduce using a semi-supervised learning model in IDS. Despite not explicitly mentioned in the paper, this is a smart way to overcome the imbalanced data since in semi-supervised and unsupervised learning, most of the data are not labeled so that the model will learn the underlying patterns of both normal and abnormal behavior fairly and not develop a bias toward any category; however, training these models may require a larger dataset for them to learn efficiently and thus consume time and computation resources. Another good method for overcoming imbalanced data is mentioned in [52], where the authors used radial basis function neural networks, which can model complex decision boundaries and could potentially help them learn patterns from both majority and minority classes, but at the cost of very high computational time required. Other contributions, such as [53–55], focused on the Federated learning method that helps DL models solve the problem of imbalanced data, but it is not really suitable in large-scale networks due to relying on multiple participants. And lastly, in [56–58], some researchers, including Wilson, agreed that hybrid approaches, such as a hybrid system or combined strategy, will be the most suitable approach to deal with imbalances since it can utilize the advantage of multiple systems or algorithms and create the opportunity to achieve a good system for IDS.

Overall, the best and lowest-cost approach to minimize false alarms is limiting the effect of the unbalanced category on the other categories in the data. This includes collecting more data, which is the best option but is very time-consuming and very difficult since if more data could be collected easily, this problem would cease to exist, discarding the overwhelming categories—mostly the “normal” traffic behavior—which would cause a loss of information and bias in prediction; and finding a way to separate the greater number of categories from the lesser number of categories, thus reducing the effect. The last approach is not easy to achieve but is more time-saving, cost-saving, and information-preserving than the other solutions. This is the solution that will be focused on in this topic and will be integrated into the hybrid system we design to form a strategy, which we call HRC.

1.2. The Methodology's Novelty and Contributions

Technically, there are two contributions from this strategy to the IDS:

The first contribution is that this strategy can be used to predict the future behavior of the wireless network to detect if there is a potential threat about to happen based on observing the IP packages' information. The idea is that the flow of IP packages can reflect the behavior of the networks as it is inherently time-dependent. Each packet in the flow has a timestamp indicating when it was transmitted or received. The order of the packets and the time intervals between them thus can provide the patterns that reflect the behavior of the network over a long period. And these patterns are distinctive in the attacks. Despite how complicated an attack is, it usually leaves a trail when occurring and is presented on the flow IP packages. By carefully exploiting these trails with modern algorithms, such as ML and DL, the system can recognize the signs of an attack before it occurs. Among the algorithms, RNN-LSTM and CNN, as mentioned, are the brightest candidates for extracting and learning the time-space relationship between features in the IP package flow, and thus, they are the most suitable for this strategy. This contribution can help the IDS to be more proactive in detecting the potential threat and having enough time to react to the attack, or

in the case of false detection, due to predicting multiple steps ahead in a short moment in the future, the IDS can have time to reconsider the decision before making a final decision in the upcoming current moment. This idea is considerably new in IDS study, and the closest to it, at this moment, is in the contribution [39].

The second contribution of this strategy is its ability to deal with imbalanced data without discarding any samples or changing the base relationship between categories, such as in [59–61]. The main task of IDS is to recognize the attack occurring on the wireless system; therefore, classifying the traffic behavior is the main task of IDS. However, imbalanced data significantly affect almost all the classification approaches because the class imbalance directly skews the distribution of the target class (or target category) toward the majority class (or majority category). If imbalances are too high, the model becomes overly familiar with the majority class, leading to poor generalization of the minority classes. In the regression task, the impact might be less severe mostly because the regression model focuses more on the distribution of the target class and is not overly biased toward a specific range or set of classes. Therefore, as long as the distribution of the target class remains balanced, the model can still learn the full range of potential output classes and is less likely to be completely biased toward the majority. Therefore, we let the regression part in the HRC strategy handle this instead of the classification part. As mentioned, the primary task of the regression part is to try to predict the behavior of the traffic in the near future, which is specifically “normal” or “attack”, so technically, the regressor also handles classifying these two categories. Regarding the attack, we combine all the attacks into one big category and, thus, raise the number of samples considerably so that it will reduce the imbalanced effect between them and the “normal” category. This idea, plus the fact that regression models are less affected by the imbalanced categorical problem, will help the IDS deal with the imbalanced data more effectively.

Overall, this strategy can provide a new method of handling two major problems in one approach and benefit the IDS by reducing the computational cost, the complicity, and the number of other approaches used in the system to overcome these problems.

With all those reasons and knowledge, in this work, we propose a strategy called HRC for an IDS framework based on DL to improve the ability of IDS that can deal with the imbalanced dataset. Overall, this strategy employs two supervised algorithms: (i) a deep hybrid neural network model using a one-dimensional convolutional layer LSTM (Conv1D-LSTM) to predict traffic behavior according to the traffic pattern; (ii) a one-dimensional convolutional network (CNN1D) to classify the incoming types of attack. Five classes (or categories) of traffic behaviors were chosen from the AWID3 for our research, including Website Spoofing, Evil twin, Botnet, Malware, and Normal.

The paper is structured as follows: Section 1 introduces the topic and reviews the existing related work; Section 2 presents definitions of the problem and the preparation of data; Section 3 describes our proposed HRC strategy; Section 4 shows the experimental results of the individual model used in the strategy; Section 5 evaluates the goodness of the HRC strategy when integrated into an IDS framework; Section 6 concludes the paper with an evaluation and future works.

2. Materials and Methods

2.1. Definition of the Problem

Our proposed strategy is based on IP data packets. Based on the idea of how reference [62] processed the sequence and time-series data, we assume that each data packet X at a specific time t has n features x , indicated as the vector $X^{(t)} = (x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)})$, with the label $Y^{(t)} = (y^{(t)})$. These features describe the IP packet information recorded by the monitor, and the label indicates the packet’s behavior type. The set of data packets containing its characteristics and categories is denoted $F^{(t)} = (X^{(t)}, Y^{(t)}) =$

$\left(\left(x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)} \right), \left(y^{(t)} \right) \right)$. Any packet sequence (or data traffic flow) F_{period} at the time $t + u$ in the future, or $t - u$ in the past, where $u = 0, 1, 2, \dots, U$, may be expressed as:

$$F_{period} = \left(F^{(t \pm u)} \right) = \left(X^{(t \pm u)}, Y^{(t \pm u)} \right) = \left\{ \left[\left(x_1^{(t \pm u)}, x_2^{(t \pm u)}, \dots, x_n^{(t \pm u)} \right), \left(y^{(t \pm u)} \right) \right], \dots, \left[\left(x_1^{(t \pm U)}, x_2^{(t \pm U)}, \dots, x_n^{(t \pm U)} \right), \left(y^{(t \pm U)} \right) \right] \right\}. \quad (1)$$

Following Equation (1), we denote B as the number of past packets (or received packets), where $b = B, \dots, 3, 2, 1$. The past packet sequence F_{past} is:

$$F_{past} = \left(F^{(t-b)} \right) = \left(X^{(t-b)}, Y^{(t-b)} \right) = \left(F^{(t-B)}, \dots, F^{(t-3)}, F^{(t-2)}, F^{(t-1)} \right). \quad (2)$$

Similarly, we denote S as the number of observed packets in the future, where $s = 0, 1, 2, \dots, S$. The future packet traffic flow (or the upcoming flow) is the sequence F_{future} and contains the set of future data packets:

$$F_{future} = \left(F^{(t+s)} \right) = \left(X^{(t+s)}, Y^{(t+s)} \right) = \left(F^{(t)}, F^{(t+1)}, F^{(t+2)}, \dots, F^{(t+S)} \right). \quad (3)$$

Our approach applies a hybrid DL model that contains a detection model and a classification model (respectively referred to as detector and classifier) to manage two different tasks; we separate F_{future} into two terms $F_{future_{det}}$ and $F_{future_{clf}}$, which differ only from each other in their labels $Y^{(t+s)}$:

$$F_{future_{det}} = \left(F_{det}^{(t+s)} \right) = \left(X^{(t+s)}, Y_{det}^{(t+s)} \right), \quad (4)$$

$$F_{future_{clf}} = \left(F_{clf}^{(t+s)} \right) = \left(X^{(t+s)}, Y_{clf}^{(t+s)} \right). \quad (5)$$

Given the input of a past IP packet sequence, we can attempt to predict the type of traffic (normal and attacks) of each future data packet through two tasks:

- Task 1: Receive the input packets and predict the behavior of future packets as normal or abnormal:

$$\widehat{F_{future_{det}}} = \arg \max_{F_{future_{det}}} P((F_{past})). \quad (6)$$

- Task 2: Predict the types of attack for any packet detected as abnormal in Task 1:

$$\widehat{Y_{clf}^{(t+s)}} = \arg \max_{Y_{clf}^{(t+s)}} P(\widehat{F_{future_{det}}}). \quad (7)$$

2.2. Data Preprocessing and Dealing with Imbalanced Problem

As we mentioned in the Introduction, we applied the AWID3 dataset to train and evaluate our DL framework. This dataset is an 802.11ac [41] security dataset recorded using the Wireshark version 3.2.7 tool by researchers at the University of the Aegean. The dataset includes 13 types of attacks that commonly occur in wireless networks. To simplify the data pre-processing, we choose to focus on four types of attacks: Spoofing, Evil Twin, Botnet, and Malware. Each data packet contained 254 features.

The first step is pre-processing the dataset; the high number of features can cause a rise in computational cost and increase the risk of overfitting while training on the neural network model. We proceeded to use the extra trees classifier, a decision-tree-based method [63], to select only the most significant and recurrent features in the data. Figure 1

indicates the proportion of the ten most common features in the data. The brief descriptions of each feature are provided in the Table 1.

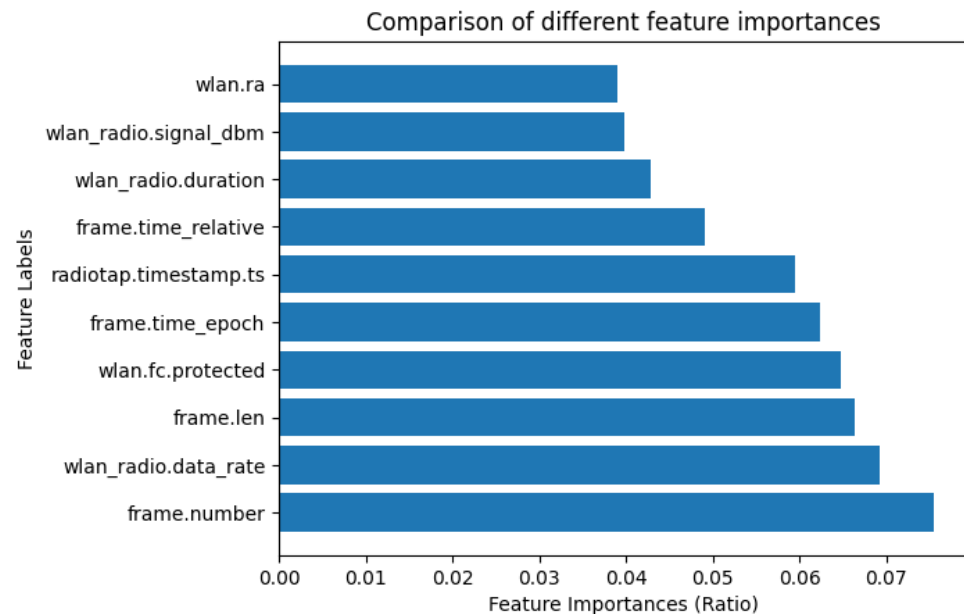


Figure 1. Significant features in the dataset.

Table 1. Descriptions of significant features in the dataset.

Features	Type	Description.
wlan.ra	string	Receiver address
wlan_radio.signal_dbm	numerical	Signal strength (dBm)
wlan_radio.duration	numerical	Duration
frame.time_relative	numerical	Time since reference or first frame
radiotap.timestamp.ts	numerical	Timestamp information
frame.time_epoch	numerical	Epoch time
wlan.fc.protected	numerical	Protected flag
frame.len	numerical	Length of frames
wlan_radio.data_rate	numerical	Data rate
frame.number	numerical	Number of received frames
labels	string	Packet behavior type (categories)

The features contained both numerical and non-numerical value types. We used one-hot encoding to encode the “labels” variable and label encoding to encode the “wlan.ra” variable. The numerical categories also varied along with a range of values, which could have caused a vanishing gradient and led to underfitting in ML and DL. We, therefore, applied a min–max scaler to the numerical features in the range 0–1 to assist in converging the features’ gradients equally. To demonstrate the imbalance problem, the proportions of each data category in our training set are shown in Figure 2.

The chart in Figure 2 indicates that the proportion of normal traffic in the training data overwhelmed the other categories. Botnet and Evil Twin contained the least data samples with only a few thousand instances, which would have led to a heavily imbalanced data problem during training. However, it is noticeable that normal traffic, not anomalous

traffic, is indeed the most frequently encountered type of traffic in practical contexts. This unavoidable problem reduced the precision and recall of the traffic in categories that represented a smaller proportion of the dataset. Previous studies reduced the number of data categories to a maximum of two or three or avoided the inclusion of normal traffic to overcome this problem. In other cases, the entire dataset was kept and a variety of preprocessing techniques such as Under Sampling, Rank Based Support, Oversampling, and Synthetic Minority Oversampling (SMOTE) were applied in [59–61]. Despite the methods applied, the problem could still not be completely eliminated and remained a significant challenge in predicting the categories that have smaller data proportions. A lack of sufficient data is a critical issue that as yet remains unsolved, and currently, the only and most effective solution is perhaps to collect more data.

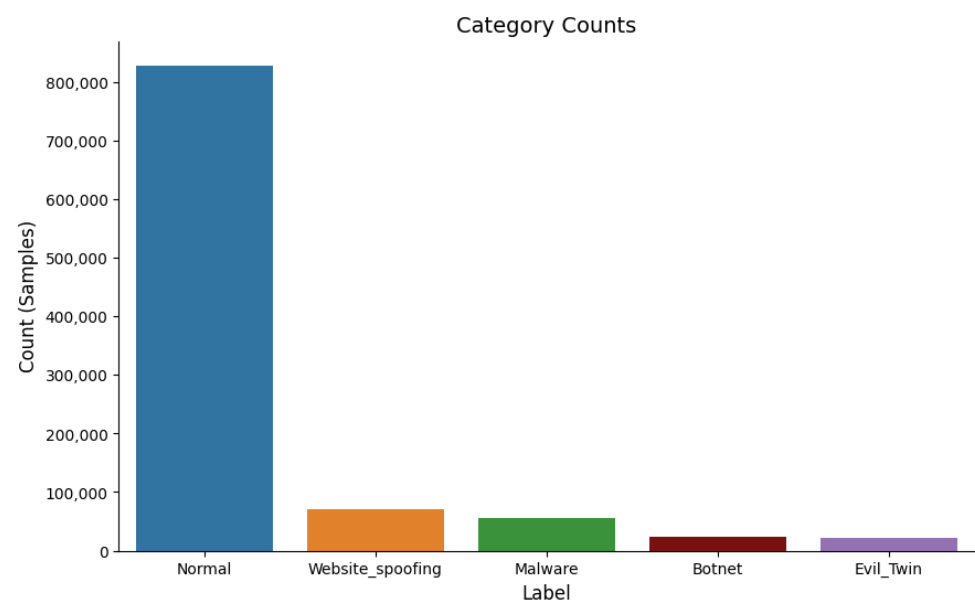


Figure 2. The proportion of training data according to each category.

The imbalanced dataset we used, which was extracted from the AWID3 dataset, contains more than 1,700,000 IP packets and is divided into the training set, validation set, and testing set with the proportions of 60%, 20%, and 20%, respectively. Because there are two models in our proposed strategy, these datasets will be processed in two slightly different ways to serve two purposes:

- For the regression task: 4 types of attacks will be combined into anomaly data and trained together with the normal data.
- For the classification task: we remove the normal data and only keep 4 attack types to train the classifier model.

3. Proposed Strategy

The proposed strategy comprises two primary components: (i) a regression model to detect anomalous flow and predict traffic flow behavior in real-time, and (ii) a classification component that includes a classification model to determine the type of attack that might occur. The functions of these two models are illustrated in the scheme in Figure 3.

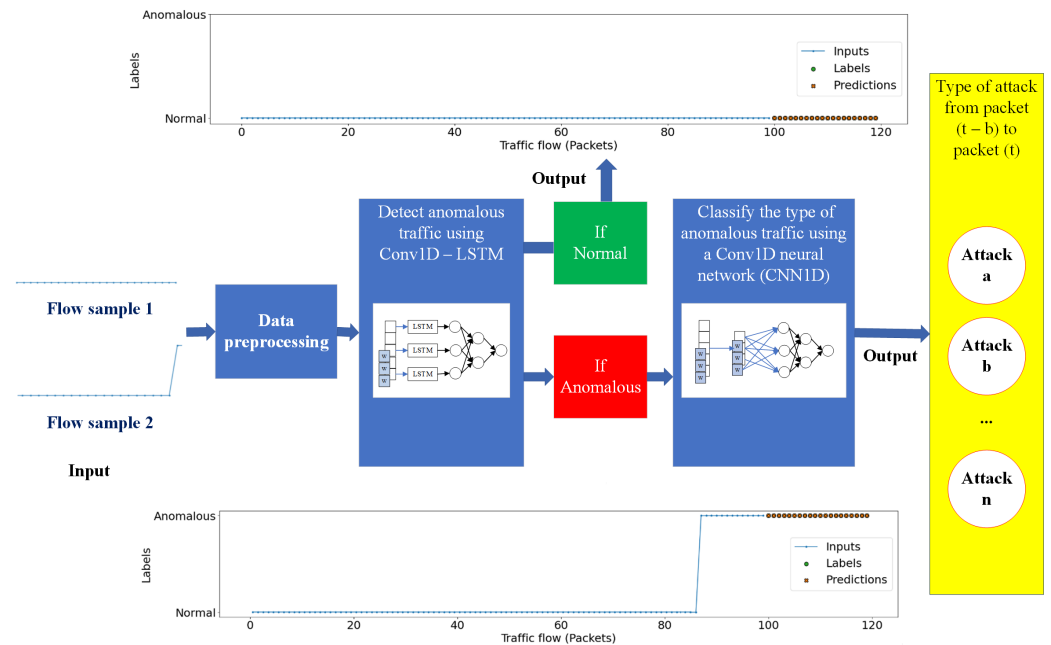


Figure 3. The illustration of our proposed strategy.

The remainder of this section describes the function of each part of the strategy, the models, and the relevant concepts applied to each component.

3.1. The Detector

The detector is the most important component of this strategy. It is used to detect whether the traffic flow input at the current time and at the future time is normal or anomalous. If the traffic is an anomaly, the classification component is triggered and information about the flow is passed to the classification model to identify the attack type and apply cautious policy programming. If traffic flow is normal, these actions will not be triggered.

The model should detect traffic correctly; otherwise, potentially harmful attacks may be missed, or normal traffic may be considered with incorrect caution. It must also predict whether an attack is probable within the next few intervals and not mislead with an alarm for the wrong type of traffic behavior. We used Conv1D-LSTM in this part of the strategy.

Almost every type of neural network uses the gradient descent method to adjust its weights and biases to fit the training problem. This method requires forward and backward propagation. In order to forward propagation, we denote the $F^{(t)}$ as the input sample, which is the sample of IP package features, at time t , of the past $t - b$ samples, and B is just an index for indicating how b changes. We also indicate w as the kernel convolution, which contains the kernel elements $w_{\pm m}$, which are the weights that will be learned during the training process; these kernels have the size of $2m + 1$ elements for $m = 0, 1, 2, \dots$. The $w_{m=0}$ is the kernel's center element, w_{-m} is the kernel's element that is m input samples away from the left side of the center kernel's element, and w_m is the kernel's element that is m input samples away from the right side of the center kernel's element. We also indicate β is the bias; *conv1D* represents the 1D convolution operation with zero-padding, and the dimension of the feature map at the Conv1D layer output is, therefore, equal to the input. Based on (2), if we denote the asterisk $*$ as the convolution operation, the calculation mechanism of the Conv1D layer performed on each input in the sequence F_{past} then can be illustrated in Figure 4:

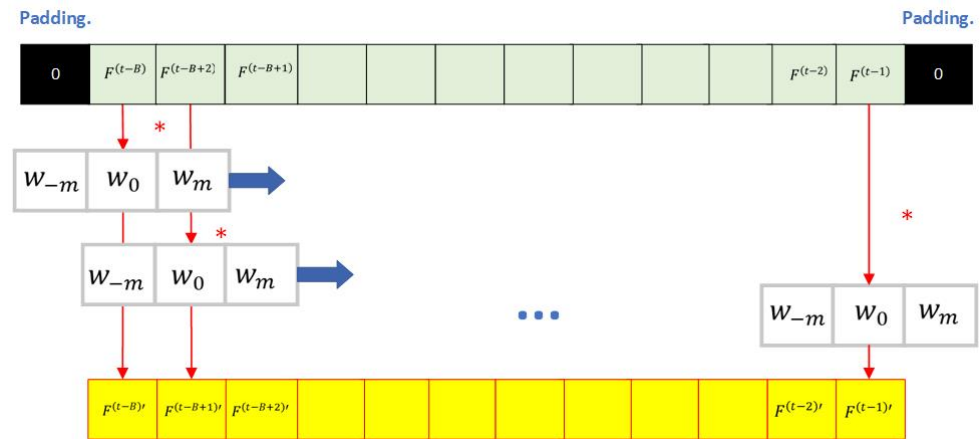


Figure 4. The convolution mechanism.

$$\text{conv1D}\left(F^{(t-b)}, w, \beta\right) = X^{(t-b)} * w + \beta = \sum_{j=-m}^m \left(F^{[(t-b)+j]} \cdot w_j\right) + \beta, \quad b = (0, 1, 2, \dots) \quad (8)$$

In Figure 4, we use $F^{(t-b)'}$ to present the output of the convolutional layers, which is the result of the convolution computation.

The rectified linear activation function (or ReLU function) is used as the activation function at the output of each neuron and is calculated according to the expression:

$$\text{ReLU}(x) = \begin{cases} 1, & x \geq 0. \\ 0, & x < 0. \end{cases} \quad (9)$$

The ReLU activation function serves only to allow the output value at a neuron to progress to the next layer if that value is a positive number and any negative numbers have been discarded. ReLU has an advantage over other activation functions in specifying an open range to the right $[0, +\infty)$ to eliminate the gradient vanishing problem and thus avoid overfitting [64]. Let Z be the output values after applying the ReLU activation function, and the output value at every sample $t - b$ after applying ReLU is calculated as:

$$Z^{(t-b)} = \text{ReLU}\left(\text{conv1D}\left(F^{(t-b)}, w, \beta\right)\right). \quad (10)$$

The final output of the Conv1D layer is a vector Z_b , which contains all neuron outputs:

$$Z_{\text{past}} = \left(Z^{(t-b)}\right) = \left(Z^{(t-B)}, \dots, Z^{(t-3)}, Z^{(t-2)}, Z^{(t-1)}\right), \quad b = B, \dots, 3, 2, 1 \quad (11)$$

Since the IP data packet flow has the characteristics of a data sequence problem, which is recognizable from the manner in which each packet is sent in succession between two nodes throughout the time interval, we can implement a data series algorithm to train the neural network to predict future incoming data from this pattern.

A CNN1D model alone can also represent the features of 1D time-series sequence data by performing 1D convolution operations with multiple filters. However, the ability to extract the feature through the time series of the CNN1D is still limited because the convolution captures neighborhood information limited to the kernel's area and will fail to exploit the time relationship between the long sequence of data from the first packets in the past to the current packets. CNN1D is nonetheless still a feed-forward neural network, which means it has no connection path to allow the association of past information with present information. The most effective method, therefore, is a recurrent neural network. This type of recursive neural network is well-known as a superb tool for solving time series and data sequence problems. To increase memory capability, we can select LSTM or GRU,

two RNN variants that are commonly applied in practical contexts. We used only LSTM since it functions better than GRU with data containing complex features. The structure of an LSTM cell is illustrated in Figure 5.

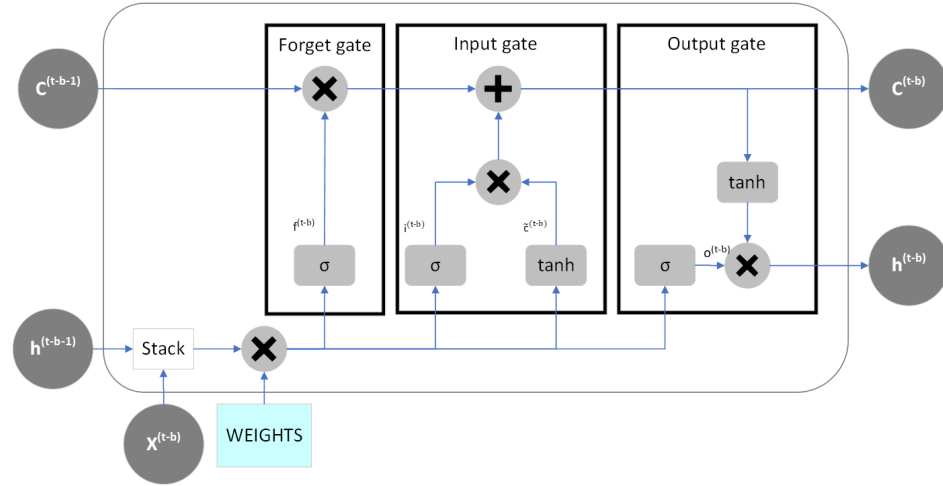


Figure 5. Structure of an LSTM cell.

LSTM cells commonly have four components: a cell state and three logical gates, referred to as the forgotten gate, input gate, and output gate. These three gates control the information flow within the cell by removing or adding information to the cell state. An LSTM cell has separate inputs for input information, the previous cell state and the hidden state, which are the corresponding outputs from the previous cell (except for the first cell state and the first hidden state in the network, which are set to a random value), and two outputs which are the hidden state and cell state of that cell. In Conv1D–LSTM, the output Z_{past} of the Conv1D layer is the input of the LSTM layer, h and c are the hidden and cell states, $(t - 1)$ indicates the earlier time step in the previous cell, t indicates the time step in the current cell, and σ and $\tanh$ represent the sigmoid and the hyperbolic tangent activation functions. We also denote the set of matrices W_f, U_f, β_f , the set of matrices b_f, W_i, U_i, β_i , and the set of matrices W_o, U_o, β_o , which are three sets of weight matrices and bias vectors associated with the input gate $i^{(t)}$, forget gate $f^{(t)}$, and output gate $o^{(t)}$ [65], respectively. The gates are processed according to the following equations:

$$f^{(t-b)} = \sigma(W_f z^{(t-b)} + U_f h^{(t-b-1)} + \beta_f), \quad (12)$$

$$i^{(t-b)} = \sigma(W_i z^{(t-b)} + U_i h^{(t-b-1)} + \beta_i), \quad (13)$$

$$o^{(t-b)} = \sigma(W_o z^{(t-b)} + U_o h^{(t-b-1)} + \beta_o). \quad (14)$$

The cell state assists the model in remembering a very long sequence. Along with the CNN layer, the system benefits by receiving both time and spatial information to form the basis for near-moment prediction. We indicate the internal output as $\tilde{c}^{(t-b)}$, which is the new information admitted to the input gate. The cell outputs, which are $c^{(t-b)}$ and $h^{(t-b)}$, can be calculated according to the element-wise product $\circ$ (or Hadamard product):

$$\tilde{c}^{(t-b)} = \tanh(W_c h^{(t-b-1)} + U_c x^{(t-b)} + b_c), \quad (15)$$

$$c^{(t-b)} = f^{(t-b)} \circ c^{(t-b-1)} + i^{(t-b)} \circ \tilde{c}^{(t-b)}, \quad (16)$$

$$h^{(t-b)} = o^{(t-b)} \circ \tanh(c^{(t-b)}). \quad (17)$$

To update the new weight and bias values, the Conv1D–LSTM performs the second task of the gradient descent algorithm using backward propagation through time in combination with convolutional backward propagation to determine the gradient. This procedure

is repeated until the loss reaches its minimum value. Since the process of backward propagation of Conv1D-LSTM requires a lengthy explanation, we recommend reading [66–68]. The final output of the detector is a set of prediction packets $F_{future_{det}}$ based on (3); F_s is the sequence of output h of the model:

$$F_{future_{det}} = H_{future} = (h^{(t)}, h^{(t+1)}, h^{(t+2)} \dots, h^{(t+s)}). \quad (18)$$

Combining the Conv1D layer and RNN with the LSTM cell layer can significantly boost prediction accuracy. This hybrid model inherits the strengths of two algorithms and allows it to extract, in fine detail, the space–time relationships between data features in a very long traffic sequence. Figure 6 depicts the parameters we selected to build our model.

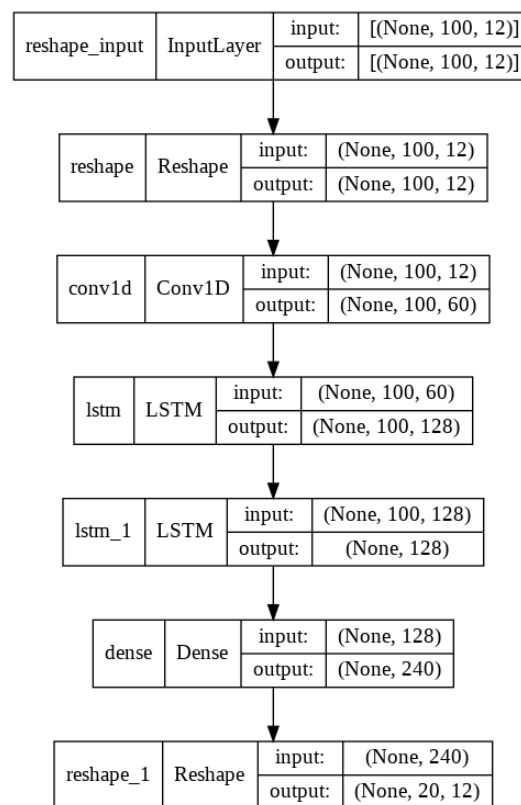


Figure 6. Structure of the Conv1D-LSTM model.

Figure 7 presents the model’s training process in terms of loss and Mean Absolute Error (MAE). We use MAE because IDS datasets, in experiments and in reality, often contain outliers due to abnormal traffic patterns. MAE is less sensitive to outliers and can minimize the overall number of misclassifications, making it a more suitable choice over the other error functions.

Figure 7 shows a sign of slight overfitting, which is probably due to the complex pattern of the IP package traffic in the AWID3 dataset, which makes the model overcomplicate the representation.

We experimented with various look-back values for b and s to determine the most reasonable numbers (which are $b = 100, s = 20$). A total of 100 look-back steps are sufficient for the detector to learn the information necessary to predict future outcomes compared to 80 steps and without consuming much memory compared to 120 steps. For steps-ahead prediction, the range from 20 to 30 steps ahead is the best range of the number of future packets the model can predict for good accuracy, as shown in Figure 8. We selected 20 steps to conserve hardware memory.

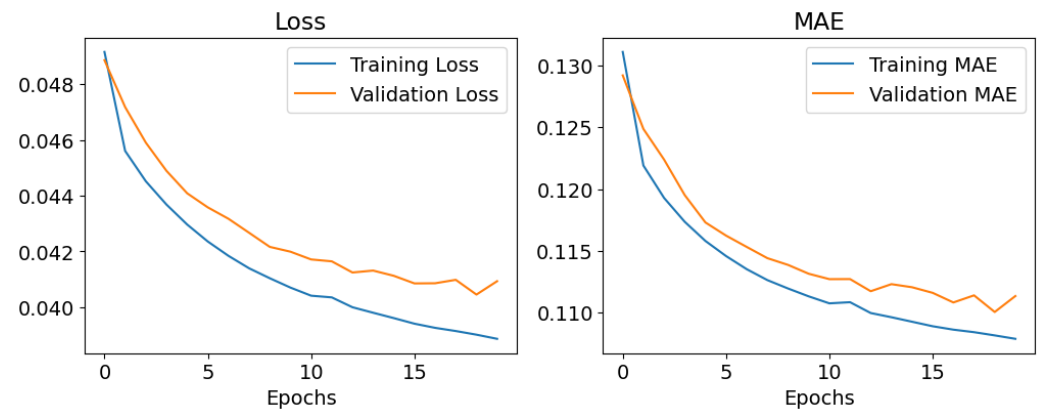


Figure 7. The training result of the regression models.

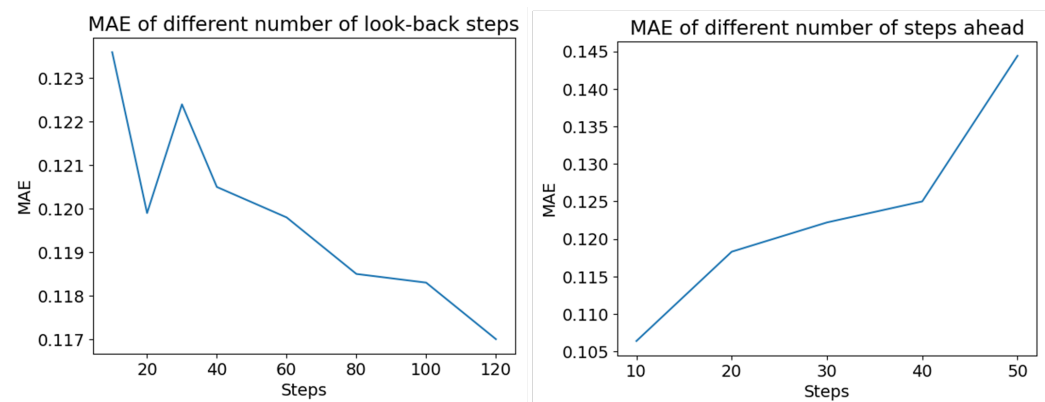


Figure 8. The MAE comparison in the case of different look-back steps (the left graph) and different steps ahead (the right graph).

3.2. The Threshold

We set a threshold on the traffic detection model's output to determine whether the attack is significant according to the proportion of malignant packets in the total number of packets after output. In a real context, because every attack needs a specific duration to successfully harm the system, for example, DDoS, the botnet army needs time to send sufficient requests to bring down the server. This time can be up to 30 s, depending on the robustness of the server's DDoS defenses.

Some minor attacks that last for a fraction of a moment or occur separately are not major threats to the system and can, therefore, be ignored. A greater priority is to focus on major attacks that occur consecutively and last for a very long period, as this is evidence of an incoming attack. We believe that a reasonable threshold to consider the current IP traffic as anomalous is 60%, which means the traffic flow contains less than 60% normal packets and more than 40% harmful packets in total. This threshold assists in decreasing the amount of work in the system since it allows the system to ignore the insignificant anomalous traffic flow and contributes to improving the prediction accuracy. Figure 9 illustrates the way the IDS decides whether the IP packets are an anomaly or normal by using the threshold.

Example:

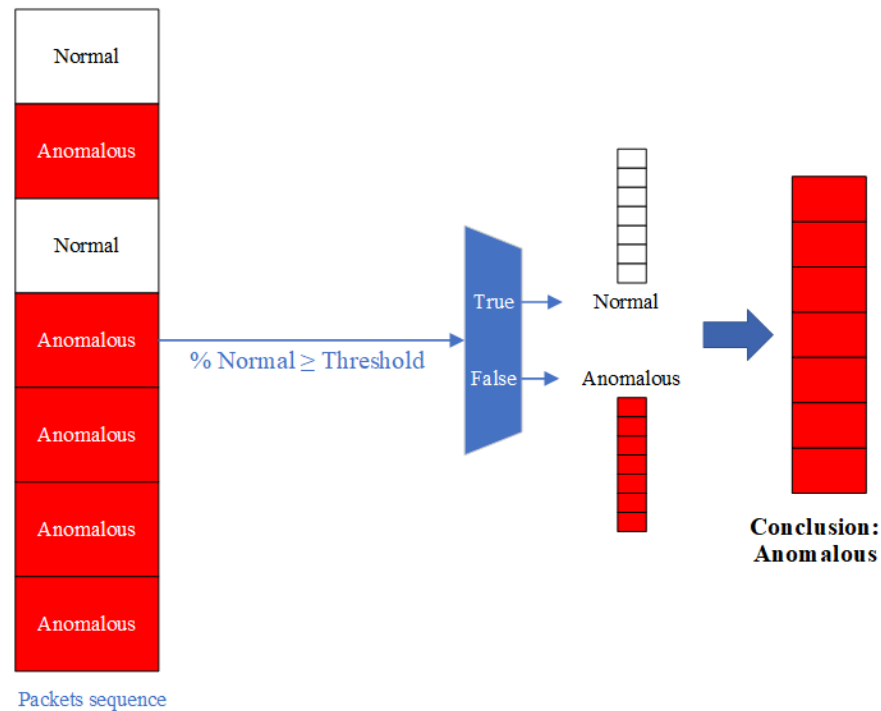


Figure 9. The threshold.

$$Traffic\ behavior = \begin{cases} Normal, & \text{if } \frac{\sum_{T=t-b}^{t-1} X_{Normal}^{(T)}}{100} \geq Threshold \\ Anomaly, & \text{if } \frac{\sum_{T=t-b}^{t-1} X_{Normal}^{(T)}}{100} < Threshold \end{cases} \quad (19)$$

3.3. The Classifier

Anomalous data traffic is directed to the classifier upon detection. The classification model then determines the attack type in each packet and reports the forthcoming attack to operators or the intrusion prevention system to manage the threat. Sufficient time is thus available to prepare the protocols to strike the attack when it occurs and prevent any harm to the system. The classifier must determine the correct type of attack to assist subsequent processes in managing the threat effectively; otherwise, further systems will deploy the wrong protocols and waste time in selecting others. A CNN1D model is one of the most reasonable choices for this task due to its high accuracy and good performance. The classifier is trained to predict only four categories of attack, excluding normal traffic and normal behavior, which is handled by the Conv1D-LSTM model. Since the sequence model is less affected by unbalanced data than the classification model, it is a good method for avoiding the bias in the classifier caused by the overwhelming quantity of normal traffic data. The results of testing and a more specific explanation of this process are given in the next section.

The mechanism of the Conv1D in the CNN1D model is similar to the Conv1D layer in the Conv1D-LSTM in terms of its method of calculating the convolution and the applied activation function. The only difference is not using the padding for the data at each layer's input so the model can down-sample the data quickly to retrieve only the most important features. Since its primary task is classifying the input packet data according to the best matching attack of four attack types, the input will be the features $(x_1^{(t+s)}, x_2^{(t+s)}, x_3^{(t+s)}, \dots, x_n^{(t+s)})$ of each IP packet $X^{(t+s)}$ in the sequence F_{future_det} . Hence,

the output is the category that corresponds to the input feature and is determined by the softmax activation function, given by

$$P^{(t+s)} = P(X^{(t+s)}) = \frac{e^{X_j^{(t+s)}}}{\sum_j^J e^{X_j^{(t+s)}}}, \quad (20)$$

where J is the number of categories (or we can say as classes) that are the attack types.

The predicted type of attack will be the type that has the largest $P^{(t+s)}$:

$$\widehat{Y^{(t+s)}} = \operatorname{argmax}(P_j^{(t+s)}), \quad j = 0, 1, 2, \dots, J. \quad (21)$$

For classification purposes, the categorical loss will be used to measure performance, which is presented by

$$L(\widehat{Y^{(t+s)}}, Y^{(t+s)}) = -\frac{1}{S} \sum_{s=1}^S \sum_{j=0}^J Y_j^{(t+s)} \cdot \log(P_j^{(t+s)}). \quad (22)$$

The model optimizes this loss during the training process until it reaches the minimum value.

After all processes, the output of the classifier, which is the system's output, is a vector of predicted attack types:

$$\widehat{Y_{future}} = (\widehat{Y^{(t+s)}}) = (\widehat{Y^{(t)}}, \widehat{Y^{(t+1)}}, \widehat{Y^{(t+2)}}, \dots, \widehat{Y^{(t+S)}}), \quad s = 0, 1, 2, 3, \dots, S. \quad (23)$$

We selected the following parameters for our CNN1D model, which are demonstrated in Figure 10:

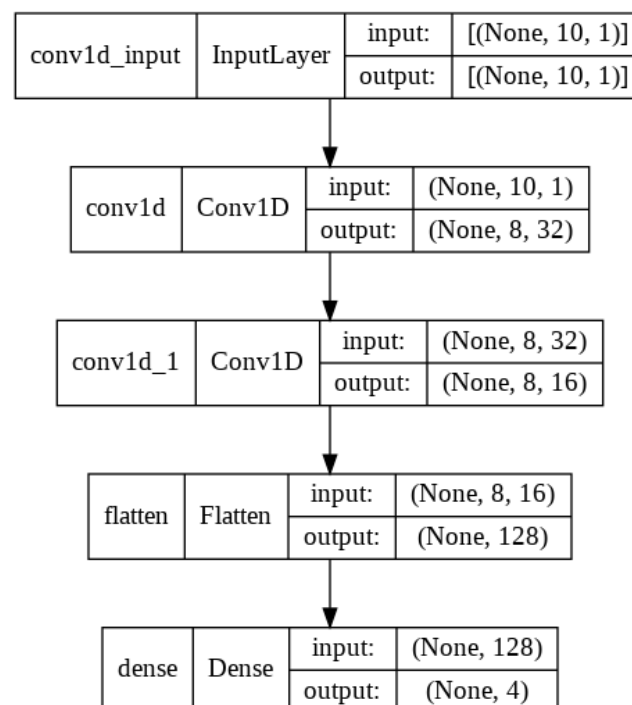


Figure 10. Conv1D structure for the classifier.

Figure 11 shows the model's training results in terms of loss and accuracy. The model reached the highest accuracy score very quickly.

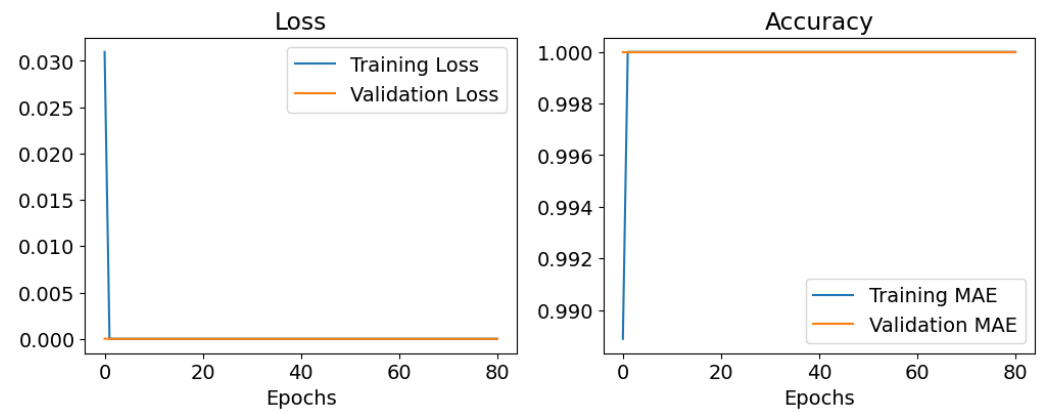


Figure 11. The training results of the classification model.

The change in dimensions of the traffic data through the entire strategy is illustrated in Figure 12.

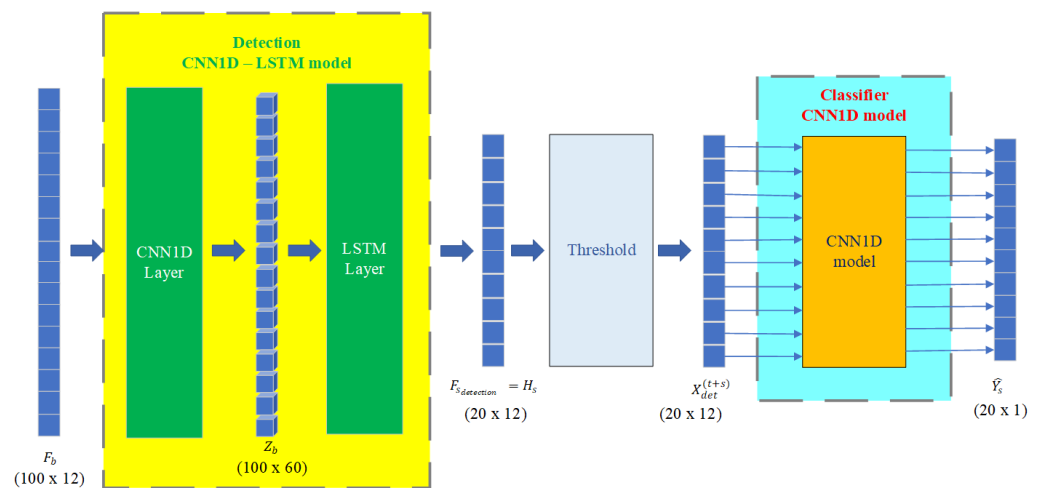


Figure 12. Traffic data dimensions and flow through the architecture.

3.4. Parameters Tuning

Figures 13 and 14 present some parameter tuning made in the strategy's models.

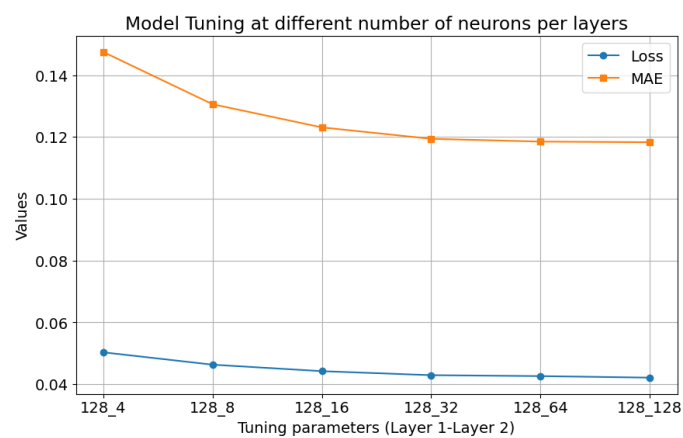


Figure 13. Tuning parameters with two different layers.

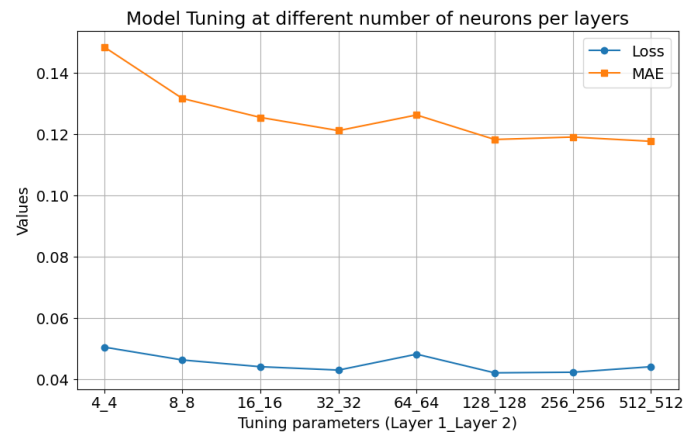


Figure 14. Tuning parameters with two similar layers.

According to Figures 13 and 14, the MAE and loss decrease gradually as the number of layers increases. A higher number of neurons per layer may slightly decrease the error in prediction but will consume more memory and will incur higher computational costs. Therefore, 128 and 128 LSTM layers in the regression models are probably the optimized choices for the HRC strategy to predict and classify traffic behaviors more efficiently without being so complex and consuming so many memory resources.

3.5. The Metrics

We applied common statistical techniques to evaluate the classification problem:

1. True Positive (TP): a packet is classified correctly by the model into its category of behavior. The result is a True Positive.
2. False Positive (FP): a packet belongs to a category of behavior but is classified into the wrong category. This result is a False Positive.
3. True Negative (TN): a packet is classified correctly by the model as not belonging to a category of behavior. This result is a True Negative.
4. False Negative (FN): a packet does not belong in a category of behavior, but the model incorrectly classifies the packet into the category. The result is a False Negative.

From these statistics, we used accuracy, precision (PC), recall (RC), and F1 score (F1) metrics to evaluate the model's efficiency during testing, and also, for observation, we applied a confusion matrix (CF) to illustrate the relationships between each PC and RC of each attack class [69].

4. Experimental Results of Individual Model

4.1. Testing the Detector

The detector, which contains the Conv1D-LSTM model, performed very well in predicting from the testing dataset. Some of the model's prediction results are illustrated in Figure 15. In addition, some poor prediction results are shown in Figure 16.

The results show that the model detected anomalous traffic satisfactorily. In this case, the traffic behavior altered slightly, but the number of incorrect predictions was insignificant and could be ignored as they passed through the threshold block. If traffic behavior changes suddenly, the number of incorrect predictions will rise. It is a natural fact that sudden changes give the Conv1D-LSTM no past information to learn from, and therefore, it cannot adjust immediately to deliver correct predictions until some time later. Even humans predict incorrectly in such circumstances.

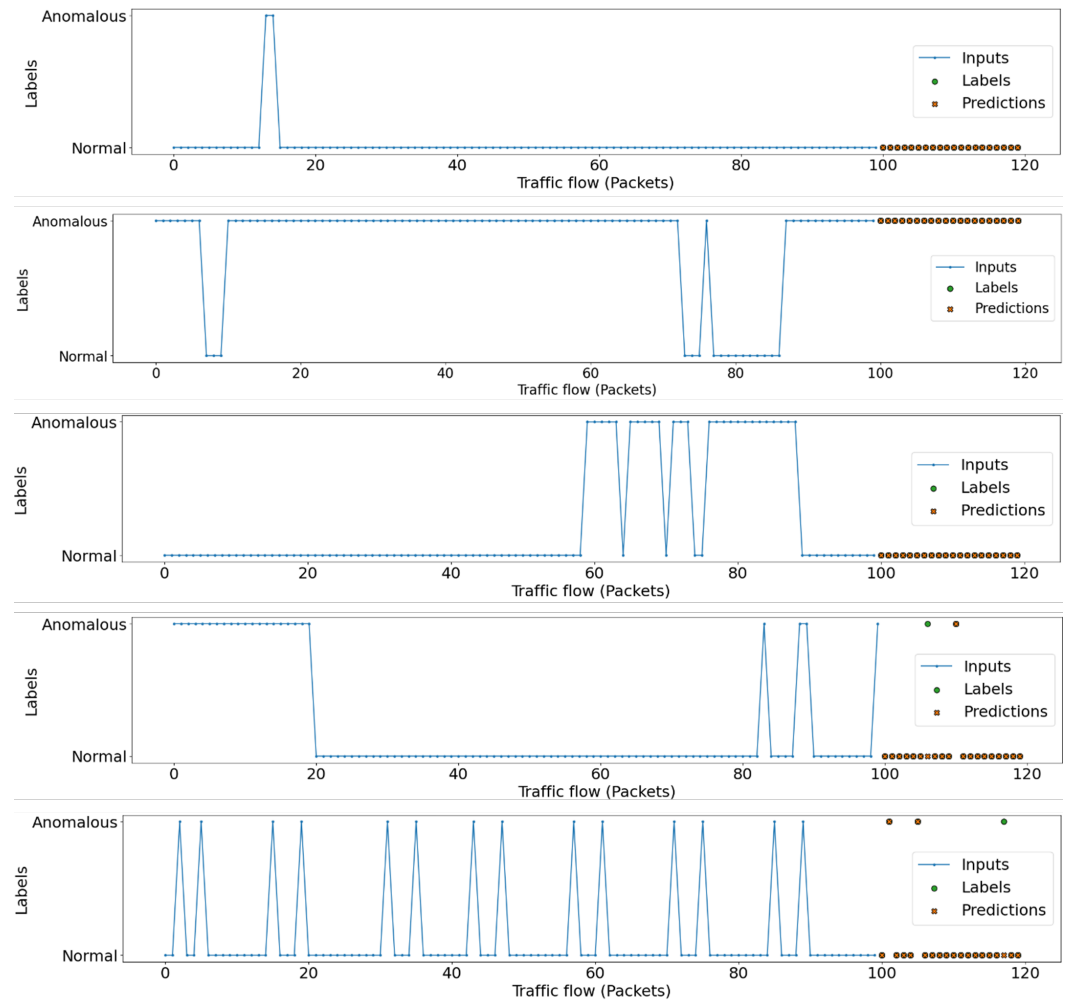


Figure 15. Correct traffic behavior predictions.

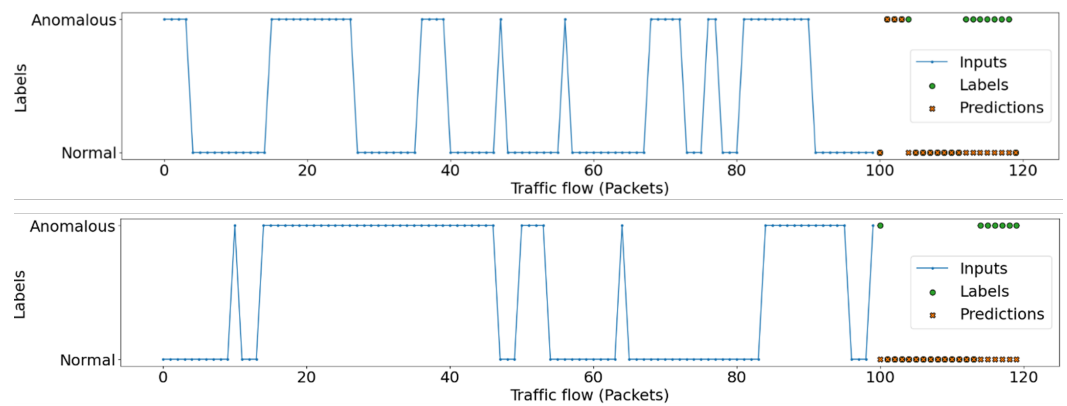


Figure 16. Incorrect traffic behavior predictions.

4.2. Testing the Classifier

As we described earlier, the benefit of the HRC strategy is reducing the effect of the imbalanced dataset on prediction accuracy. Since the regression model already deals with the “Normal” and “Anomaly” data, the classification model only needs to classify the attacks within the “Anomaly” data without concern for the “Normal” data. Therefore, training the classifier only on the “Anomaly” data will significantly reduce the error caused by imbalanced data. This is illustrated in Figure 17.

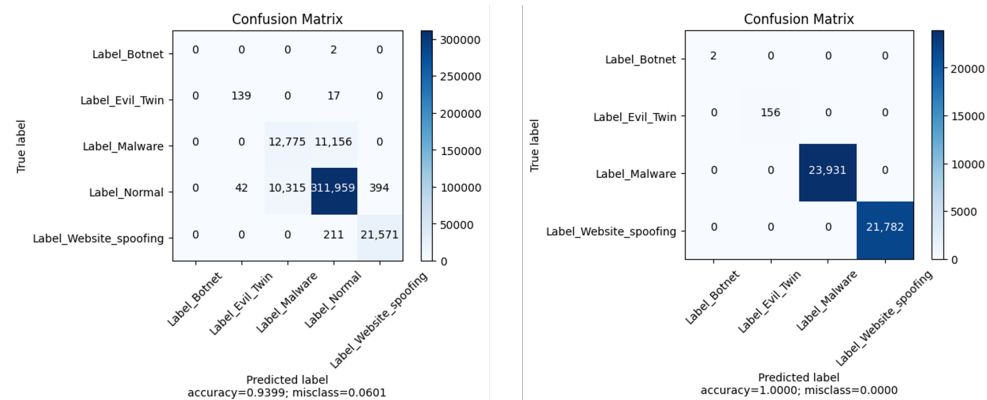


Figure 17. The confusion matrices demonstrate the classifier’s prediction results when including the normal category (left graph) and excluding the normal category (right graph).

Figure 17 shows two confusion matrices that represent the classification results of the same model but were trained on two different cases: with (left) and without (right) the normal IP packages category. The confusion matrix on the left indicates a high proportion of misclassification of the four attack types as a result of imbalanced data caused by the overwhelmingly large number of normal IP package data. In this case, the prediction accuracy is only more than 0.93. The confusion matrix on the right shows the result of training the classifier to predict only the four types of attack (the detector handled normal traffic). The classifier predicted correctly even with the very little Botnet data available. The model, in this case, did not suffer any bias from the normal traffic category, and the biases between the four attack categories were too small to affect the model.

5. Applying the HRC Strategy to an IDS

5.1. Framework Testing

We test the IDS framework that applied our HRC strategy with the testing dataset to evaluate its performance. The results are shown in the confusion matrix in Figure 18 and the classification report in Table 2:

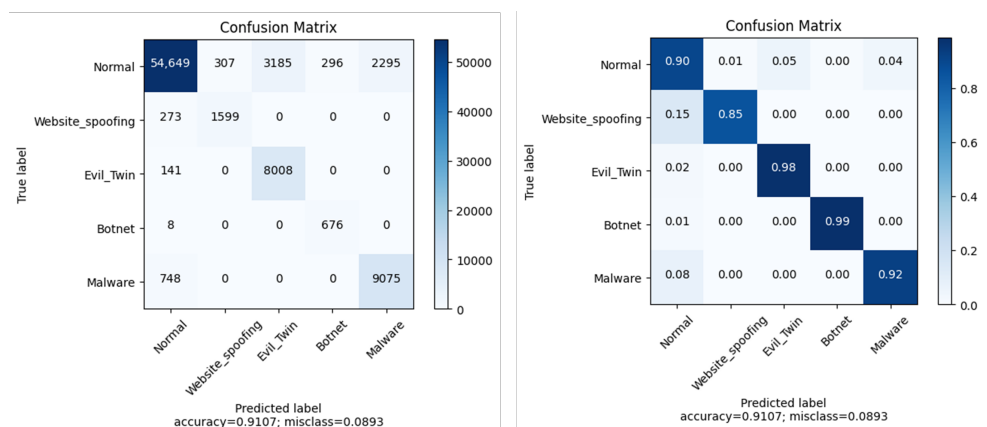


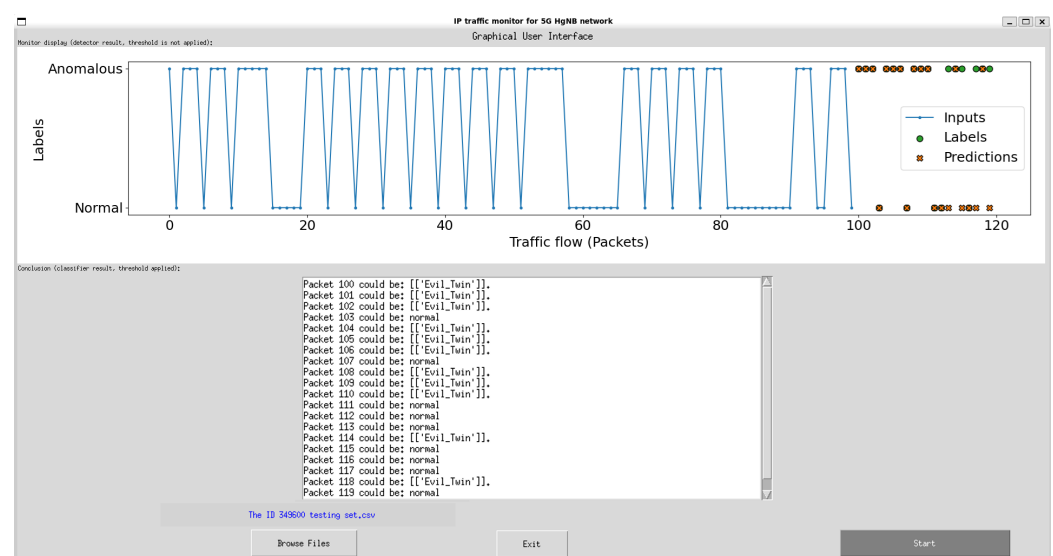
Figure 18. The confusion matrix and normalized confusion matrix of the framework’s predictions from the test dataset.

Table 2. Classification report.

Type of Traffic Flow	Precision	Recall	F1 Score	Support
Normal	0.98	0.90	0.94	60,732
Website Spoofing	0.84	0.85	0.85	1872
Evil Twin	0.72	0.98	0.83	8149
Botnet	0.70	0.99	0.82	684
Malware	0.80	0.92	0.86	9823
Accuracy	–	–	0.91	81,260
Macro average	0.81	0.93	0.86	81,260
Weighted average	0.93	0.91	0.91	81,260

Generally, the proposed IDS framework's strategy shows a promising result of the overall accuracy reaching over 90%. According to the confusion matrices in Figure 18, the prediction accuracy for each category is no less than 85%; thus, the model satisfactorily predicted four types of attack and normal IP packets. Having a detailed look at Table 2, all the categories' recall scores are higher than 85%, showing that the model correctly identifies the right category for each new piece of data. In the aspect of the precision score, Botnet and Evil Twin, respectively, despite showing a low recall of 70% and 72%, have very high recall scores of 0.99 and 0.98, implying that the system effectively identifies these two attacks but may also occasionally misclassify some normal packets with them. Those precisions and recalls resulted in the overall false alarm rate for the IDS using this strategy being relatively low. Both categories have good F1 scores, showing a balance in their ability to identify relevant categories with their accuracy in avoiding misclassification. The weighted average F1 score of 0.92 indicates that the model performs very well overall. This proved the effectiveness of the proposed strategy in reducing the effect of imbalanced data for the IDS frameworks.

We created an interface for convenient visualization of the results, which is shown in Figure 19. The interface receives the input testing files of IP traffic flows and displays the detection results (the detector's output before the threshold is applied) as a graph and the entire framework's final output (threshold applied).

**Figure 19.** Visualization interface of the framework.

5.2. Comparison and Validation

To evaluate the performance of our chosen models in this framework's strategy, we created other hybrid models with combinations of different types of ML and DL algorithms and compared them to the proposed model for accuracy and F1 score in each prediction category and parameter. We used Conv1D-LSTM, regular LSTM and regular CNN1D for our regression part, combining with each of LR, SVC, DNN, LSTM and GRU for the classification part. We combined them and performed the experiment to see which one was the best for our proposed hybrid strategy. Table 3 presents the results of each model's performance.

As shown in Table 3, the proposed hybrid model between Conv1D-LSTM and CNN1D has the highest accuracy in prediction and the best set of F1 scores due to having a recursive model and feed-forward model that can capture both space and time relationships between features in the dataset. Regarding those hybrid models that comprise the CNN1D as their detector, they were less efficient than the hybrid model with the recursive detector because they contained only feed-forward neural networks that can only capture the space relationship between features in the dataset. The hybrid models that have an ordinary LSTM model as their detectors, such as LSTM-DNN or LSTM-SVC, are not as good as the Conv1D-LSTM and CNN1D model due to lacking the strength of exploiting the space relationship between category features in the dataset. However, despite having lower accuracy in BN attack prediction, they still reach approximately 91% and 81% overall accuracy when having fewer parameters compared to our proposed model. This accuracy is not too bad. Therefore, in the case of the systems that focus on memory saving more than accuracy, they can be used instead. This trade-off is mentioned by Igino and colleagues in the paper [70], in that the choice between complex or simple approaches for an IDS is not always about which one is better than the other; it is about finding the right balance between accuracy, efficiency, and scalability for each situation. Therefore, the task will determine if the used approach is costly or not in terms of computational cost and if it is easy or hard to scale according to the growing demands.

Based on the problem in this paper, which is about classifying the attacks with complex patterns and imbalanced datasets, not only at the current time but also in the near moment in the future, the problem can be considered a complex task, and our proposed strategy with 264,144 parameters is able to handle the task well and can be considered as still relatively small and efficient. Additionally, our strategy can deal with other datasets with more attacks and imbalanced categories, which is mentioned in Section 6. This shows its ability to scale with the growing problems at a certain rate before it cannot handle the problem anymore. And when this happens, we just need to change the models within the strategy to be more suitable for the task. Therefore, compared to other mentioned approaches, the proposed HRC strategy with this setting-up model is one of the brightest candidates for efficient computational cost and scalability.

Table 3. Comparison of our proposed model (we marked with an asterisk) and other models.

Hybrid Model		Accuracy	F1 Score					Params
Detector	Classifier		Normal	WSP	ETW	BN	MW	
Conv1D-LSTM	* CNN1D	0.91	0.94	0.85	0.83	0.82	0.86	264,144
	DNN	0.91	0.94	0.84	0.91	0.41	0.86	272,896
	GRU	0.91	0.94	0.84	0.91	0.40	0.86	327,200
	LSTM	0.91	0.94	0.84	0.91	0.40	0.86	344,608
	LR	0.81	0.94	0.88	0.00	0.43	0.62	262,987
	SVC	0.91	0.94	0.83	0.84	0.72	0.86	262,987
	XGBoost	0.72	0.91	0.10	0.27	0.05	0.00	263,011

Table 3. Cont.

Hybrid Model		Accuracy	F1 Score					Params
Detector	Classifier		Normal	WSP	ETW	BN	MW	
LSTM	CNN1D	0.83	0.89	0.82	0.79	0.73	0.82	82,808
	DNN	0.91	0.94	0.84	0.92	0.39	0.87	244,678
	GRU	0.91	0.94	0.84	0.92	0.39	0.87	235,908
	LSTM	0.91	0.94	0.84	0.92	0.39	0.87	316,372
	LR	0.81	0.94	0.89	0.00	0.42	0.60	234,751
	SVC	0.91	0.94	0.84	0.85	0.69	0.87	234,751
	XGBoost	0.73	0.92	0.10	0.29	0.05	0.00	234,775
CNN1D	CNN1D	0.78	0.83	0.78	0.68	0.65	0.72	15,076
	DNN	0.90	0.94	0.88	0.88	0.44	0.77	23,828
	GRU	0.90	0.94	0.88	0.88	0.43	0.77	78,132
	LSTM	0.90	0.94	0.97	0.79	0.67	0.77	95,540
	LR	0.90	0.94	0.90	0.79	0.75	0.77	13,919
	SVC	0.90	0.94	0.87	0.80	0.70	0.77	13,919
	XGBoost	0.78	0.93	0.12	0.26	0.07	0.00	13,943

5.3. Compare with Other Approaches

Two popular approaches used for classifications in IDS recently are Bayesian, which was applied in the contributions [21], and histogram gradient boosting, a type of histogram-based ensemble algorithm introduced in the research of [71]. We performed an experiment on these two approaches and obtained the results shown in Figures 20 and 21 and Tables 4 and 5.

Table 4. Classification report of the Bayesian approach.

Type of Traffic Flow	Precision	Recall	F1 Score	Support
Normal	1.00	0.88	0.93	368,581
Website spoofing	0.00	0.00	0.00	0
Evil Twin	0.00	0.00	0.00	0
Botnet	0.00	0.00	0.00	0
Malware	0.00	0.00	0.00	0
Accuracy	–	–	0.88	368,581
Macro average	0.20	0.18	0.19	368,581
Weighted average	1.00	0.88	0.93	368,581

According to the results in Figure 18 (our proposed strategy) and classification reports in Table 2 (our proposed strategy), the Bayesian model shows the lowest accuracy in prediction, with a score of only about 0.88, compared to the Histogram Gradient Boosting model, which scored nearly 0.99, and our HRC strategy, which scored 0.91.

The histogram gradient boosting model, despite reaching the highest accuracy score, has a very low F1 score in the two categories with the fewest sample numbers, Botnet and Evil Twin, while archiving a perfect F1 score in the “Normal” category. This happens because the effect of heavily imbalanced data on the model makes the model biased

toward the major category, which is very well-known to occur in almost every classification algorithm if no assisting methods are used.

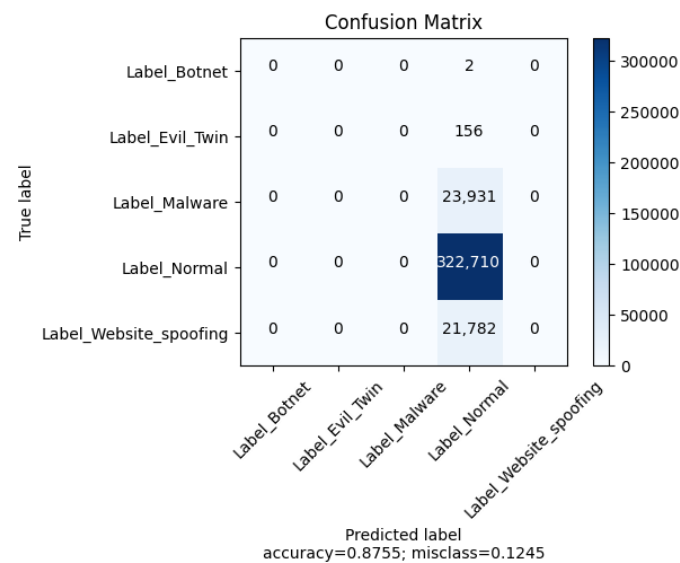


Figure 20. Confusion matrix of the Bayesian approach.

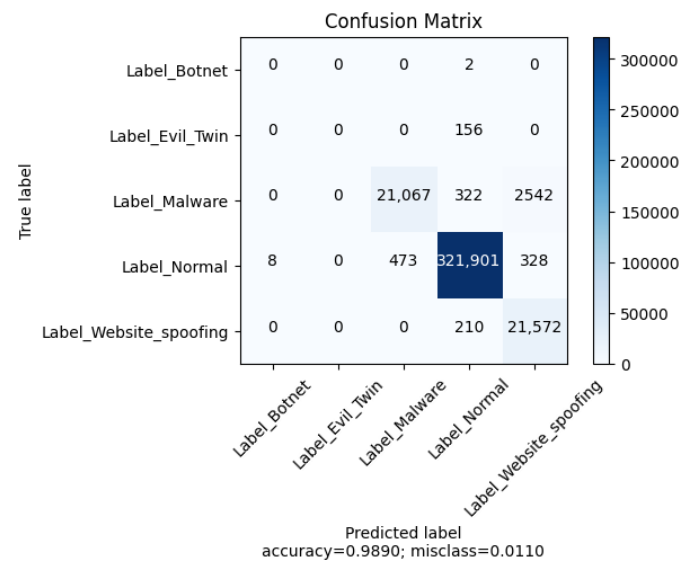


Figure 21. Confusion matrix of the histogram gradient boosting approach.

Our proposed HRC strategy itself can significantly reduce the bias caused by this problem without requiring assisting methods or modifying the data ratio, thus showing an advantage over the other approaches. This is one of the two primary goals of our contribution. The aim is to increase the prediction accuracy among categories in the case of imbalanced data. Therefore, despite the overall accuracy of the prediction not reaching an excellent score, the strategy helps significantly reduce the prediction gap (Precision, Recall, and F1 score) between the minor categories (in our experiment, the Botnet and Evil twins) and the overwhelming major Normal category.

To conclude, by separating the IDS's tasks into prediction and detection and applying the hybrid model, both the detector and the classifier can utilize their true potential in their tasks. Using this strategy, the IDS can predict the traffic flow behavior in the future to prevent incoming threats more effectively. It can also help the IDS to deal with imbalanced data and contributes to reducing the number of false alarms.

Table 5. Classification report of the histogram gradient boosting approach.

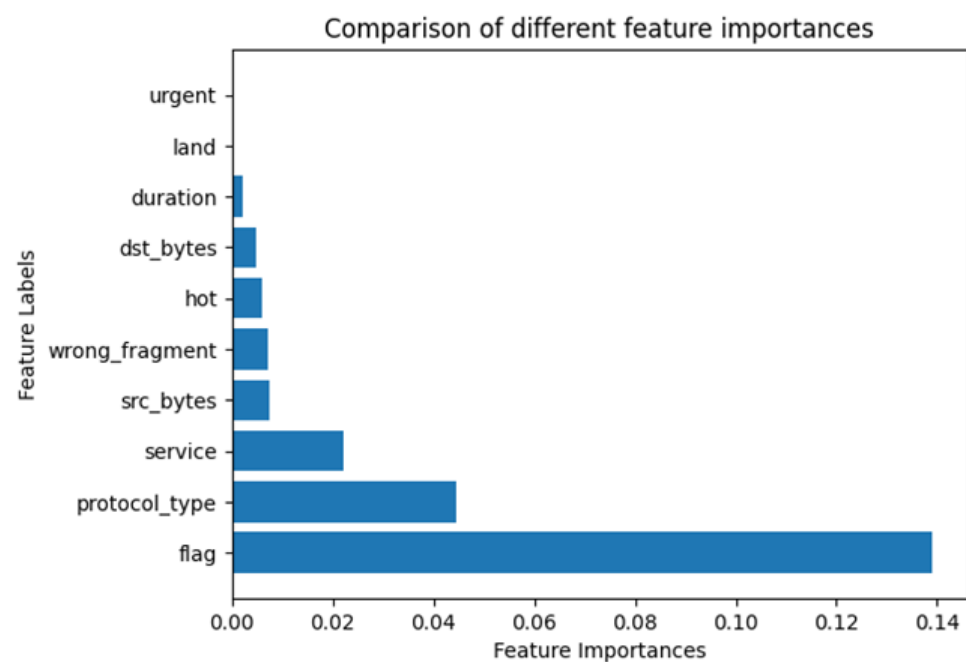
Type of Traffic Flow	Precision	Recall	F1 Score	Support
Normal	1.00	1.00	1.00	322,591
Website spoofing	0.99	0.88	0.93	24,442
Evil Twin	0.00	0.00	0.00	0
Botnet	0.00	0.00	0.00	8
Malware	0.88	0.98	0.93	21,540
Accuracy	–	–	0.99	368,581
Macro average	0.57	0.57	0.57	368,581
Weighted average	0.99	0.99	0.99	368,581

6. Testing the HRC Strategy on Another Dataset

We test our proposed HRC strategy on another dataset to more generally evaluate the applicability of the proposed strategy in another different situation.

The dataset we use here is NSL-KDD, which is the latest version of the KDD cup99 dataset family [46,72]. This dataset contains 22 instances of training intrusion attacks and 41 features, with 21 features related to the connection and 19 features detailing connections within the same host. We proceed to extract the “normal” category alongside seven types of attack, including Neptune, IP sweep, port sweep, smurf, back, and teardrop, to experiment with the proposed HRC strategy.

The process is the same as the process we used for the AWID3 dataset; we chose the best possible features from the data to reduce the computation time and the overfitting effect using the Extra Trees Classifier. Figures 22 and 23 show the data’s most significant features according to the Extra Trees Classifier and the comparison between each category’s number of samples in the training set:

**Figure 22.** Important features according to Extra Trees Classifier.

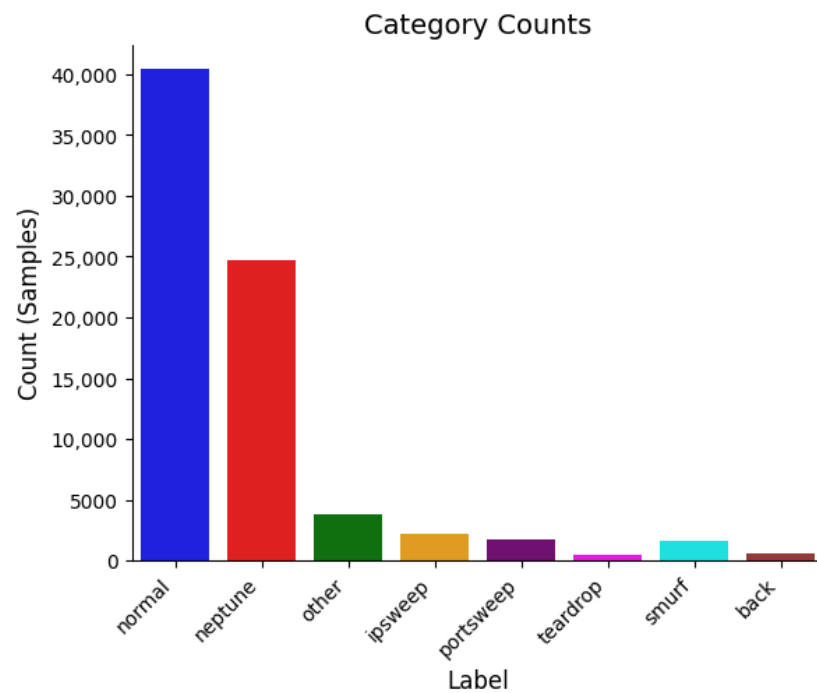


Figure 23. Number of each category's samples in the training set.

The training and validation processes of our Con1D-LSTM and CNN1D models in regression and classification problems are shown in Figures 24 and 25, respectively.

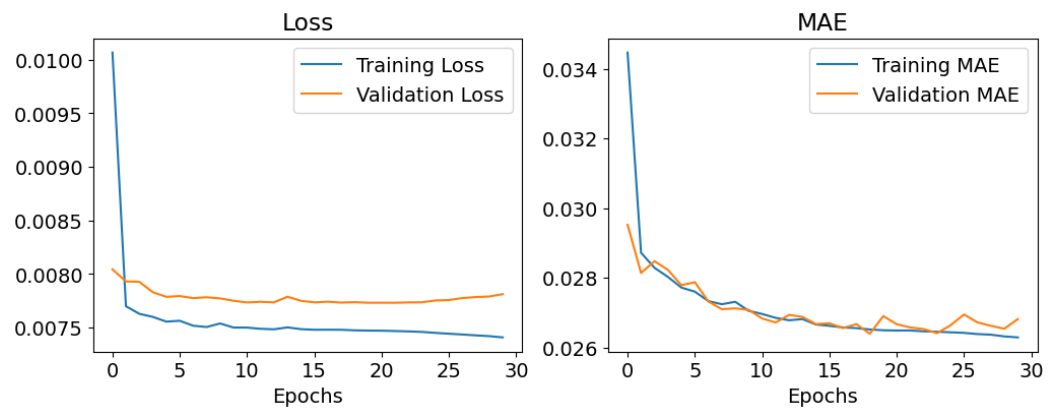


Figure 24. The training results of the regression model on the NSL-KDD training dataset.

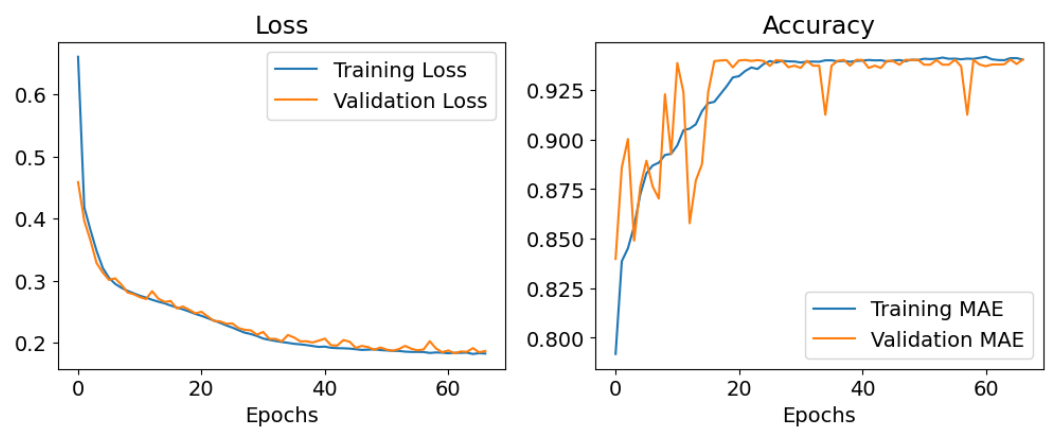


Figure 25. The training results of the classification model on the NSL-KDD training dataset.

We apply the full strategy to the testing data and present the confusion matrix and classification report in Figure 26 and Table 6

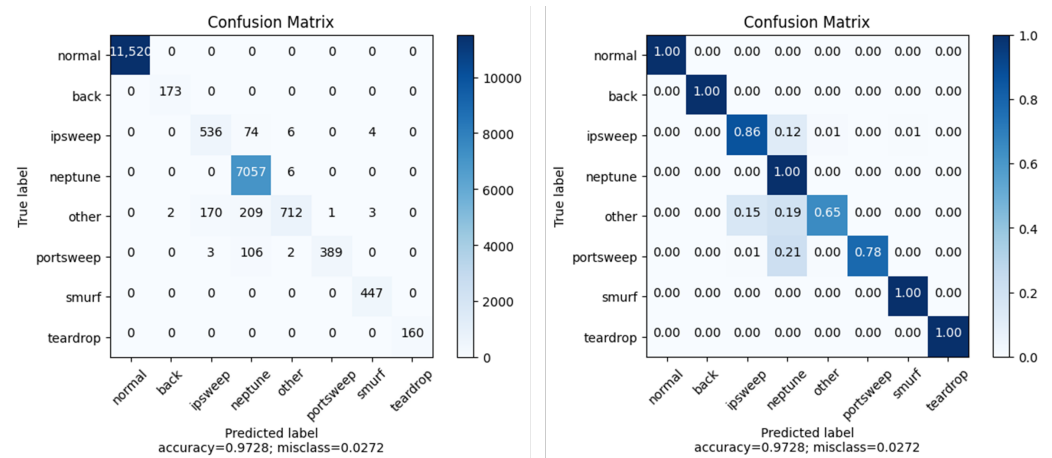


Figure 26. Confusion matrix of the experiment on IDS when applying the proposed strategy.

Table 6. Classification report of the experiment.

Type of Traffic Flow	Precision	Recall	F1 Score	Support
Normal	1.00	1.00	1.00	11,520
Back	0.99	1.00	0.99	173
IP Sweep	0.76	0.86	0.81	620
Neptune	0.95	1.00	0.97	7063
Other	0.98	0.65	0.78	1097
Port sweep	1.00	0.78	0.87	500
Smurf	0.98	1.00	0.99	447
Teardrop	1.00	1.00	1.00	160
Accuracy	–	–	0.97	21,580
Macro average	0.96	0.91	0.93	21,580
Weighted average	0.97	0.97	0.97	21,580

Based on the experimental results, the HRC strategy also achieves good performance in the case of different datasets with more imbalanced categories with an accuracy of approximately 0.97. The two fewest in the number of sample attacks are Back and Teardrop, reaching the high F1 score of about 0.99 and 1.00. The category with the lowest F1 is Port Sweep, with a score of 0.78, while having double the number of data samples compared to the former two. We think the Port Sweep shares some similar features to Neptune or Other attacks that made the models in the strategy have difficulty distinguishing between them during training. This score is not low, relatively, but actually still higher than the Botnet attack in the case of the AWID3 dataset. Overall, it is better than the result of the AWID3 dataset. We personally think that the NSL-KDD dataset has cleaner IP package data and a clearer pattern of the IP traffic flow that helps the models in our proposed strategy capture the information better.

7. Conclusions

In this research, we proposed a new strategy for IDS development called HRC. This strategy comprises two parts: the regression part to predict the near future behavior and

decide if an attack will potentially occur, and the classification part to classify the types of those potential attacks correctly so that the IDS will proactively have a solution to deal with them. We suggested using the Conv1D—LSTM model for our regression part, due to its powerful ability to exploit the time–space relationship among the IP packages in the traffic and predict incoming network behaviors, and using a simple CNN model for the primary task of classifying the attacks due to its light structure and excellent classification abilities.

Typically, most of the former research only focused on developing the classification algorithm to raise the classification results in IDS frameworks at the current timestamp, with little to no consideration for handling imbalanced data. Our strategy is able to deal with imbalanced data without modifying the data and retains the number of samples as well as the ratio between them; in addition, it also helps the IDS accurately predict the incoming behavior of the network in the near future.

In our research, we primarily use the AWID3 dataset, one of the latest and most trustful datasets in the IDS field of research, with an additional NSL-KDD dataset to strengthen the evaluation of the experiments on the AWID3 dataset. The results indicate that the model achieves a high accuracy of 91% overall accuracy in predicting the current and future behavior of IP traffic, specifically 20 IP packages ahead according to our setup. Not only that, but it also achieves a balance in prediction between categories despite the heavy imbalance in sample numbers among categories, particularly F1 scores of 0.83 and 0.82 for the Evil Twins and Bot Net attacks in the AWID3 dataset, with their number of samples only roughly 1/20 compared to the normal category; and 1.00 and 0.99 for Back and Teardrop attacks in the NSL-KDD dataset despite their number of samples being barely 1/50 compared to their normal category. Based on these results, this approach can potentially significantly reduce the false alarming rate and false attacks in IDS classification. One limitation of our paper is that we only tested on available datasets and not in a practical context, which we will aim to achieve in the future.

In future research, we will try to deploy our proposed HRC strategy in practical IDS and investigate developing this into an even better framework for IDS that examines lighter models but still raises the prediction accuracy to a greater extent through reinforcement learning algorithms. Our secondary aim is to create datasets that can emphasize a large area of wireless communication from wireless networks to cellular networks (such as 5G) so that these datasets can be used to test the currently proposed approaches for IDS and, furthermore, contribute great material for future across different fields.

Author Contributions: Conceptualization, K.D.N.D., M.V. and P.F.; methodology, K.D.N.D.; software K.D.N.D.; validation K.D.N.D., M.V. and P.F.; formal analysis K.D.N.D.; investigation, K.D.N.D.; resources M.V.; data curation, K.D.N.D.; writing—original draft preparation, K.D.N.D.; writing—review and editing M.V. and P.F.; visualization K.D.N.D.; supervision, P.F. and M.V.; project administration, P.F. and M.V.; funding acquisition, M.V. All authors have read and agreed to the published version of the manuscript.

Funding: The research was co-funded by the European Union (EU) within the REFRESH project—Research Excellence For REgion Sustainability and High-tech Industries ID No. CZ.10.03.01/00/22_003/0000048 of the European Just Transition Fund and also supported by the Ministry of Education, Youth and Sports of the Czech Republic (MEYS CZ) within a Student Grant Competition in the VSB-Technical University of Ostrava under project ID No. SGS SP2024/061.

Data Availability Statement: The dataset used for the experiment in this contribution is an extraction from the AWID3 dataset [41]. The programs and datasets used in the experiments can be found at the Google Drive link: <https://drive.google.com/drive/folders/1xDKy9HM1k2aZaPYOf6zC6KH0MsX6F2ab?usp=sharing>, accessed on 24 July 2024.

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

Abbreviations

The following abbreviations are used in this manuscript:

AWID3	Aegean Wifi Intrusion Dataset 3
IDS	Intrusion Detection Systems
HRC	Hybrid Regression–Classification
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
CNN1D	Convolutional Neural Network for 1-Dimensional Data
DDoS	Distributed Denial of Service
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ReLU	Rectifier Linear Unit
Conv1D	1-Dimension Convolutional Layer
WSP	Website Spoofing
ETW	Evil Twin
BN	Bot Net
MW	Malware

References

- Gentile, A.F.; Fazio, P.; Miceli, G. A Survey on the Implementation and Management of Secure Virtual Private Networks (VPNs) and Virtual LANs (VLANs) in Static and Mobile Scenarios. *Telecom* **2021**, *2*, 430–445. [\[CrossRef\]](#)
- Nguyen, T.N.; Tu, L.T.; Fazio, P.; Van Chien, T.; Le, C.V.; Binh, H.T.T.; Voznak, M. On the Dilemma of Reliability or Security in Unmanned Aerial Vehicle Communications Assisted by Energy Harvesting Relaying. *IEEE J. Sel. Areas Commun.* **2024**, *42*, 52–67. [\[CrossRef\]](#)
- Quincozes, S.E.; Albuquerque, C.; Passos, D.; Mossé, D. A survey on intrusion detection and prevention systems in digital substations. *Comput. Netw.* **2021**, *184*, 107679. [\[CrossRef\]](#)
- Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [\[CrossRef\]](#)
- Cyril, O.O.; Elmissaoui, T.; Okoronkwo, M.C.; Ihedioha, U.; Ugwuishiwu, C.H.; Onyebuchi, O.B. Signature based Network Intrusion Detection System using Feature Selection on Android. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 0110667. [\[CrossRef\]](#)
- Einy, S.; Oz, C.; Dorostkar Navaei, Y. The Anomaly- and Signature-Based IDS for Network Security Using Hybrid Inference Systems. *Math. Probl. Eng.* **2021**, *2021*, 1–10. [\[CrossRef\]](#)
- Al-Qarni, E.; Al-Asmari, G. Addressing Imbalanced Data in Network Intrusion Detection: A Review and Survey. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 0150215. [\[CrossRef\]](#)
- Mohammad, R.; Saeed, F.; Almazroi, A.A.; Alsubaei, F.S.; Almazroi, A.A. Enhancing Intrusion Detection Systems Using a Deep Learning and Data Augmentation Approach. *Systems* **2024**, *12*, 79. [\[CrossRef\]](#)
- Veeramreddy, J.; Prasad, K.M. *Anomaly-Based Intrusion Detection System*; IntechOpen: London, UK, 2019. [\[CrossRef\]](#)
- Assy, A.T.; Mostafa, Y.; El-khaleq, A.A.; Mashaly, M. Anomaly-Based Intrusion Detection System using One-Dimensional Convolutional Neural Network. *Procedia Comput. Sci.* **2023**, *220*, 78–85. [\[CrossRef\]](#)
- Alhasan, S.; Abdul-Salaam, G.; Missah, Y.; Anisi, M. Hybrid Network Intrusion Detection Systems: A Systematic Review. *Sci. Pract. Cyber Secur. J.* **2024**, *7*, 1–35.
- Qiu, W.; Ma, Y.; Chen, X.; Yu, H.; Chen, L. Hybrid intrusion detection system based on Dempster-Shafer evidence theory. *Comput. Secur.* **2022**, *117*, 102709. [\[CrossRef\]](#)
- Zhao, R.; Mu, Y.; Zou, L.; Wen, X. A Hybrid Intrusion Detection System Based on Feature Selection and Weighted Stacking Classifier. *IEEE Access* **2022**, *10*, 71414–71426. [\[CrossRef\]](#)
- Chen, Z.; Simsek, M.; Kantarci, B.; Bagheri, M.; Djukic, P. Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. *Comput. Netw.* **2024**, *250*, 110576. [\[CrossRef\]](#)
- Tsai, C.F.; Hsu, Y.F.; Lin, C.Y.; Lin, W.Y. Intrusion detection by machine learning: A review. *Expert Syst. Appl.* **2009**, *36*, 11994–12000. [\[CrossRef\]](#)
- Ablat, H. Study on Multi-grade Intrusion Detection Model Based on Data Mining Technology. In Proceedings of the 2011 10th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Wuxi, China, 14–17 October 2011; pp. 259–265. [\[CrossRef\]](#)
- Kajal, A.; Nandal, S.K. A hybrid approach for cyber security: improved intrusion detection system using Ann-Svm. *Indian J. Comput. Sci. Eng.* **2020**, *11*, 325–412. [\[CrossRef\]](#)
- Wazirali, R. An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. *Arab. J. Sci. Eng.* **2020**, *45*, 10859–10873. [\[CrossRef\]](#)

19. Kolukisa, B.; Dedetürk, B.K.; Hacilar, H.; Gungor, V.C. An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm. *Comput. Stand. Interfaces* **2024**, *89*, 103808. [\[CrossRef\]](#)
20. Liang, X.; Jiang, A.; Li, T.; Xue, Y.; Wang, G. LR-SMOTE—An improved unbalanced data set oversampling based on K-means and SVM. *Knowl.-Based Syst.* **2020**, *196*, 105845. [\[CrossRef\]](#)
21. Shi, Q.; Kang, J.; Wang, R.; Yi, H.; Lin, Y.; Wang, J. A Framework of Intrusion Detection System based on Bayesian Network in IoT. *Int. J. Perform. Eng.* **2018**, *14*, 2280. [\[CrossRef\]](#)
22. Aburomman, A.A.; Reaz, M.B.I. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Comput. Secur.* **2017**, *65*, 135–152. [\[CrossRef\]](#)
23. Abdallah, E.E.; Eleisah, W.; Ootom, A.F. Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey. *Procedia Comput. Sci.* **2022**, *201*, 205–212. [\[CrossRef\]](#)
24. Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M. Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Comput. Sci.* **2020**, *171*, 1251–1260. [\[CrossRef\]](#)
25. Ghanem, W.A.; Jantan, A. A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm. *Neural Comput. Appl.* **2020**, *32*, 11665–11698. [\[CrossRef\]](#)
26. Bilot, T.; Madhoun, N.E.; Agha, K.A.; Zouaoui, A. Graph Neural Networks for Intrusion Detection: A Survey. *IEEE Access* **2023**, *11*, 49114–49139. [\[CrossRef\]](#)
27. Zhong, M.; Lin, M.; Zhang, C.; Xu, Z. A Survey on Graph Neural Networks for Intrusion Detection Systems: Methods, Trends and Challenges. *Comput. Secur.* **2024**, *141*, 103821. [\[CrossRef\]](#)
28. Al-Milli, N.R.; Al-Khassawneh, Y.A. Intrusion Detection System using CNNs and GANs. *WSEAS Trans. Comput. Res.* **2024**, *12*, 281–290. [\[CrossRef\]](#)
29. Mohammadpour, L.; Ling, T.C.; Liew, C.S.; Aryanfar, A. A survey of CNN-based network intrusion detection. *Appl. Sci.* **2022**, *12*, 8162. [\[CrossRef\]](#)
30. ElSayed, M.S.; Le-Khac, N.A.; Albahar, M.A.; Jurcut, A. A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *J. Netw. Comput. Appl.* **2021**, *191*, 103160. [\[CrossRef\]](#)
31. Gautam, S.; Henry, A.; Zuhair, M.; Rashid, M.; Javed, A.R.; Maddikunta, P.K.R. A composite approach of intrusion detection systems: Hybrid RNN and correlation-based feature optimization. *Electronics* **2022**, *11*, 3529. [\[CrossRef\]](#)
32. Ibrahim, M.; Elhafiz, R. Modeling an intrusion detection using recurrent neural networks. *J. Eng. Res.* **2023**, *11*, 100013. [\[CrossRef\]](#)
33. Laghrissi, F.; Douzi, S.; Douzi, K.; Hssina, B. Intrusion detection systems using long short-term memory (LSTM). *J. Big Data* **2021**, *8*, 65. [\[CrossRef\]](#)
34. Chen, A.; Fu, Y.; Zheng, X.; Lu, G. An efficient network behavior anomaly detection using a hybrid DBN-LSTM network. *Comput. Secur.* **2022**, *114*, 102600. [\[CrossRef\]](#)
35. Imrana, Y.; Xiang, Y.; Ali, L.; Abdul-Rauf, Z. A bidirectional LSTM deep learning approach for intrusion detection. *Expert Syst. Appl.* **2021**, *185*, 115524. [\[CrossRef\]](#)
36. Jain, S.; Pawar, P.M.; Muthalagu, R. Hybrid intelligent intrusion detection system for internet of things. *Telemat. Inform. Rep.* **2022**, *8*, 100030. [\[CrossRef\]](#)
37. Halbouni, A.; Gunawan, T.S.; Habaebi, M.H.; Halbouni, M.; Kartiwi, M.; Ahmad, R. CNN-LSTM: Hybrid deep neural network for network intrusion detection system. *IEEE Access* **2022**, *10*, 99837–99849. [\[CrossRef\]](#)
38. Khan, M.A.; Karim, M.R.; Kim, Y. A scalable and hybrid intrusion detection system based on the convolutional-LSTM network. *Symmetry* **2019**, *11*, 583. [\[CrossRef\]](#)
39. Psychogios, K.; Papadakis, A.; Bourou, S.; Nikolaou, N.; Maniatis, A.; Zahariadis, T. Deep Learning for Intrusion Detection Systems (IDSs) in Time Series Data. *Future Internet* **2024**, *16*, 73. [\[CrossRef\]](#)
40. Hand, D.; Christen, P.; Kirielle, N. F*: An interpretable transformation of the F-measure. *Mach. Learn.* **2021**, *110*, 451–456. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Chatzoglou, E.; Kambourakis, G.; Kolias, C. Empirical Evaluation of Attacks Against IEEE 802.11 Enterprise Networks: The AWID3 Dataset. *IEEE Access* **2021**, *9*, 34188–34205. [\[CrossRef\]](#)
42. Aminanto, A.E.; Aminanto, M.E. Deep learning models for intrusion detection in Wi-Fi networks: A literature survey. In *Sustainable Architecture and Building Environment: Proceedings of ICSDAMS 2020*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 115–121.
43. Leevy, J.L.; Khoshgoftaar, T.M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *J. Big Data* **2020**, *7*, 104. [\[CrossRef\]](#)
44. Kumar, S.; Gupta, S.; Arora, S. Research Trends in Network-Based Intrusion Detection Systems: A Review. *IEEE Access* **2021**, *9*, 157761–157779. [\[CrossRef\]](#)
45. Dhanabal, L.; Shantharajah, S. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.
46. Al-Tobi, A.; Duncan, I. KDD 1999 generation faults: A review and analysis. *J. Cyber Secur. Technol.* **2018**, *2*, 1–37. [\[CrossRef\]](#)
47. Wilson, D.L.R. *Towards Effective Wireless Intrusion Detection Using AWID Dataset*; Rochester Institute of Technology: Rochester, NY, USA, 2021.
48. Wu, T.; Fan, H.; Zhu, H.; You, C.; Zhou, H.; Huang, X. Intrusion detection system combined enhanced random forest with SMOTE algorithm. *EURASIP J. Adv. Signal Process.* **2022**, *2022*, 39. [\[CrossRef\]](#)

49. Telo, J. Intrusion detection with supervised machine learning using smote for imbalanced datasets. *J. Artif. Intell. Mach. Learn. Manag.* **2021**, *5*, 12–24.
50. Puri, A.; Kumar Gupta, M. Improved hybrid bag-boost ensemble with K-means-SMOTE-ENN technique for handling noisy class imbalanced data. *Comput. J.* **2022**, *65*, 124–138. [\[CrossRef\]](#)
51. Abdel-Basset, M.; Hawash, H.; Chakraborty, R.K.; Ryan, M.J. Semi-Supervised Spatiotemporal Deep Learning for Intrusions Detection in IoT Networks. *IEEE Internet Things J.* **2021**, *8*, 12251–12265. [\[CrossRef\]](#)
52. Heidari, A.; Jafari Navimipour, N.; Unal, M. A Secure Intrusion Detection Platform Using Blockchain and Radial Basis Function Neural Networks for Internet of Drones. *IEEE Internet Things J.* **2023**, *10*, 8445–8454. [\[CrossRef\]](#)
53. Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions. *Comput. Commun.* **2021**, *195*, 346–361. [\[CrossRef\]](#)
54. Lee, B.S.; Kim, J.W.; Choi, M.J. Federated Learning Based Network Intrusion Detection Model. In Proceedings of the 2023 24th Asia-Pacific Network Operations and Management Symposium (APNOMS), Sejong, Republic of Korea, 6–8 September 2023; pp. 330–333.
55. Li, J.; Tong, X.; Liu, J.; Cheng, L. An Efficient Federated Learning System for Network Intrusion Detection. *IEEE Syst. J.* **2023**, *17*, 2455–2464. [\[CrossRef\]](#)
56. Qazi, E.U.H.; Faheem, M.H.; Zia, T. HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System. *Appl. Sci.* **2023**, *13*, 4921. [\[CrossRef\]](#)
57. Zhang, H.; Huang, L.; Wu, C.Q.; Li, Z. An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. *Comput. Netw.* **2020**, *177*, 107315. [\[CrossRef\]](#)
58. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A Multitiered Hybrid Intrusion Detection System for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 616–632. [\[CrossRef\]](#)
59. Bach, M.; Werner, A.; Palt, M. The Proposal of Undersampling Method for Learning from Imbalanced Datasets. *Procedia Comput. Sci.* **2019**, *159*, 125–134. [\[CrossRef\]](#)
60. Yu, L.; Zhou, N. Survey of Imbalanced Data Methodologies. *arXiv* **2021**, arXiv:2104.02240.
61. Fu, G.; Wang, J.B.; Zong, M.J.; Yi, L. Feature Ranking and Screening for Class-Imbalanced Metabolomics Data Based on Rank Aggregation Coupled with Re-Balance. *Metabolites* **2021**, *11*, 389. [\[CrossRef\]](#)
62. Dang, K.; Fazio, P.; Vozňák, M. *High-Speed Users' Mobility Prediction Scheme Based on Deep Learning for Small Cell and Femtocell Networks*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 446–458. [\[CrossRef\]](#)
63. Ampomah, E.; Qin, Z.; Nyame, G. Evaluation of Tree-Based Ensemble Machine Learning Models in Predicting Stock Price Direction of Movement. *Information* **2020**, *11*, 332. [\[CrossRef\]](#)
64. Parhi, R.; Nowak, R.D. The Role of Neural Network Activation Functions. *IEEE Signal Process. Lett.* **2020**, *27*, 1779–1783. [\[CrossRef\]](#)
65. Lillicrap, T.P.; Santoro, A. Backpropagation through time and the brain. *Curr. Opin. Neurobiol.* **2019**, *55*, 82–89. [\[CrossRef\]](#)
66. Kuo, W.C.; Chen, C.H.; Chen, S.Y.; Wang, C.C. Deep Learning Neural Networks for Short-Term PV Power Forecasting via Sky Image Method. *Energies* **2022**, *15*, 4779. [\[CrossRef\]](#)
67. Vlachas, P.; Pathak, J.; Hunt, B.; Sapsis, T.; Girvan, M.; Ott, E.; Koumoutsakos, P. Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **2020**, *126*, 191–217. [\[CrossRef\]](#) [\[PubMed\]](#)
68. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [\[CrossRef\]](#)
69. Helmud, E.; Fitriyani, F.; Romadiana, P. Classification Comparison Performance of Supervised Machine Learning Random Forest and Decision Tree Algorithms Using Confusion Matrix. *J. Sisfokom (Sist. Inf. Dan Komput.)* **2024**, *13*, 92–97. [\[CrossRef\]](#)
70. Corona, I.; Giacinto, G.; Roli, F. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Inf. Sci.* **2013**, *239*, 201–225. [\[CrossRef\]](#)
71. Saied, M.; Guirguis, S.; Madbouly, M. A Comparative Study of Using Boosting-Based Machine Learning Algorithms for IoT Network Intrusion Detection. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 177. [\[CrossRef\]](#)
72. Gbashi, E.; Mohammed, B. Intrusion Detection System for NSL-KDD Dataset Based on Deep Learning and Recursive Feature Elimination. *Eng. Technol. J.* **2021**, *39*, 1069–1079. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.