applied sciences

MDPI

Article

A Combined Multi-Classification Network Intrusion Detection System Based on Feature Selection and Neural Network Improvement

Yunhui Wang [1,2,†], Zifei Liu [3,†], Weichu Zheng [3], Jinyan Wang [1,2], Hongjian Shi [3,*] and Mingyu Gu [4]

[1] National Key Laboratory of Science and Technology on Avionics System Integration, Shanghai 200233, China; wang.yh@outlook.com (Y.W.); wangjy121@avic.com (J.W.)
[2] China National Aeronautical Radio Electronics Research Institute, Shanghai 200233, China
[3] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; liuzifei@sjtu.edu.cn (Z.L.); sjtu_zwc0518@sjtu.edu.cn (W.Z.)
[4] Sino-European School of Technology, Shanghai University, Shanghai 200244, China; 22124558@shu.edu.cn
* Correspondence: shhjwu5@sjtu.edu.cn; Tel.: +86-137-1795-9365
† These authors contributed equally to this work.

Featured Application: Increased parallelism in edge network security issues, Feature loss handling.

Abstract: Feature loss in IoT scenarios is a common problem. This situation poses a greater challenge in terms of real-time and accuracy for the security of intelligent edge computing systems, which also includes network security intrusion detection systems (NIDS). Losing some packet information can easily confuse NIDS and cause an oversight of security systems. We propose a novel network intrusion detection framework based on an improved neural network. The new framework uses 23 subframes and a mixer for multi-classification work, which improves the parallelism of NIDS and is more adaptable to edge networks. We also incorporate the K-Nearest Neighbors (KNN) algorithm and Genetic Algorithm (GA) for feature selection, reducing parameters, communication, and memory overhead. We named the above system as Combinatorial Multi-Classification-NIDS (CM-NIDS). Experiments demonstrate that our framework can be more flexible in terms of the parameters of binary classification, has a fairly high accuracy in multi-classification, and is less affected by feature loss.

Keywords: feature loss; network intrusion detection systems; multi-classification; attack-type identification

Citation: Wang, Y.; Liu, Z.; Zheng, W.; Wang, J.; Shi, H.; Gu, M. A Combined Multi-Classification Network Intrusion Detection System Based on Feature Selection and Neural Network Improvement. *Appl. Sci.* **2023**, *13*, 8307. https://doi.org/10.3390/app13148307

Academic Editor: Christos Bouras

Received: 12 June 2023
Revised: 4 July 2023
Accepted: 14 July 2023
Published: 18 July 2023

1. Introduction

Smart edge computing systems are distributed computing architectures that bring computing and data processing power closer to the network edge from traditional centralized cloud computing environments [1]. The design goal of smart edge computing systems is to place computing resources as close as possible to the data generation source or data usage endpoint to provide a lower latency, reduce network bandwidth pressure, and enhance user experience.

Smart edge computing systems face important challenges. Security and privacy have become more complex and critical in distributed environments [2,3]. Protecting systems from the threat of cyber attacks, data breaches, and malicious behavior is critical. At the same time, ensuring the security and privacy of data during transmission and storage, as well as effective authentication, access control, and vulnerability patching mechanisms, are key measures to ensure the security and privacy of smart edge computing systems.

One security measure in smart edge computing systems is NIDS (network intrusion detection system), which is used to monitor and detect potential intrusions in network traffic. NIDS detects intrusion attacks before they cause harm to the system and uses the

alarm and protection system to expel them [4]. After the attack, information about the intrusion attack can be collected and added to the knowledge base as a prevention system, thus enhancing prevention capability [5]. NIDS should provide a better picture of the system and facilitate establishing a system security system.

A large proportion of NIDS is based on machine learning for algorithm design. Machine learning methods have been used to try to improve IDSs to detect malicious communications [6]. In addition, benchmark datasets on malware and cyber attacks have been published to provide IDS research [7–9]. Deep neural networks [10,11] can learn complex causal relationships by training large amounts of data; after training is complete, detection can be performed quickly, regardless of the amount of training data [12].

NIDS consists of several stages: data collection, pre-processing, FS (feature selection), and classification. Recently, attention has been focused on FS and optimization techniques to select basic features. Feature selection has proven to be an effective method to prepare large amounts of dimensional data for machine learning and data mining [13]. The FS phase is critical because NIDS must process large amounts of data, and the direct use of raw feature sets leads to a significant decrease in efficiency [14]. At the same time, edge computing systems in IoT environments may not be able to provide the full functionality and features of certain devices due to the diversity of devices, network connectivity, and resource constraints. By performing feature selection, features that are irrelevant or unimportant for the data analysis task in a particular scenario can be eliminated. This reduces the feature dimensionality and reduces the computational complexity of model training and inference, while also reducing sensitivity to feature incompleteness.

However, the detection rate of NIDS also decreases due to the loss of some features caused by FS. This is unacceptable for IoT environments with high traffic [15]. Also, current NIDSs mainly support binary classification; they only determine whether a packet is anomalous or not. For multi-ecological network systems, the large number of data values leads to systems that often have multiple security measures. Binary classification is challenging in informing and guiding other security deployments, which is very detrimental to systems in the IoT.

In this paper, we propose a new network intrusion detection framework named CM-NIDS, which is divided into two main parts: pre-processing and multi-classification. In the pre-processing part, we filter the dataset and select features based on genetic algorithms (GA) to obtain the feature set. In the multi-classification part, we use K-Nearest Neighbor (KNN) to determine the type of packets and an innovative combinatorial architecture based on a neural network to determine the particular type of data packet.

The main contributions of this work are as follows:

(1) Introduction of CM-NIDS: This work presents a new network intrusion detection framework called Combined Multi-Classification-NIDS (CM-NIDS). This framework is based on an improved neural network and supports other security systems by accurately matching the attack type of the packets. The introduction of CM-NIDS addresses the need for effective network intrusion detection and enhances overall system security and parallelism.

(2) KNN-GA-Based Feature Selection: This work proposes a feature selection approach using a combination of the K-Nearest Neighbors (KNN) algorithm and Genetic Algorithm (GA). This approach significantly reduces the number of parameters the neural network uses, leading to reduced communication and memory overhead. Additionally, it helps mitigate the impact of feature incompleteness, thereby improving the model's robustness and performance. Introducing this feature selection technique contributes to more efficient and effective network intrusion detection.

(3) Data Pre-Processing and Feature Selection: This work emphasizes the importance of data pre-processing methods and specifically highlights the significance of feature selection. The authors provide detailed insights into the pre-processing techniques employed, which contribute to improving the quality and relevance of the input data.

(4) Comparative Evaluation: This work evaluates the proposed CM-NIDS framework alongside existing frameworks. The evaluation demonstrates that CM-NIDS achieves higher attack-type matching rates than other methods.

The remainder of this paper is as follows. Section 2 describes the methodology of our proposed CM-NIDS approach. Section 3 illustrates the results and discussion. Section 4 concludes this research work.

2. Related Works

The Internet of Things (IoT) is a transformative technology that connects everyday objects to the internet, enabling them to send and receive data. It involves a vast network of interconnected devices, ranging from simple sensors and actuators to complex machinery, vehicles, and even smart homes and cities. In recent years, as networks have grown, the data traffic from IoT has created challenges for security maintenance. Waseem et al. [16] discuss the current scale and security challenges facing the IoT, provide a detailed analysis of the attacks emerging at each layer of the IoT, and argue for the need for a comprehensive security framework. Ramalingam et al. [17] think that the rapidly expanding volume of computerized matter poses many research challenges for efficiently storing, processing, and viewing large quantities.

A network intrusion detection system (NIDS) detects behavior that compromises the security of a computer system by collecting information. Its ultimate aim is to identify potential attacks from the flow of messages on the network. Ennaji et al. [18] propose five collaborative learning models for the IDS problem, aiming at classifying network traffic and achieving optimal results. This is based on a stacked integrated learning technique considering some essential machine learning algorithms. However, they only consider 10 features that do not meet the needs of real situations.

Over the past decade, researchers have proposed various ML- [19] and DL-based techniques to improve the efficacy of NIDS in identifying malicious threats. Nevertheless, the significant increase in network traffic and associated attack vectors has created challenges for NIDS systems to identify malicious intrusions. Tosin et al. [19] applied four machine learning models in parallel, K-Nearest Neighbor (K-NN), Parsimonious Bayes (NB), Logistic Regression (LR), and Artificial Neural Network (ANN) with multilevel feature selection methods, and compared multiple metrics to determine which model has the best detection capability in terms of accuracy. Desale et al. [20] used the Genetic Algorithm (GA) to optimize a feature analysis, reducing the number of features selected and the cost of feature maintenance.

The IoT environment is prone to feature loss problems due to network transmission problems, energy limitations, sensor failures, and data collection and storage problems, which can be solved with feature selection. Recent research on NIDS has partly focused on FS (feature selection) techniques and optimization techniques to select the most important features. Jyoti et al. [21] provide an overview of feature selection methods used in NIDS, describe existing feature selection frameworks and classifications for NIDS, and argue that there is an urgent need for new FS to meet the demands of big data. Su et al. [22] consider the feature redundancy of FS and propose a learning automata approach to select optimal and salient features for network traffic intrusion detection. Heather et al. [23] pre-process feature detection, and features were pre-screened based on an existing and admittedly more comprehensive dataset to improve the efficiency of machine learning.

3. Materials and Methods

3.1. Framework

Our work is divided into two main modules, pre-processing and multi-classification. The main purpose of pre-processing is to perform feature selection to reduce the impact of feature loss on NIDS and reduce the workload of NIDS, improve efficiency, and reduce memory usage. Multi-classification is based on neural networks to improve the parallelism and accuracy of NIDS. Their organizational structure is shown in Figure 1.

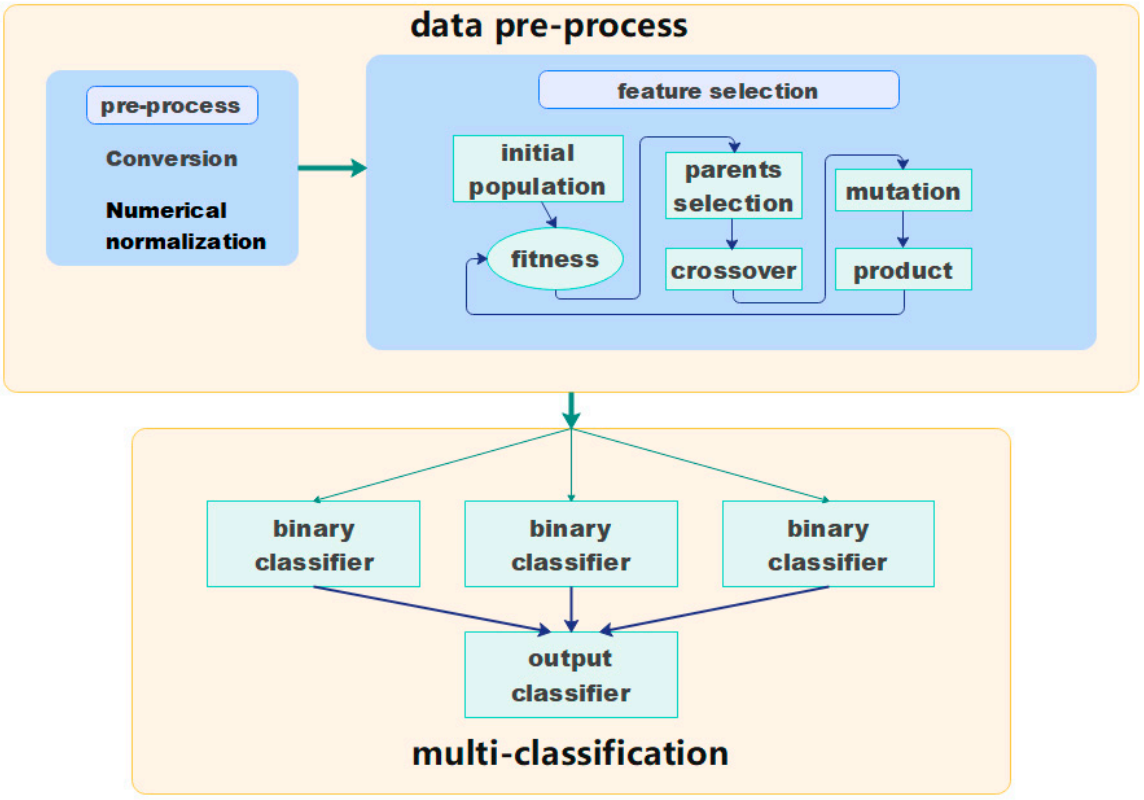

Figure 1. The overall framework of our model. The yellow parts are the two important components of our model, data pre-processing and multi-classification, respectively. The two components of data pre-processing are shown in blue, the process of multi-classification is shown in green boxes, and the arrows indicate the data flow direction.

Pre-processing includes two parts: data pre-processing and feature selection. The output of pre-processing is the filtered dataset and the selected feature set. The work of multi-classification is taking the previous dataset and determining the specific type of attack; its final output is the type of attack for all abnormal packets.

3.2. Pre-Processing

We used the representative KDDCup99 dataset [24], which is a stream-based dataset with additional information from packet-based data or host-based log files. These data are available in a total dataset volume and 10% data volume. They have 41 features with five distinct attack classes. The specific types of attacks they include are shown in Table 1.

Table 1. Specific attack types in KDDCup99, corresponding to five attack classes.

Attack Categories	Specific Attack Types
DoS	Back, land, Neptune, pod, smurf, teardrop
Probe	Ipsweep, nmap, portsweep, satan
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R	buffer overflow, loadmodule, perl, rootkit

For data pre-processing, our input is the initial dataset, and we obtained numerical datasets that can be used directly for FS through pre-processing. The pre-processing of the dataset consists of two steps: the conversion of character-based features to numeric features (the next part we will refer to as the conversion), and numerical normalization. In the conversion part, we performed the following: convert all feature types (including normal) into numeric identifiers.

In the numerical normalization part, we first calculated the mean and mean absolute error for each attribute using the following formula:

$$\bar{x}_k = \frac{1}{n}\sum_{i=1}^{n} x_{ik} \tag{1}$$

$$S_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ik} - \bar{x})^2} \tag{2}$$

where x_k denotes the mean value of the kth attribute, S_k denotes the mean absolute error of the kth attribute, and x_k denotes the kth attribute of the i-th record.

Then, we apply the normalization metric to each data record:

$$Z_{ik} = \frac{x_{ik} - \bar{x}_k}{S_k} \tag{3}$$

where Z_{ik} denotes the kth attribute value of the i-th data record after normalization.

We then normalize each value to the interval [0, 1] as follows:

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4}$$

where x_{max} is the maximum value of the sample data, x_{min} is the minimum value of the sample data, and x is the previously obtained data. In this way, the data pre-processing gives us a numerical, normalized dataset suitable for FS use.

For feature selection, we used feature selection to obtain the feature set that can be used for classification. FS is the method of selecting unique features based on specific criteria in machine learning. For a dataset, three components do not contribute to feature selection, which we call noise. These noises are irrelevant, redundant, and noisy. We distinguish between these three types of noise by using Table 2 as follows.

Table 2. Different feature components during feature selection.

Feature Type	Description	Accuracy	Training Time
Strongly Relevant	It is necessary for an optimal feature subset.	High	Low
Irrelevant	It cannot help discriminate between supervised or unsupervised data.	Low	High
Redundant	It can be completely replaced with a set of other features.	Low	High
Noisy	It is not relevant to the learning or mining task.	Low	High

In feature selection, feature subsets are selected from the original feature set based on feature redundancy and relevance [25]. We considered GA-KNN as our algorithm. In machine learning, one of the uses of Genetic Algorithms is to obtain the correct number of variables to create predictive models. The use of GA for FS has been more widely discussed and researched [26,27]. Feature selection techniques are used to accurately identify relevant subsets of features, reducing the number of dimensions to improve overall efficiency. Feature selection helps reduce the possibility of overfitting while reducing the effect of feature loss by focusing on certain features.

In this process, the addition of KNN improves the efficiency of GA. This is because it can artificially enhance the composition of the population by selecting new members that are closer to quality individuals as much as possible.

The pseudocode of GA is as Algorithm 1.

Algorithm 1. GA for feature selection

```
Initialize population P from the dataset randomly
for each iteration do
    Randomly select a subset of the population P as Psub
    Crossing operation:
    for each pair of individual(P^i,P^{i+1}) in Psub do
    Generate a random probability p
    if Pc > p then
        randomly choose position P_a.
        Exchange the genes
            P^i = [ ... , P_a^{i+1}, P_{a+1}^{i+1}, P_{a+2}^{i+1}, ... ]
            P^{i+1} = [ ... , P_a^i, P_{a+1}^i, P_{a+2}^i, ... ]
            end if
      end for
      Mutation operation:
      for each individual P^i in Psub do
            Generate a random probability p
            if Pm > p then
                Random change a gene in Pi with a random device
            end if
        end for
      Reverse operation:
        for each individual P^i in Psub do
              Randomly choose two positions Pa and Pb where Pa < Pb
              Reverse the genes between Pa and Pb to get P^{i'}
              if f (P^{i'}) > f (P^i) then
                  P^i = P^{i'}
              end if
        end for
        Add Psub into population P.
        Calculate the fitness values f (P^i) = 1/L[P^i] for all individuals in P
        Sorted P by descending order of fitness value.
        Keep the first several individuals in P according to population size
    end for
    return The best feature set S^best = P^0
```

GA simulates natural selection and inheritance. Continuous random selection, crossover, and mutation operations generate individuals better suited to the environment, converging into the fittest individuals. This approach yields quality solutions to problems. The 41 chromosome features are converted into binary code for machine learning algorithms. Each individual's genotype consists of 41 genes. In GA chromosome coding, binary strings (1 s and 0 s) represent selected and non-selected feature subsets. Random binary bit assignment ensures population diversity.

For functional selection, the proposed enhancement of GA in feature selection starts with the training data as the input and then generates the feature chromosomes. Each chromosome will be evaluated. The chromosome with the highest classification accuracy will be selected and considered as the optimized feature chromosome. The next step is to use the optimized feature chromosomes from the training data to train the KNN (K-Nearest Neighbor). The KNN algorithm is a basic classification and regression algorithm—a method in supervised learning. Given a training set M and a test object n, where the object is a vector with an attribute value and unknown category label. The distance (typically Euclidean distance) or similarity (typically cosine similarity) between object m and each object in the training set is calculated. The nearest neighbors are identified, and the top K categories with the highest frequency among them are assigned to the test object z. This classifier helps identify network activity classes as normal or abnormal using the calculated accuracy.

On the other hand, test data are used to evaluate the performance of the classifier to check whether the trained algorithm is able to generalize new information based on the evaluation results encountered from the training data. This process will improve the new generation of chromosomes without sacrificing population diversity.

The selection operation ensures that the best chromosomes representing the best solution will be selected as parents to produce better-performing offspring for the next generation. This study used tournament selection. After selecting the best-performing chromosomes, crossover operations were applied to produce new progeny, resulting in better chromosomes. This is performed by mixing the parents' genes using crossover probabilities (Pc). This study applied a single-point crossover. The mutation is performed with random bit-flip mutations in individual chromosomes, where substrings are randomly selected and flipped using the mutation probability (Pm). Specifically, we randomly selected and reversed a contiguous segment of a selected individual's genes. Here, we used 30 iterations, a crossover rate (Pc) of 0.7, and a variation rate (Pm) of 1/41, i.e., 0.02. The algorithm described above was designed to efficiently select features with the highest accuracy scores in the training data.

3.3. Multi-Classification

For multi-classification, we already obtained the dataset and the feature set. For this section, we obtained the specific type of attack on the set of anomalies. For the sake of description, we consider the normal packet as a particular case of the abnormal packet.

We did not use a neural network directly for the classification of the anomaly dataset. Instead, it was deformed and simplified to some extent. On the one hand, the penalty of error detection is too small for the neural network to provide a good training effect. On the other hand, we simplified the multi-classification step as much as possible; the direct deployment of neural networks is unrealistic due to the parameters and complexity of deploying neural networks at a particular scale [28]. Edge devices usually have a limited computational power and storage resources. Neural networks usually require many computational resources to perform training and inference tasks, especially when the models are large and the number of parameters is large. The computational power of edge devices may need to be improved to meet the high requirements of complex neural networks. Parallel computing can accelerate the training and inference process of neural networks by simultaneously utilizing multiple computing devices or processing units, thus improving overall performance and efficiency. We improved the architecture of the multi-classifier by splitting it into 23 binary classifiers, resulting in the attack set. The framework is shown in Figure 2.

As shown in Figure 2, using neural networks alone for multi-classification means that the packets need to be matched directly for 23 cases, which is demanding regarding the data volume and computing power. We improved the above algorithm by delivering the packets to each of the 23 binary-classification neural networks simultaneously. Each neural network was responsible for determining whether the packet was of one of its corresponding types. We set up a mixer after the 23 neural networks, which aggregated and decided the type of packets.

The results of the sub-models were integrated through the classifier layer. Specifically, each sub-model output a two-dimensional vector representing the probabilities of the corresponding sample belonging to two different classes. These output vectors were processed through the classifier layer, which consisted of a linear layer that mapped the input dimension from 23 to 23.

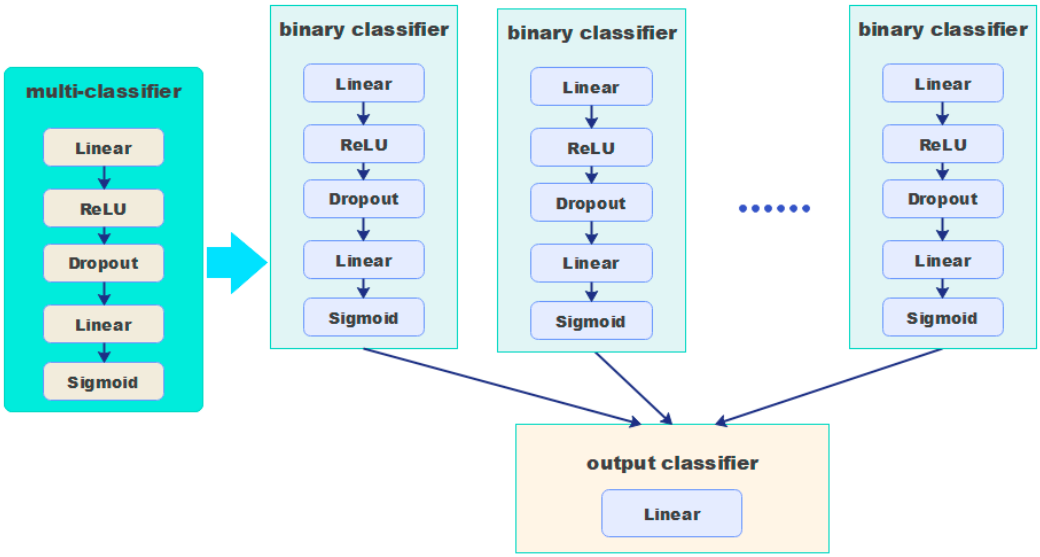

Figure 2. A multi-classification scheme based on binary classification. The left figure shows the general multi-classification model, which uses a multi-classifier directly for attack-type matching based on the extracted feature set and dataset. The figure on the right is a multi-classifier based on 23 binary classifiers. The output classifier will aggregate and filter the results of the binary classifiers.

The pseudocode of CM-NIDS is as Algorithm 2.

The final output vector can be viewed as a prediction for multi-class classification, where each dimension corresponds to a class, and the values indicate the model's confidence or probability for that class.

The integration process is not performed directly within the sub-models but rather through the model's overall structure. The outputs of the sub-models are passed as inputs to the classifier layer, which linearly maps these outputs to generate the final prediction. Through the forward propagation process of the entire network, the input features undergo processing with the feature layer, multiple sub-models, and the classifier layer to obtain the predicted class for the input sample.

Overall, our model leverages multiple sub-models to learn diverse and specialized feature representations, integrates their outputs through a classifier layer for ensemble benefits, allows flexibility in adapting to different tasks and datasets, increases model depth for enhanced expressive capacity, and provides interpretability through a clear delineation of sub-model functionality. These characteristics contribute to an improved performance, broader representation capabilities, and a better understanding and interpretability of the model, advancing the application of neural network models in scientific research.

We obtained specific types of abnormal attacks to facilitate a detailed and reliable reference for other network security systems. Our model can be better adapted to edge networks. First, since the model is split into multiple sub-models, these can be deployed on different nodes or devices of the distributed system, thus enabling parallel computation and processing. Such parallelism can accelerate the inference process of the model and improve the overall computational efficiency and throughput. Second, for edge networks, the structure split into multiple sub-models may be better adapted to the characteristics of the edge environment. Edge networks usually have limited computational resources and bandwidth and must make real-time decisions and reasoning locally. By splitting the model into multiple sub-models, the computation can be distributed to the edge devices, reducing the reliance on the central server and communication overhead. This distributed deployment can improve the real-time responsiveness and reliability of the edge network.

Algorithm 2. NIDS based on neural network with 23 sub-structures

Input: Training set $\mathbf{D} = \{(x^{(n)}, y^{(n)})\}^{N}n = 1$, Validation set **V**, Learning rate $\boldsymbol{\alpha}$, Regularization factor $\boldsymbol{\lambda}$, Number of network layers **L**, Number of neurons $\mathbf{M_l}$.

Initialize **D, V**
repeat
  Random reordering of the samples in the training set **D**
  for n = 1 . . . N **do**
    Select sample (x(n), y(n)) from the training set **D**
    Feedforward calculation of net input $\mathbf{z}^{(l)}$ and activation value $\boldsymbol{\alpha}^{(l)}$ for each layer, up to the last layer
    // Feedforward through feature extraction layers
    for l = 1 **to L − 1 do**
      $\mathbf{z^{(l)} = W^{(l)} * \alpha^{(l-1)} + b^{(l)}}$
      $\mathbf{\alpha^{(l)} = \sigma(z^{(l)})}$
    end for
    // Feedforward through the 23 sub-structures
    for i = 1 to 23 **do**
      $\mathbf{z^{(L,i)} = W^{(L,i)} * \alpha^{(L-1)} + b^{(L,i)}}$
      $\mathbf{\alpha^{(L,i)} = \sigma(z^{(L,i)})}$
    end for
    Backpropagation calculates the error $\boldsymbol{\delta}^{(l)}$ for each layer
    for i = 1 to 23 **do**
      $\mathbf{\delta^{(L,i)} = \partial L(y^{(n)}, \bar{y}^{(n)})/\partial z^{(L,i)}}$
    end for
    for l = L − 1 to 1 **do**
      $\mathbf{\delta^{(l)} = (\partial L(y^{(n)}, \bar{y}^{(n)}))/\partial z^{(l)}) * (W^{(l+1)T} * \delta^{(l+1)})}$
    end for
    for l = 1 to L **do**
      $\partial L(y^{(n)}, \bar{y}^{(n)})/\partial W^{(l)} = \delta^{(l)} * \alpha^{(l-1)T}$
      $\partial L(y^{(n)}, \bar{y}^{(n)})/\partial b^{(l)} = \delta^{(l)}$
    end for
    Update parameters
    for l = 1 to L − 1 **do**
      $\mathbf{W^{(l)} = W^{(l)} - \alpha * \partial L(y^{(n)}, \bar{y}^{(n)})/\partial W^{(l)} - \lambda * W^{(l)}}$
      $\mathbf{b^{(l)} = b^{(l)} - \alpha * \partial L(y^{(n)}, \bar{y}^{(n)})/\partial b^{(l)}}$
    end for
    //Update parameters for the 23 sub-structures
    for i = 1 to 23 **do**
      $\mathbf{W^{(L,i)} = W^{(L,i)} - \alpha * \partial L(y^{(n)}, \bar{y}^{(n)})/\partial W^{(L,i)} - \lambda * W^{(L,i)}}$
      $\mathbf{b^{(L,i)} = b^{(L,i)} - \alpha * \partial L(y^{(n)}, \bar{y}^{(n)})/\partial b^{(L,i)}}$
    end for
  end for
until The accuracy of neural networks no longer decreases on the validation set **V**

Output: W, b

4. Results

4.1. Settings

The dataset we used is KDDCup99 (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, accessed on 2 November 2022), a stream-based dataset enriched with additional information from packet-based data or host-based log files, so its data is adequate and representative.

This experimental section involves three main parts. First, we explain the environment and settings we used; afterward, we compare our method with others regarding accuracy and speed. Finally, regarding carrying out other experiments, we analyze them in light of the above.

We experimented on a seven-device cluster where each device was a GeForce RTX 2080 Ti GPU. The implementation was based on a PyTorch distributed library. In conjunction with the use of the PyTorch distributed library, we simulated the situation in an edge network, with distributed training through multiple GPU devices.

In terms of evaluation indicators, we considered two main aspects. Firstly, for NIDS, its detection and correctness rates for errors are significant. We used TPR, Precision, and Recall to evaluate this. In statistics, True Positives (TP) are the correctly predicted positive values, which means that the value of the actual class is yes and the value of the predicted class is also yes. True Negatives (TN) are the correctly predicted negative values, which means that the actual class's value is no and the predicted class's value is also no. False Positives and False Negatives occur when the actual class contradicts the predicted class. False Positives (FP) indicate the situation when the real class is no and the predicted class is yes. False Negatives (FN) indicate the case when the real class is yes, but the predicted class is no. In the evaluation index of the binary-classification results, we used the following parameters:

$$\text{TPR} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5}$$

$$\text{recision} = \frac{\text{FP}}{\text{TP} + \text{FP}} \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

Furthermore, for multi-classification networks, correctly identifying the type of attack is fundamental. We call this the type detection rate (TDR):

$$\text{TDR} = \frac{\text{TD}}{\text{TOTAL}} \tag{8}$$

TD is the number of packets for which the type of attack was correctly identified. TOTAL is the total number of packets.

It is also essential for NIDS to work efficiently. We compared efficiency as a metric by using the same dataset and recording the running time of each method.

The backbone of the neural networks we used in our experiments is shown in Figure 3.

4.2. Baseline

We evaluated NIDS primarily in terms of accuracy in dichotomous and multi-classification. Firstly, we compared the differences between our method and other NIDSs regarding False Negatives, accuracy, etc. KNN-NIDS [29] uses KNN as the classification algorithm of the project and uses PCA for dimension reduction. LR-NIDS [30], Bayes-NIDS [31], DT-NIDS [32], and AB-NIDS [33] use a Logistic Regression, Naive Bayesian algorithm, Decision Tree algorithm, Ada Boost algorithm, and Random Forest algorithm, respectively, in the NIDS scenario.

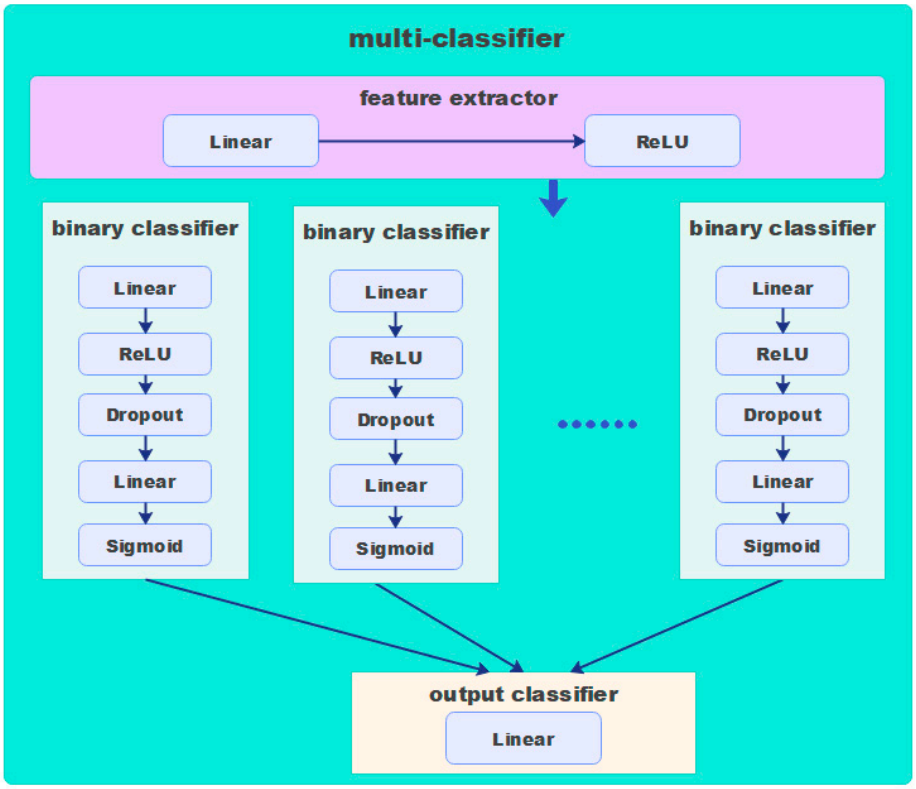

Figure 3. The backbone of the neural network model, used in the experiment of the feature extractor, binary classification, and multi-classification.

In order to ensure good training results and evaluate the classifier's performance, we combined the above considerations. The F1-Score combines two metrics, Precision and Recall, to evaluate the overall performance of the model:

$$F - Sc = \frac{2 \times rcs \times Rcl}{Prcs + Rcl} \tag{9}$$

The higher the value of F-Sc, the better the training effect is proved to be. The results are as Table 3:

Table 3. Comparison of different algorithms in terms of correct packet detection rate.

Methods	TPR	Precision	Recall	F-Sc
CM-NIDS	1.000	1.000	1.000	1.000
KNN-NIDS	0.772	0.157	0.607	0.172
SVM-NIDS	0.788	0.147	0.594	0.153
LR-NIDS	0.619	0.076	0.643	0.135
Bayes-NIDS	0.283	0.588	0.624	0.605
DT-NIDS	0.566	0.082	0.699	0.147
AB-NIDS	0.610	0.076	0.652	0.137

For NIDS, its primary task is to identify whether the packets are anomalous or not, so it has very demanding requirements for the accuracy of binary classification. Our approach (CM-NIDS) has a clear advantage in binary classification. On the TPR metric, CM-NIDS outperforms the optimal algorithm by 21 percentage points, and has a clear advantage on the three remaining commonly used measures of binary classification accuracy (especially on Precision and F-Sc comparisons). This is partly due to our adequate data pre-processing combined with KNN-based feature selection and partly due to our use of a more accurate binary-classification method.

Furthermore, it is fundamental to correctly identify the attack type for multi-classification networks. At the same time, we examined the difference between the model and some classical models in terms of the total tensor used. We were able to see that our model uses a significantly smaller number of tensors. The comparative algorithms we used include APFL [34], Ditto [35], FedBN [36], FedFomo [37], and PerAvg [38]. They are all among more emerging algorithms applicable to NIDS. Our comparison of several algorithms is shown in Table 4.

Table 4. Comparison of different algorithms in terms of TDR and total tensors.

Methods	TDR	Total Tensors
CM-NIDS	0.910	2,100,391
APFL-NIDS	0.500	5,537,952
Ditto-NIDS	0.471	5,537,952
FedBN-NIDS	0.502	2,900,832
FedFomo-NIDS	0.640	6,856,712
PerAvg-NIDS	0.394	2,900,832

In terms of TDR, we compared our approach with other frameworks. It is clear to see that we have an outstanding advantage in terms of attack-type classification. The CM-NIDS algorithm has a TDR value of 0.910, the highest among the provided algorithms. This indicates that the algorithm can classify different data classes more accurately and has a higher detection accuracy in network intrusion detection tasks. Also, the CM-NIDS algorithm uses a total tensor number of 2,100,391, which is less than other algorithms. This may imply that the CM-NIDS algorithm requires less computational and storage resources in the computation process and, thus, is more competitive in terms of efficiency. The smaller number of tensors may imply less memory usage and computational burden and, therefore, it can run more efficiently in the same hardware environment. This is mainly due to our improved approach to multi-classification. It is performed through a combination of binary classification, resulting in a much higher accuracy rate.

4.3. Other Experiments

In terms of parameters and methods, we conducted separate comparative experiments on some of the more critical methods and parameters to demonstrate that our framework is the optimal solution. Our comparison of several algorithms is shown in Table 5.

Table 5. Comparison of different algorithms in terms of accuracy and number of selected features.

Methods	FN	Accuracy
GA + KNN	3	0.991
PCA + KNN	5	0.989
PCA + KNN	10	0.989
PCA + KNN	15	0.990

In the above table, FN refers to the number of selected features, and we compared GA and PCA, KNN, and NN. In terms of run efficiency, KNN was much faster than NN. We have yet to list this set of experiments due to NN's unpredictably long run time. With an equal number of FNs, GA was slightly more accurate than PCA. As the number of FNs increased, the accuracy of PCA + KNN remained essentially constant. However, the FN of GA is the result of the output under the fitness action rather than a pre-set parameter. Therefore, the GA + KNN approach has a more significant advantage in terms of feature selection.

In addition, we deliberately evaluated the robustness of our framework. First, given that the number of parameters is positively correlated with the training effectiveness of the network, we adjusted and retrained the parameters. The results are as Table 6.

Table 6. Comparison of accuracy with the different neural networks with different parameters.

Type	ACR	Type	ACR
True-Set	0.977	APFL-NIDS	0.500
NOM-1	0.909	Ditto-NIDS	0.471
NOM-2	0.885	FedBN-NIDS	0.502
NOM-3	0.790	SVM-NIDS	0.406
KNN-NIDS	0.448		

NOM refers to the number of model parameters. For NOM-1, NOM-2, and NOM-3, the situation of the number of our model parameters is that NOM-2 has the largest number of model parameters, NOM-1 the second largest, and NOM-3 the smallest. It can be noticed that as the number of parameters decreases, the worst case of the number of model parameters remains around 80% and is consistently significantly better than the other frameworks. Even in the case of the NOM-3 dataset, the ACR of our model is about twice that of the other methods.

In a realistic network environment, the number of normal packets will far exceed the abnormal ones. Based on the above premise, we adjusted the ratio of normal to abnormal packets to 10:1, named the True-Set. For the True-Set, binary classification can work more evenly, resulting in better results for multi-classification than the previous training.

To illustrate the rationality of the parameters we set on the neural network, we performed parameter tuning to prove that our solution is optimal. And we examined the performance of our model on other datasets. Figure 4 shows the results of our adjustment of the parameters of Dropout. Here, Dropout = 0.01 is the parameter value we used in experiments.

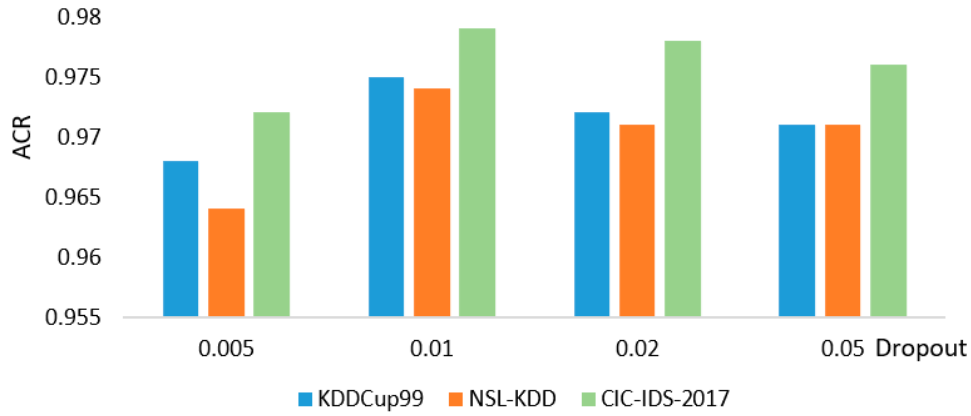

Figure 4. ACR comparison graph for the CM-NIDS algorithm with different Dropout values, where 0.01 was used in our program CM-NIDS.

As shown above, our parameters are the optimal solutions for all cases. At the beginning, there is a significant increase in ACR with the rise of the epoch, but it peaks after the epoch increases to 40. Similarly, the effect of the Dropout parameter on the ACR in the experiment yields the same trend. The main datasets compared are two other classic datasets in the NIDS domain, NSL-KDD (https://www.unb.ca/cic/datasets/nsl.html, accessed on 2 November 2022) and CIC-IDS-2017 (https://www.unb.ca/cic/datasets/ids-2017.html, accessed on 2 November 2022). Overall, the accuracy of our model is stable above 96% under the three datasets, and the multi-classification accuracy of the model is around 97.5% under the optimal parameters, demonstrating the robustness of our model in the multi-classification task. It can be seen that the ACR trends are the same on the three datasets, but the model has a relatively better performance on CIC-IDS-2017, which may be due to the adequate CIC-IDS-2017 dataset.

To illustrate the benefits of using FS, we set up the following experiment and its result is as Table 7:

Table 7. Comparison of NIDS systems with FS and without FS in terms of metrics.

Method	Dataset	With FS?	ACR	Time	Feature No.
CM-NIDS	NOM	YES	0.8144	28.195	3
CM-NIDS	True-Set	YES	0.9737	27.619	3
CM-NIDS	NOM	NO	0.8134	27.313	41
CM-NIDS	True-Set	NO	0.9733	28.766	41

As the time used for the experiments was related to the amount of data, we kept the overall amount of data constant and adjusted the other parameters. As mentioned above, the overall time used remained the same under the different datasets, even though we increased the FS step. At the same time, the number of features was greatly reduced, allowing us to use some of the features for accurate classification without acquiring all of them, in line with the objective reality of the IoT environment and reducing the time required. There was a 92.68% reduction in the number of feature sets after using FS. This also means a smaller memory footprint.

We also performed a comparison of the KNN as well as the two neural network algorithms, the results of which are shown in Table 8.

Table 8. Comparison of accuracy of each algorithm in two cases.

Method	Accuracy	Case
Neural network	100.00%	Binary
Neural network—Single	99.18%	Binary
KNN	99.70%	Binary
Neural network	91.13%	Multi
Neural network—Single	99.15%	Multi
KNN	68.05%	Multi

Neural network—CM refers to the neural network with 23 sub-structures used in our experiments, and Neural network—Single refers to a single neural network. From the above results, it can be seen that the neural network is more accurate than KNN in determining normal/abnormal packets in a multi-classification situation. The binary classification accuracy and multi-classification accuracy of Neural network—CM are both higher than those of Neural network—Single, which proves that our structure is effective.

5. Discussion

We compared our method with others in terms of detection accuracy. In particular, we used a diverse set of accuracy-related metrics. Our binary-classification and multi-classification matching rates outperform other algorithms, especially multi-classification. In addition, we compared the parameters of the neural networks used, the different ways FS is used, the use of FS or not, the different datasets used, and the performance of the models under different datasets. Our framework is very robust.

6. Conclusions

This paper shows work on pre-processing by applying a pre-trained GA-KNN algorithm. First, we use a GA for feature selection to perform dimensionality reduction for attack-type detection, after which we use a KNN algorithm for attack-type matching. This approach can effectively contribute to the attack detection rate of NIDS in terms of multiple classifications, thus informing security systems. Meanwhile, we propose a multi-classification model for anomaly detection based on this and an improved neural network model that provides detailed guidance for other network security systems. We also offer solutions such as modified objective functions for the inherent drawbacks of traditional datasets.

Author Contributions: Conceptualization, Y.W.; methodology, Z.L. and W.Z.; software, Z.L. and W.Z.; validation, Y.W., J.W. and H.S.; formal analysis, Y.W., Z.L. and W.Z.; investigation, Y.W. and J.W.; data curation, H.S.; writing—original draft preparation, Z.L. and W.Z.; writing—review and editing, Y.W., J.W., H.S. and M.G.; visualization, Z.L. and W.Z.; supervision, J.W.; project administration, M.G.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Shanghai Key Laboratory of Scalable Computing and Systems, Internet of Things special subject program, China Institute of IoT (Wuxi), Wuxi IoT Innovation Promotion Center under Grant 2022SP-T13-C, and Industry-university-research Cooperation Funding Project from the Eighth Research Institute in China Aerospace Science and Technology Corporation (Shanghai) under Grant USCAST2022-17.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs. Publicly available datasets were analyzed in this study. This data can be found here: KDDCup99 (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, accessed on 2 November 2022), NSL-KDD (https://www.unb.ca/cic/datasets/nsl.html, accessed on 2 November 2022), and CIC-IDS-2017 (https://www.unb.ca/cic/datasets/ids-2017.html, accessed on 2 November 2022).

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

References

1. Panchali, B. Edge Computing- Background and Overview. In Proceedings of the International Conference on Smart Systems and Inventive Technology, Tirunelveli, India, 13–14 December 2018; pp. 580–582.
2. Shi, H.; Wang, H.; Ma, R.; Hua, Y.; Song, T.; Gao, H.; Guan, H. Robust Searching-based Gradient Collaborative Management in Intelligent Transportation System. *ACM Trans. Multimed. Comput. Commun. Appl.* **2022**. [CrossRef]
3. Cai, Z.; Ren, B.; Ma, R.; Guan, H.; Tian, M.; Wang, Y. A Hardware-Assisted Distributed Framework to Enhance Deep Learning Security. *IEEE Trans. Comput. Soc. Syst.* **2023**, 1–9. [CrossRef]
4. Li, D.; Deng, M.; Wang, H. IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning. *Int. J. Inf. Manag.* **2019**, *49*, 533–545. [CrossRef]
5. Sudar, K.M.; Beulah, M.; Deepalakshmi, P.; Nagaraj, P.; Chinnasamy, P. Detection of Distributed Denial of Service Attacks in SDN using Machine learning techniques. In Proceedings of the International Conference on Computer Communication and Informatics, Coimbatore, India, 27–29 January 2021; pp. 1–5.
6. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the International Converence on Information Systems Security and Privacy, Funchal Madeira, Portugal, 22–24 January 2018; Volume 1, pp. 108–116.
7. Panigrahi, R.; Borah, S. A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. *Int. J. Eng. Technol.* **2018**, *7*, 479–482.
8. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
9. Loye, R.A.Y. Training and testing anomaly-based neural network intrusion detection systems. *Int. J. Inf. Secur. Sci.* **2013**, *2*, 57–63.
10. Sato, H.; Kobayashi, R. A machine learning-based NIDS that collects training data from within the organization and updates the discriminator periodically and automatically. In Proceedings of the International Symposium on Computing and Networking Workshops, Matsue, Japan, 23–26 November 2021; pp. 420–423.
11. Zhang, J.; Hua, Y.; Song, T.; Wang, H.; Xue, Z.; Ma, R.; Guan, H. Improving Bayesian Neural Networks by Adversarial Sampling. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 10110–10117.
12. Alazab, A.; Hobbs, M.; Abawajy, J.; Alazab, M. Using feature selection for intrusion detection system. In Proceedings of the International Symposium on Communications and Information Technologies, Gold Coast, Australia, 2–5 October 2012; pp. 296–301.
13. Guo, H.; Wang, H.; Hua, Y.; Song, T.; Ma, R.; Guan, H. Siren: Byzantine-robust Federated Learning via Proactive Alarming. In Proceedings of the ACM SoCC, Seattle, WA, USA, 1–4 November 2021.
14. Zhang, R.; Chu, X.; Ma, R.; Zhang, M.; Lin, L.; Gao, H.; Guan, H. OSTTD: Offloading of Splittable Tasks with Topological Dependence in Multi-Tier Computing Networks. *IEEE J. Sel. Areas Commun.* **2022**, *41*, 555–568. [CrossRef]
15. Sivasankari, N.; Kamalakkannan, S. Building NIDS for IoT Network using Ensemble Approach. In Proceedings of the International Conference on Communication and Electronics Systems, Coimbatore, India, 22–24 June 2022; pp. 328–333.
16. Iqbal, W.; Abbas, H.; Daneshmand, M.; Rauf, B.; Bangash, Y.A. An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security. *IEEE Internet Things J.* **2020**, *7*, 10250–10276. [CrossRef]

17. Ramalingam, P.; MurugaPrabu, S.; Samuel, S.J. Enhanced protection for multimedia content in cloud. In Proceedings of the International Conference on Communication and Signal Processing, Tamilnadu, India, 6–8 April 2017; pp. 1369–1372.
18. Ennaji, S.; Akkad, N.E.; Haddouch, K. A Powerful Ensemble Learning Approach for Improving Network Intrusion Detection System (NIDS). In Proceedings of the International Conference on Intelligent Computing in Data Sciences, Tartu, Estonia, 15–16 November 2021; pp. 1–6.
19. Olayinka, T.C.; Ugwu, C.C.; Okhuoya, O.J.; Olusọla Adetunmbi, A.; Solomon Popoola, O. Combating Network Intrusions using Machine Learning Techniques with Multilevel Feature Selection Method. In Proceedings of the IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development, Lagos Nigeria, Nigeria, 5–7 April 2022.
20. Desale, K.S.; Ade, R. Genetic algorithm based feature selection approach for effective intrusion detection system. In Proceedings of the International Conference on Computer Communication and Informatics, Coimbatore, India, 8–10 January 2015; pp. 1–6.
21. Verma, J.; Bhandari, A.; Singh, G. Feature Selection Algorithm Characterization for NIDS using Machine and Deep learning. In Proceedings of the IEEE International IoT, Electronics and Mechatronics Conference, Toronto, ON, Canada, 1–4 June 2022; pp. 1–7.
22. Su, Y.; Qi, K.; Di, C.; Ma, Y.; Li, S. Learning automata-based feature selection for network traffic intrusion detection. In Proceedings of the IEEE Third International Conference on Data Science in Cyberspace, Guangzhou, China, 18–21 June 2018; pp. 622–627.
23. Lawrence, H.; Ezeobi, U.; Bloom, G.; Zhuang, Y. Shining New Light on Useful Features for Network Intrusion Detection Algorithms. In Proceedings of the IEEE 19th Annual Consumer Communications & Networking Conference, Las Vegas, NV, USA, 8–11 January 2022.
24. Eldos, T.; Siddiqui, M.K.; Kanan, A. On the KDD'99 Dataset: Statistical analysis for feature selection. *J. Data Min. Knowl. Discov.* **2012**, *3*, 88.
25. Li, J.; Cheng, K.; Wang, S.; Morstatter, F. Feature selection: A data perspective. *Assoc. Comput. Mach.* **2017**, 50. [CrossRef]
26. Thejaswee, M.; Srilakshmi, P.; Karuna, G.; Anuradha, K. Hybrid IG and GA based Feature Selection Approach for Text Categorization. In Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, Coimbatore, India, 5–7 November 2020; pp. 1606–1613.
27. Ghoualmi, L.; Benkechkache, M.E.A. Feature Selection Based on Machine Learning Algorithms: A weighted Score Feature Importance Approach for Facial Authentication. In Proceedings of the International Informatics and Software Engineering Conference, Ankara, Turkey, 15–16 December 2022; pp. 1–5.
28. Lawrence, T.; Zhang, L. IoTNet: An efficient and accurate convolutional neural network for IoT devices. *Sensors* **2019**, *19*, 5541. [CrossRef] [PubMed]
29. Htun, P.T.; Khaing, K.T. Anomaly intrusion detection system using Random Forests and k-Nearest Neighbor. *Probe* **2013**, *41102*, 2377.
30. Kim, D.S.; Nguyen, H.N.; Park, J.S. Genetic algorithm to improve SVM based network intrusion detection system. In Proceedings of the International Conference on Advanced Information Networking and Applications, Taipei, Taiwan, 28–30 March 2005; Volume 2, pp. 155–158.
31. Sultana, N.; Chilamkurti, N.; Peng, W.; Alhadad, R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-Peer Netw. Appl.* **2019**, *12*, 493–501. [CrossRef]
32. Hashemi, M.J.; Keller, E. Enhancing robustness against adversarial examples in Network Intrusion Detection Systems. In Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, Leganes Madrid, Spain, 10–12 November 2020; pp. 37–43.
33. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Recurrent Neural Network for intrusion detection in SDN-based networks. In Proceedings of the IEEE Conference on Network Softwarization and Workshops, Montreal, QC, Canada, 25–29 June 2018; pp. 202–206.
34. Deng, Y.; Kamani, M.M.; Mahdavi, M. Adaptive Personalized Federated Learning. *arXiv* **2020**, arXiv:2003.13461.
35. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and robust federated learning through personalization. In Proceedings of the International Conference on Machine Learning, Virtual Event, 18–24 July 2021; pp. 6357–6368.
36. Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; Dou, Q. Fedbn: Federated Learning on Non-Iid Features via Local Batch Normalization. *arXiv* **2021**, arXiv:2102.07623.
37. Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; Alvarez, J.M. Personalized Federated Learning with First Order Model Optimization. *arXiv* **2020**, arXiv:2012.08565.
38. Fallah, A.; Mokhtari, A.; Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020.