Trustworthy Generative Few-Shot Learning based Intrusion Detection Method in Internet of Things

Ze Zhang, Pan Wang, *Member, IEEE*, Tianqi Zhang, Minyao Liu, Xiaokang Zhou *Member, IEEE*

***Abstract*—The application of Artificial Intelligence (AI) and Internet of Things (IoT) technologies enhances device smart decision-making, yet it also makes them more susceptible to cyber-attacks. AI-enabled Network Intrusion Detection System (NIDS) has been regarded as an effective way to mitigate such attacks. However, large-scale attacks data collection and labeling is prohibitively costly and time-consuming. Therefore, effective Few-Shot Learning based NIDS for IoT are crucial. Variational Autoencoders and Generative Adversarial Networks are widely used in generative few-shot learning but struggle with limited sample diversity and unstable training. Meanwhile, Deep Learning-based NIDS often lacks trustworthiness due to the "black box" problem. In response to these challenges, we propose a network traffic sample generation method that leverages the Conditional Denoising Diffusion Probabilistic Model (CDDPM) to tackle the issues of samples. Then, we introduce an Intrusion Detection method named CNNBiGRU that combines Convolutional Neural Networks with Bidirectional Gated Recurrent Units based on the synthetic data generated by CDDPM. Furthermore, we apply the SHAP method to interpret the results to ensure model users' trust. Finally, the proposed method is evaluated on the two public datasets, which illustrate that our approach can successfully generates high-quality samples and improves the performance of NIDS.**

***Index Terms*—Internet of Things, Consumer Electronics, Diffusion Models, Network Intrusion Detection, Generative Few-Shot Learning, Model interpretability.**

I. INTRODUCTION

The rapid development of Artificial Intelligence and Internet of Things (AIoT) technology has driven the formation of a series of innovative products, business models, and ecosystems centred on AI in the consumer electronics industry, especially in the fields of computers, mobile phones, homes, cars, and wearable devices. With the continuous progress and application of these technologies, the scale of the consumer electronics market is also continuously expanding. According to Future Market Insights analysis, the global consumer electronics market is expected to continue growing at a compound annual growth rate of 5.80% from 2023 to 2033. By 2033, the market valuation is expected to reach 5.82065 billion US dollars [1]. The widespread application of AIoT in the field of consumer electronics and the increase in the number of consumer electronic terminal devices owned by users have enriched the application scenarios of interconnection, data transmission, and sharing among multiple consumer electronic devices [2]. However, this technological advancement and high degree of interconnection between devices have also brought new challenges. The connected consumer electronic devices, for example, a connected refrigerator, a connected car or any other home automated IoT device, may create an entry point for hackers to use them for malicious acts, with serious consequences.

Consumer electronic devices are limited by their hardware performance, and their network structures are simple, making them vulnerable to various attack threats, such as botnets, brute force, denial of service, distributed denial of service, and web attacks, among others. Attackers can obtain users' private information through network attacks, leading to severe consequences. Therefore, a network traffic intrusion detection system for consumer electronic device networks is essential. Network intrusion detection systems are used to monitor network traffic or system activity in order to detect and respond to potential security threats or unauthorized access attempts. For example, in smart home systems, network intrusion detection algorithms can be used to identify unauthorized intrusions or abnormal operations of devices [3].

As consumer electronic devices increasingly connect to the internet and consumers demand more personalization and customization, the complexity of intrusion detection methods for consumer electronics networks also increases. The network traffic of consumer electronics, such as smartphones and smart home devices, is diverse, and the traffic patterns of each device may vary greatly. Therefore, determining which traffic is normal and which is attack traffic requires a deep understanding of the network behavior of each type of consumer electronic device. Otherwise, it is difficult to label them, resulting in a scarcity of labeled samples. Moreover, in practical applications, attack events are relatively rare compared to normal events, leading to an insufficient collection of attack samples. This makes it challenging for intrusion detection models based on supervised learning to obtain enough training data, thereby affecting the model's performance. Hence, a few-shot learning based network intrusion detection method for consumer electronic is necessary.

In network intrusion detection systems facing few-shot or class-imbalance issues, Generative Few-Shot Learning(GFSL) [4] is an effective solution. It aims to use a small amount of labeled sample data to train the model, and can generate new and diverse data samples, so as to enhance the generalization ability and performance of the model. Traditional Generative Few-Shot Learning Methods such as Synthetic Minority Over-Sampling Technique(SMOTE), Variational Au-

Z.Zhang is with the School of Internet of Things, Nanjing University of Post&Telecommunications, Nanjing, China, (e-mail:zhangze@njupt.edu.cn)

P.Wang, T.Zhang, M.Liu are with the School of Modern Posts, Nanjing University of Post&Telecommunications, Nanjing, China, (e-mail: wangpan, zhangtianqi, liuminyao@njupt.edu.cn).

Xiaokang Zhou is with the Faculty of Business Data Science, Kansai University, Osaka 565-0823, Japan (e-mail: zhou@kansai-u.ac.jp).

toencoder (VAE), and Generative Adversarial Network (GAN) have been widely applied. However, these technologies also have limited diversity of generated samples and inconsistent quality. Recently, diffusion models [5], as a new form of deep generative model, have shown outstanding performance in image generation and multimodal generation tasks, which generate data by gradually adding and removing noise. Given the robust capabilities of the diffusion model, we attempt to apply it to network traffic intrusion detection in consumer electronics products to obtain diverse and high-quality traffic samples.

The main goal of the paper is to propose a traffic sample generation method based on the Conditional Denoising Diffusion Probabilistic Model (CDDPM) and an intrusion detection classifier based on Convolutional Neural Networks [6] and Bidirectional Gated Recurrent Units(CNNBiGRU) to improve the performance and efficiency of network intrusion detection in consumer electronics. Additionally, using SHapley Additive exPlanations(SHAP) [7] to interpret the classification results of the CNNBiGRU model locally enhances the system's transparency and user trust. The main contributions of this study are as follows:

1) The paper proposes a network traffic sample generation technique based on the CDDPM. It can effectively reduce dependence on real data and alleviate the problem of intrusion detection performance degradation caused by the scarcity of labeled attack samples in the IoT.
2) A network intrusion detection classifier named CNNBiGRU is proposed which combines Convolutional Neural Networks and Bidirectional Gated Recurrent Units. This combination leverages the strengths of CNN in feature extraction and the efficiency of BiGRU in processing time-series data, enabling more precise identification and classification of various attack behaviors, thereby significantly improving the accuracy and efficiency of NIDS.
3) Extensive evaluations on the public datasets CICDDOS2019 and CICIDS2017 have validated the effectiveness of the proposed method. The test results not only prove the effectiveness of the generated samples in maintaining predictive performance but also demonstrate the effectiveness and reliability of the IoT NIDS.
4) The SHAP method is used to analyze the top 10 features that each contribute the most when making decisions using the CNNBiGRU method on the CICDDOS2019 and CICIDS2017 datasets. Explaining the reasons and discovering that they match real experience further enhances the interpretability of the model, thereby improving the user's perception of system transparency and trust.

II. Related Work

A. Network Intrusion Detection based on ML and DL

Intrusion detection for encrypted traffic has evolved through three main phases. It starts with dependency on port characteristics and statistical traffic analysis, which becomes less effective due to tactics like port masking. The focus then shifts towards Machine Learning (ML) techniques that utilize network flow features, employing algorithms such as naive Bayes, support vector machines, clustering [8] and decision trees [9]. However, these methods often rely on manually designed features and struggle with nonlinear traffic data, leading to a pivot towards Deep Learning (DL) models like Multi-Layer Perceptron(MLP) [10], Random Forests [11], Convolutional Neural Network(CNN) and GAN [12], which can automatically extract features and handle large, complex datasets more effectively. For example, Javeed et al. [13] proposed an interpretable and elastic industrial 5.0 IDS, combining Bidirectional Long Short Term Memory Network (BiLSTM) and Bidirectional Recurrent Unit (BiGRU) to enhance the attack detection process in Industry 5.0. Wang et al. [14] proposed a semi-supervised model to improve the accuracy and effectiveness of the system. Hwang et al. [15] developed an anomaly traffic detection mechanism named D-PACK, consisting of a CNN and an unsupervised deep learning model for automatic analysis of traffic patterns and filtering of anomalous traffic.

However, whether ML or DL, most researchers assume a balanced and abundant sample set when studying IDS. In reality, normal samples are plentiful, while anomalous samples are often scarce and difficult to collect. This imbalance poses a challenge to the model's performance and generalization capabilities.

B. Generative Few-Shot Learning

In the network intrusion detection system, it is a common and difficult problem that the sample is sparse or the data is unbalanced. In this case, Generative Few-Shot Learning becomes an excellent solution [16]. By combining the advantages of generative models and few-shot learning, generative few-shot learning uses a small amount of training data to train the model. It can also generate new data samples that fit the data distribution and improve the performance of the model on unseen tasks or categories. There are various ways to implement it, such as the Synthetic Minority Over-sampling Technique, the Variational Autoencoders [17], the Generative Adversarial Networks [18]–[20] and the Meta-Learning [21]–[23], among others. For example, Roy et al. [24] proposed a lightweight intrusion detection model that, by eliminating multicollinearity and applying SMOTE oversampling, identifies the most critical features. Chen et al. [25] used a conditional wasserstein GAN to generate effective data in a smart home environment, filling the dataset and achieving data augmentation for few-shot data.

Although these techniques have been widely studied and applied, they still face some challenges, such as easy introduction of noise, insufficient sample quality and diversity, and unstable model training [26]. In contrast, diffusion model, as an emerging generative model, with its unique training mechanism and theoretical foundation, can not only generate high-quality and realistic samples, but also show higher stability during training [27]. Consequently, we are exploring the use of diffusion models for sample generation in IDS research.

C. Explainable Network Intrusion Detection

Explainable network intrusion detection techniques are classified into pre-model, in-model, and post-model categories. Pre-model techniques [28], such as feature selection, prepare data for better interpretability. Pre-model techniques are implemented before the network intrusion detection process. For example, Wu et al. [29] noted that using high-level, human-understandable feature representations for network intrusion detection can reduce the complexity of subsequent network intrusion detection models to improve interpretability. In-model techniques employ inherently interpretable models, eliminating the need for additional efforts to provide explanations. Bohmer et al. [30] developed the ADAR model to detect and explain anomalies in process runtime behaviors. Kraiem et al. [31] introduced the composite decision tree to detect and explain anomalies in time series. Post-model techniques analyze the decisions of network intrusion detection models after processing to correlate inputs with outputs, using tools like SHAP for deeper insights into the contributions of various features and enhancing the transparency and trustworthiness of the detection systems. Schlegl et al. [32] built a model based on DNNs that learns interpretable feature representations from unlabeled time series, facilitating the evaluation and deployment of subsequent network intrusion detection algorithms. Barbado et al. [33] applied several rule extraction techniques to the OCSVM model for anomaly explanation and evaluated the quality of the explanations produced. This paper adopts Post-model techniques, applying the SHAP explainable method to the interpretability of the intrusion detection model.

III. Trustworthy Generative Few-Shot Learning based Intrusion Detection Method

A. Overall Framework

The framework of the NIDS system proposed in this article is shown in Fig. 1. After gathering the traffic input data, we transform it into a Network Traffic Feature Matrix (NTFM). Then, we perform data preprocessing operations, such as removing useless features and data, normalizing data, and digitizing labels. The training NTFM is fed into a conditional denoising diffusion probabilistic model to train the UNET model for noise prediction. Following this, high-quality samples are generated using the trained model, combined with a small number of real samples, and input into the preset CNNBiGRU model for training. Finally, the testing NTFM is input into the trained CNNBiGRU model to obtain prediction results, and the confusion matrix and SHAP method are employed to evaluate and explain the results.

B. Network Traffic Feature Matrix

The traffic captured by consumer electronic device networks is raw packet bytes and cannot be directly put into model training. In this study, CICFlowMeter [34] is used to convert raw packet bytes into traffic feature vectors. Flows are categorized using a quintuple consisting of source address, destination address, source port, destination port, and TCP/UDP protocol. The flow features include three categories: packet-level features, flow-level features, and statistical features. Table I lists a few example features. Flow feature vector is defined as $f = f^1, f^2, \cdots, f^N$, where N is the number of features contained in the flow feature vector.

As shown in the Fig. 2, the Network Traffic Feature Matrix is defined [35] by combining the traffic feature vectors $F = [f_1, f_2, \cdots, f_M]$ and the behavioral labels $L = [l_1, l_2, \cdots, l_M]$, where M is the number of flow feature vectors. The resulting feature matrix, NTFM $= [F, L]$, can be preprocessed for model training.

C. Conditional Denoising Diffusion Probabilistic Model

Denoising Diffusion Probabilistic Model(DDPM) is a generative model that generates data by simulating diffusion processes. This model gradually adds noise to the data, turns it into pure noise, and then learns the reverse process to remove it, ultimately generating new samples similar to the original data.

In the forward diffusion process, given an initial data x_0, a small amount of gaussian noise is gradually added to generate a series of noisy data $x_1, x_2, \cdots, x_T$. This process can be expressed as Eq. 1 and Eq. 2:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

$$x_t = \sqrt{1-\beta_{t-1}}x_{t-1} + \sqrt{\beta_t}\epsilon_t \tag{2}$$

Where β_t is a known noise-increasing parameter that represents the parameter used to control the degree of randomness. ϵ_t is the gaussian noise added at step t, sampled from $N(0, I)$.

This diffusion process is a Markov chain process. We can calculate the relationship between sample x_t and the original sample x_0 at any intermediate step t without iteration. As shown in Eq. 3

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{3}$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\alpha_t = 1 - \beta_t$,

The inverse diffusion process is the inverse process of a forward process aimed at reconstructing the original data x_T from the noisy data x_0. We cannot directly obtain $q(x_{t-1}|x_t)$, which requires us to understand the complete distribution of real data. But if x_0 is introduced, we can get the distribution of Eq. 4.

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t\mathbf{I}) \tag{4}$$

Derive and simplify to obtain the mean and variance:

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \tag{5}$$

$$\tilde{\mu}(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right). \tag{6}$$

Using the variance and mean described above, through the reparameterization trick, we can sample the current x, which is x_{t-1}, as shown in Eq. 7.

$$x_{t-1} = \tilde{\mu}(x_t, t) + \sqrt{\tilde{\beta}_t}\epsilon \tag{7}$$

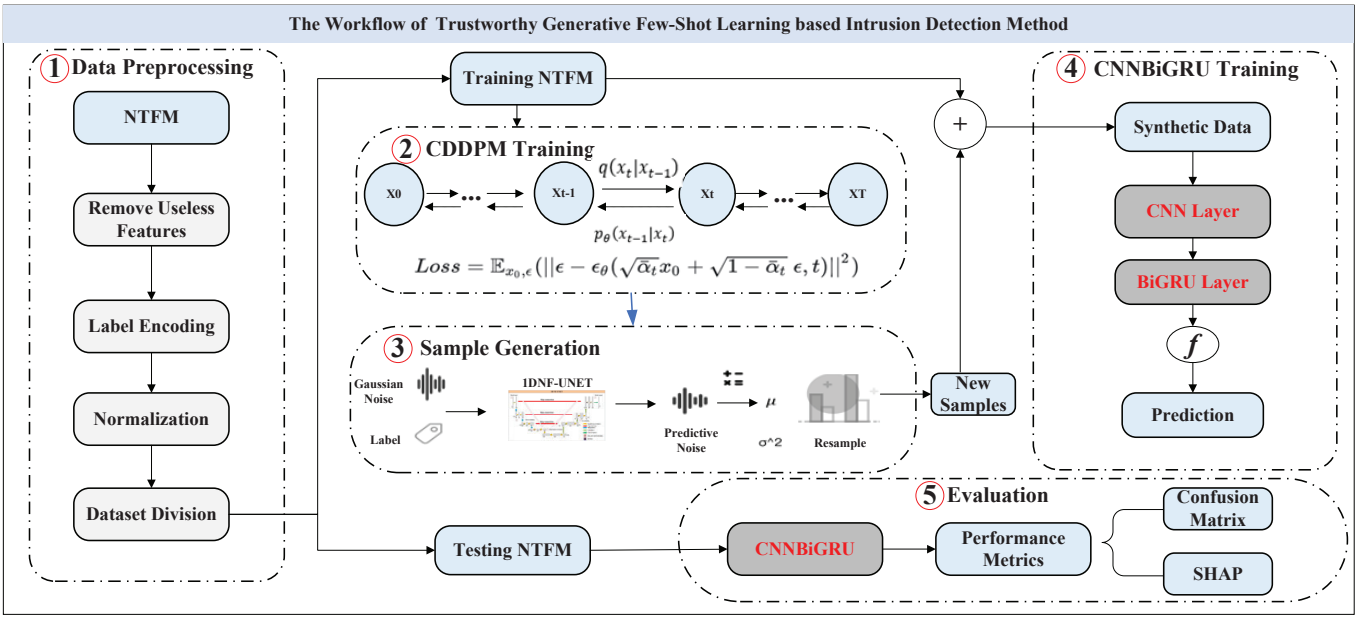


Fig. 1: The Workflow of Trustworthy Generative Few-Shot Learning based Intrusion Detection Method

TABLE I: A Typical Example (Partial) of Network Traffic Features

Attributes	Catogory	Description
ACK Flag Count	Packet-Level	The total number of ACK flags in all packets in a TCP session.
Min Packet Length	Packet-Level	The minimum length of all packets observed in a network flow.
Flow Bytes/s	Flow-Level	The average number of bytes transmitted per second during the duration of the entire stream.
Flow Duration	Flow-Level	The total time from the beginning to the end of the flow.
Active Mean	Statistical	The average time interval between activities in the flow.
Idle Mean	Statistical	The average idle time interval in the flow.

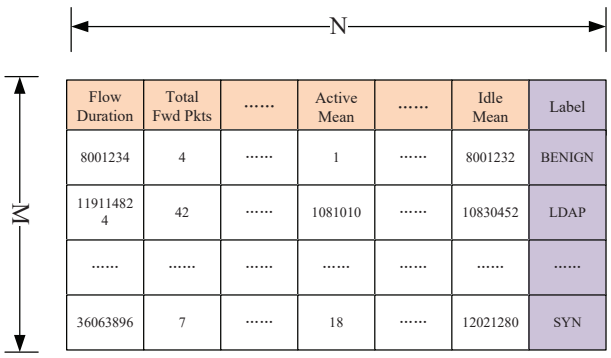


Fig. 2: The Example of NTFM

We use a parameterized model $p_\theta(x_{t-1}|x_t)$, such as UNET network, to approximate the distribution of Eq. 4, represented as Eq. 8.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{8}$$

Because the variance is usually set as a constant $\tilde{\beta}_t$, we just need to predict the parameter mean $\tilde{\mu}_\theta(x_t, t)$.

As known from Eq. 6, to estimate the mean, we only need to use the model to estimate the noise. The model's training loss is calculated by minimizing the negative log-likelihood $-\log(p_\theta(x_0))$. The negative log-likelihood is optimized using the Variational Lower Bound (VLB), and by applying the definition of KL divergence and its non-negativity property, we can ultimately simplify and obtain the loss as Eq. 9.

$$Loss = \mathbb{E}_{x_0,\epsilon}(\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2) \tag{9}$$

Removing the coefficients before loss can make the training more stable [5], and the final loss is shown in Eq. 10.

$$\text{Loss} = \mathbb{E}_{x_0,\epsilon}\left(\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right) \tag{10}$$

Where ϵ is the actual added noise, and $\epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)$ is the noise predicted by the model through inputs x_t and t, both of which calculate $L2$ loss.

However, the samples generated by the diffusion model trained in this way are unconditional, and each type of sample needs to be retrained. Therefore, we embed conditional information into model training, and there are currently two mainstream methods for introducing conditions: classifier guidance and classifier free guidance. Classifier guidance additionally trains classifiers in the model, while classifier guidance learns both conditional and unconditional generation tasks during the training phase. We use a classifier-free guidance approach to implement a conditional introduction.

$$\bar{\epsilon}_\theta(x_t, t, y) = (1-w)\epsilon_\theta(x_t, t) + w\epsilon_\theta(x_t, t, y) \tag{11}$$

Algorithm 1 Algorithm for CDDPM Training and Sample Generation

Input:

Training data: Real traffic feature matrix $NTFM_{real} = [F_{real}, L_{real}]$, where $F_{real} = \{f_{re0}, f_{re1}, ...f_{reM}\}$ and $L_{real} = \{l_{re0}, l_{re1}, ...l_{reM}\}$;

Training parameters: Total number of noise steps T, Training epochs num_epochs, Generate sample size N , Generate labels for data $L_{generate} = \{l_{ge0}, l_{ge1}, ...l_{geN}\}$;

Output: CDDPM model and Generated samples $NTFM_{generate}$.

Initial dataset: Remove useless labels and invalid data, normalize numerical values, and digitize labels;
Define the exact architectures of CDDPM;
Calculate the forward diffusion parameter array $\beta = \{\beta_0, ...\beta_T\}$,$\alpha = \{\alpha_0, ...\alpha_T\}$ and $\bar{\alpha} = \{\bar{\alpha}_0, ...\bar{\alpha}_T\}$ based on T
for epoch in num_epochs **do**
  for f,l in F_{real},L_{real} **do**
    Randomly selecting t from $\{1, 2, ..., T\}$;
    Sampling noise ϵ_t in $N(0, I)$;
    Inputing f, t, and ϵ_t into the formula to directly generate f_t: $f_t = \sqrt{\bar{\alpha}_t} f + \sqrt{1 - \bar{\alpha}_t}\epsilon_t$;
    Inputing f_t, t and l into 1D NF-UNET for training. Loss $= \mathbb{E}_{f,\epsilon}\left(\|\epsilon - \epsilon_\theta(f_t, t, l)\|^2\right)$;
    Update the weights and bias;
  end for
end for
for n in $\{1, 2, ..., N\}$ **do**
  Setting l_n as $L_{generate}(n)$;
  Sampling noise f_t in $N(0, I)$;
  for t in $\{T, T-1.., 1\}$ **do**
    Sampling noise ϵ_t in $N(0, I)$ if $t > 1$, else $\epsilon_t = 0$;
    Inputing noise ϵ_t, l_n and t into the trained model 1D NF-UNET to obtain the predicted noise ϵ_θ;
    Inputing ϵ_θ into the formula to obtain f_{t-1}: $f_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(f_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\right) + \sqrt{\tilde{\beta}_t}\epsilon$
  end for
  Record the generated f_0 in F_g
end for The generated data F_g and labels $L_{generate}$ are composed to obtain the generated $NTFM_{generate}$

Eq. 11 introduces this process, where w is the label-guided weight. During model training, we directly introduce category information, add the time step embedding and category embedding, and set the label guided weight w to 1 when generating samples.

In order to make the noise prediction network in the CDDPM model applicable to one-dimensional NTFM, we modified the UNET network and named it 1DNF-UNET. The specific structure and parameters are shown in Fig. 3, with a height of N as the feature dimension of one sample and 1 as the number of channels. The network consists of five modules:

1) **Input module:** Embeds the category information into the step information, then converts it into the required input size through a doubleconv1d operation.

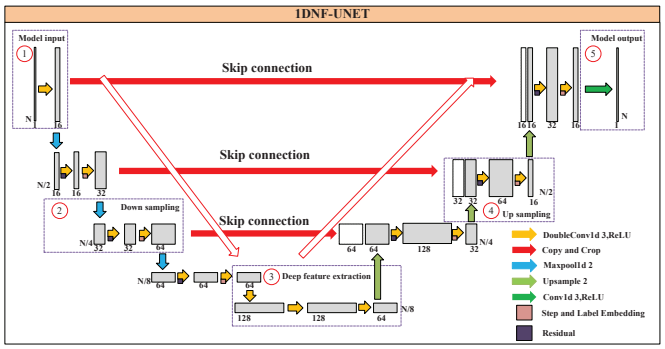


Fig. 3: The Structure of 1DNF-UNET

2) **Downsampling module:** Performs a maxpool1d downsampling operation, then performs two doubleconv1d operations, the first using a residual structure, and the second embedding the step and label information.
3) **Deep convolution module:** Performs three doubleconv1d operations on the downsampled data.
4) **Upsampling module:** Performs an upsampling operation, fills in any missing dimensions if necessary, then concatenates the upsampled data with the corresponding downsampled module data, and performs two doubleconv1d operations like the downsampling to make the dimensions consistent with the corresponding downsampled module.
5) **Output module:** Employs a single convolution operation on the upsampled data to restore it to its initial input shape.

Alg.1 shows the whole training process of CDDPM. In this way, we can use diffusion model to generate samples of specified categories, effectively reducing dependence on real samples.

D. Intrusion Detection Model

The CNNBiGRU proposed in this study is a deep learning model that combines CNN and BiGRU. It leverages the capability of CNN in feature extraction and the efficiency of BiGRU in capturing temporal dependencies to detect intrusions in network traffic for consumer electronic devices.

In the CNN portion, the main task is to process the network traffic sequence to extract key local features. This process is achieved by sliding the convolution kernel along the sequence and calculating the dot product at each position. Given a one-dimensional input sequence X and a convolution kernel k, containing M weights, the convolution output c at position i can be expressed as:

$$c(i) = \sum_{m=0}^{M-1} k(m) \cdot x(i \times S + m - P) \tag{12}$$

where $x(i + m)$ represents the value at the index $(i + m)$ in the input sequence x, and $k(m)$ represents the m-th weight of the convolution kernel k. By adjusting the stride S and boundary padding P, one can effectively control the length of

the output sequence and ensure the convolution kernel covers all key areas of the sequence.

The BiGRU structure includes two GRU layers, one processing the data in the correct temporal order and the other in reverse. The bidirectional architecture allows the model to capture both prior and future contextual information simultaneously. By merging the outputs of both directions, a comprehensive representation of the entire sequence's information is included.

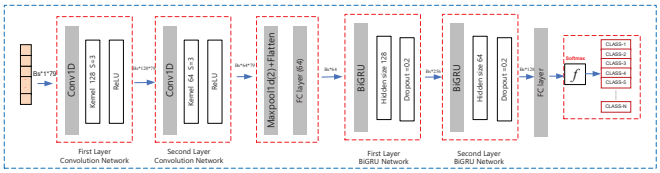


Fig. 4: The Structure of CNNBiGRU

Each GRU has two gates: the reset gate and the update gate. The reset gate decides how to combine new input information with past memories, while the update gate defines how much of the past memory should be retained, making it more effective in capturing long-term dependencies in time series. Each GRU unit's state at time step t is updated as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{13}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{14}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b_h) \tag{15}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{16}$$

where σ is the sigmoid activation function, W, W_z, W_r are learnable weight matrices, $\tanh$ is the hyperbolic tangent activation function, z_t is the update gate, r_t is the reset gate, $\tilde{h}_t$ is the candidate hidden state, and h_t is the current time step's hidden state.

As shown in Fig. 4, The model is configured with two convolutional layers, with their output channel numbers set to 128 and 64, respectively, both using convolution kernels of size 3 and corresponding boundary padding to maintain consistency in feature dimensions. The subsequent two GRU layers are set to bidirectional, with the first BiGRU layer enhancing the 64-dimensional features to 128 dimensions and the second BiGRU layer continuing to process data and outputting 64-dimensional features. A dropout ratio of 0.2 is introduced in the BiGRU layer to suppress overfitting and enhance the model's generalization performance. After passing through the fully connected layer, the model outputs the final classification results, which integrate the spatial features learned by the CNN layer with the temporal features revealed by the BiGRU layer, providing precise recognition capabilities for network intrusion detection tasks.

E. Model Interpretable Methods

SHAP is an advanced model interpretability method based on shapley values in cooperative game theory. Its purpose is to quantify and understand the contribution of each feature to the model prediction results by calculating the changes that each feature incorporates or excludes from the model prediction. For each feature i, its shapley value is calculated as follows:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (f(S \cup \{i\}) - f(S)) \tag{17}$$

Where N is the set of all features, S is any subset of features excluding feature i, $|S|$ is the number of features in set S, $f(S)$ is the model's prediction result with only the features in set S, and $f(S \cup i)$ is the model's prediction result with the features in set S plus feature i. By quantifying the average contribution of each feature across all possible combinations of features, SHAP provides a detailed explanation for model decisions. In this study, we calculate the SHAP values for each feature using selected samples and then take the average absolute value of the SHAP values for each feature across all selected samples to depict the average impact of features on model predictions. These average absolute values are then sorted by size to form a ranking of feature importance that reflects the effect on the model's outputs, thereby revealing the contribution of each feature in the model's decision-making process. This not only provides a global understanding of the model's behavior but also specifically elucidates the explainability of the decision-making process.

IV. EVALUTION

A. Dataset

This study uses publicly available datasets CICDDOS2019 [36] and CICIDS2017 [37], both generated and published by the Canadian Institute for Cybersecurity, containing various types of attack traffic for research and testing of Network Intrusion Detection Systems. The specific situation is shown in Table II, for the larger number of classes we take the first 30000 records.

TABLE II: Dataset Description

CICIDS2017		**CICDDOS2019**	
Category	Quantity	Category	Quantity
Dos slowloris	5796	DoS_UDP	30000
Heartbleed	11	DoS_DNS	30000
DDoS	30000	TFTP	30000
SSH-Patator	5897	WebDDoS	439
Infiltration	36	BENIGN	30000
FTP-Patator	7935	DoS_NTP	30000
Brute Force	1507	DoS_SSDP	30000
DoS Slowhttptest	54099	DoS_LDAP	30000
PortScan	30000	Syn	30000
XSS	652	UDPLag	30000
DoS GoldenEye	10293	DoS_MSSOL	30000
Sql Injection	21	Portmap	30000
BENIGN	30000	DoS_SNMP	30000
Bot	1956	DoS_NetBIOS	30000
DoS Hulk	30000		

The dataset contains two forms: PCAP files and CSV files. This paper uses CSV format data because both are derived using CICFlowMeter to extract network traffic characteristics. Thus, the two datasets have similar features.

B. Experimental Setting

The experimental setup for this study is shown in Table III. We use a 12th Gen Intel Core i9-12900H CPU with 16GB of RAM and an NVIDIA GeForce RTX 3060 graphics card. The python version is 3.8.5, and the primary tool for model construction is torch, version 2.0.1+cu117, optimized for CUDA 11.7 to leverage GPU acceleration for enhanced computational efficiency.

TABLE III: Experimental Setting

System Aspect	Specification
CPU	12th Gen Intel Core i9-12900H
Memory Capacity	16GB RAM
Graphics Card	NVIDIA GeForce RTX 3060
Python Version	3.8.5
Torch Version	2.0.1+cu117

The evaluation metrics for this experiment include $Precision$, $Recall$, $F1$-score, and $Accuracy$. The performance evaluation indicators of all models, except for accuracy, are calculated based on weighted average, which can balance the imbalance of sample sizes in different categories and more comprehensively reflect the overall performance of the model.

C. Experimental Design

We have conducted three experiments to verify the effectiveness of the NIDS proposed in this paper. This section introduces the specific experimental setup.

1) **The Evaluation of Network Intrusion Detection Performance:** To comprehensively evaluate the performance of the proposed CNNBiGRU model with finite samples, A series of experiments were carried out to compare it with traditional machine learning methods such as random forest and K-Nearest Neighbor (KNN) and the latest deep learning algorithms such as BiLSTMBiGRU [13],BiLSTMBiLSTM [38] and CNNBiLSTM [39] in the finite sample scenario. We randomly selected 5, 50, 500, and 5000 samples from each category of the two datasets as training data. When the number of samples was insufficient, we uniformly selected about half of the total samples of this class for training. Each algorithm was trained on these training samples for 100 rounds with a batch size of 64 and a uniform learning rate of 0.0001. During the testing phase, we randomly selected 1000 samples from each class on the CICIDS2017 dataset for testing and 10000 samples from each class on the CICDDOS2019 dataset for testing. When the number of samples was insufficient, we selected all samples not used in training for testing.
2) **The Evaluation of CDDPM Sample Generation:** To find the optimal model parameters, this study conducted multiple rounds of experiments around different diffusion steps (500, 1000, 1500, and 2000 steps). In each experiment round, the noise prediction 1DNF-UNET network was uniformly trained for 2000 rounds, selecting 5 real samples from each category and using the conditional diffusion model to generate 50 enhanced samples per category, which were merged with the original samples to form a final training set of 55 samples per category. Each configuration was regenerated three times and tested twice, each time training the CNNBiGRU and testing it against 1000 real test samples per category, taking the remaining samples if not enough. To comprehensively evaluate and compare the performance of the proposed models, we also compared it with the popular Synthetic Minority Oversampling technique (SMOTE) [24] and the latest Wasserstein Conditional GANs with Gradient Penalty (WCGAN-GP) [40] method. By comparing the data generation quality and prediction ability of the three methods on real samples, the advantages and practical application value of the proposed model were evaluated.
3) **The Interpretability Analysis of the CNNBiGRU Model:** In this experiment, we used the SHAP method to conduct an interpretability assessment of the CNNBiGRU model on the CICDDOS2019 and CICIDS2017 datasets, analyzing the traffic features that contributed the most to its decisions. The specific experimental design was as follows: First, 500 examples from each category of each dataset were taken to train the model. Then, for the analysis of the local interpretability of the model's decisions, we selected one untrained sample from each category of each dataset as an explanation sample and input it into SHAP for calculation to parse which features played a decisive role in making each specific judgment.

TABLE IV: Evaluation Reports for Different Methods

Parameters	Dataset	Quantity	CNNBiLSTM	BiLSTMBiGRU	BiLSTM	Random Forest	KNN	Proposed Method
Accuracy	2019	5000	0.9643	0.9692	0.9720	0.6609	0.6203	0.9940
		500	0.9593	0.9201	0.9165	0.6412	0.5820	0.9801
		50	0.8191	0.7372	0.7776	0.5967	0.5290	0.9011
		5	0.1549	0.2487	0.2341	0.4706	0.3188	0.4096
	2017	5000	0.9910	0.9925	0.9989	0.9640	0.9500	0.9988
		500	0.9974	0.9752	0.9916	0.9572	0.9158	0.9954
		50	0.9310	0.9638	0.9692	0.9229	0.8080	0.9765
		5	0.3966	0.2837	0.5324	0.7649	0.2751	0.5189
Precision	2019	5000	0.9740	0.9765	0.9784	0.6827	0.6500	0.9942
		500	0.9661	0.9214	0.9364	0.6542	0.6336	0.9828
		50	0.8122	0.7544	0.8217	0.6142	0.5565	0.9316
		5	0.1657	0.1814	0.1880	0.5000	0.3692	0.4953
	2017	5000	0.9903	0.9926	0.9989	0.9602	0.9418	0.9988
		500	0.9975	0.9814	0.9920	0.9612	0.9168	0.9957
		50	0.9683	0.9721	0.9793	0.9381	0.8269	0.9797
		5	0.3822	0.2710	0.6233	0.8163	0.3288	0.6789
Recall	2019	5000	0.9643	0.9692	0.9720	0.6609	0.6203	0.9940
		500	0.9593	0.9201	0.9165	0.6412	0.5820	0.9801
		50	0.8191	0.7372	0.7776	0.5967	0.5290	0.9011
		5	0.1549	0.2487	0.2341	0.4706	0.3188	0.4096
	2017	5000	0.9910	0.9925	0.9989	0.9640	0.9500	0.9988
		500	0.9974	0.9752	0.9916	0.9572	0.9158	0.9954
		50	0.9310	0.9638	0.9692	0.9229	0.8080	0.9765
		5	0.3966	0.2837	0.5324	0.7649	0.2751	0.5189
F1-score	2019	5000	0.9626	0.9682	0.9712	0.6357	0.6015	0.9940
		500	0.9582	0.8979	0.8913	0.6226	0.5406	0.9799
		50	0.8020	0.7063	0.7751	0.5848	0.5191	0.8770
		5	0.0898	0.1700	0.1557	0.4531	0.2426	0.3248
	2017	5000	0.9904	0.9921	0.9989	0.9609	0.9448	0.9986
		500	0.9974	0.9684	0.9916	0.9586	0.9148	0.9954
		50	0.9229	0.9642	0.9702	0.9270	0.8044	0.9743
		5	0.3372	0.2222	0.5073	0.7839	0.2175	0.4643

D. Results Analysis

*1) **Network Intrusion Detection Performance Evaluation**:* The results, as shown in Table IV and Fig. 5, indicating that as the number of training samples decreases, the detection performance of all six methods decreases, highlighting the importance of sample size on model performance. For all compared methods, whether traditional machine learning algorithms or deep learning models, there is a dependency on a large number of samples. Fewer samples mean the model struggles to capture sufficient features to effectively distinguish between different categories, leading to a performance drop and potential loss of detection capability.

As the number of samples increases, the prediction accuracy also increases. It can be clearly seen that the method proposed

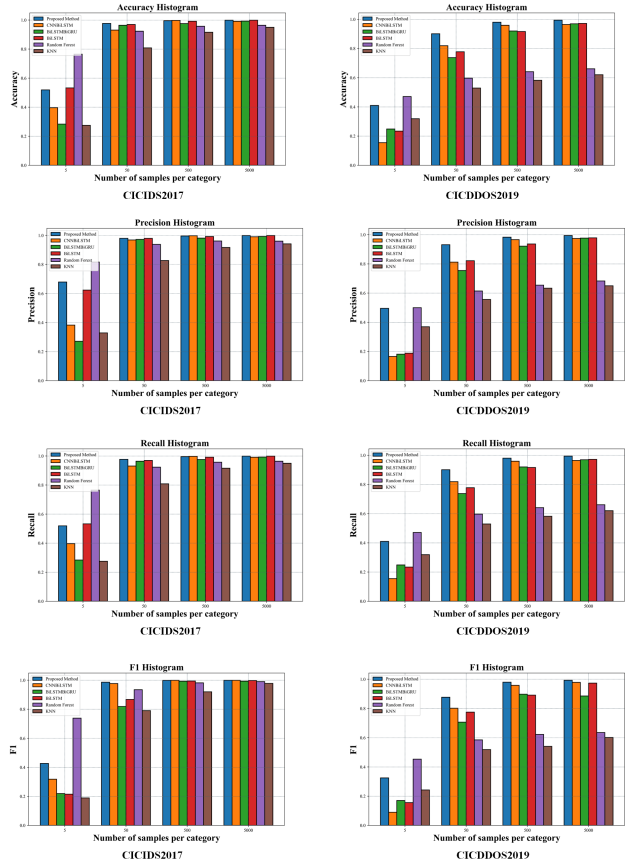


Fig. 5: Evaluation Bar Charts for Different Methods

in this article is superior to other comparison methods in a relatively small number of samples. On the CICDDOS2019 dataset, various indicators of the 5000, 500, and 50 orders are basically better than other methods. For example, for each class of 500 samples, the $F1$-score can reach 0.9800, and the other optimal methods only reach 0.9593. On the CICIDS2017 dataset, although the proposed method did not achieve optimal results at various scales, it remained stable in the top two levels, indicating that the proposed method can more effectively learn the characteristics of various attacks and normal traffic, and can achieve better results with relatively less learning.

In order to see the predictive performance of each method more specifically, we selected 5000 samples per case and plotted the confusion matrices of the first four methods on two datasets to demonstrate in detail the performance of these models in various attack detection. As shown in Fig. 6:

By comparison, it can be found that the detection performance of the method proposed in this article is only slightly inferior to BiLSTM on the CICIDS2017 dataset, and achieves the best performance on the CICDDOS2019 dataset. On CICIDS2017, the proposed method predicted poorly on Infiltration, with 9 out of 17 samples predicted as BENGIN, but other categories predicted well and were basically error free. On CICDDOS2019, the prediction of Portmap was slightly worse, with 629 prediction errors, but it was also better than the BiLSTM and BiLSTMBiGRU methods.

TABLE V: The Quality Comparison of Generated Samples

	500step	1000step	1500step	2000step	SMOTE	WCGAN-GP	5True	55True
CICIDS2017								
First	0.8441	0.7886	0.8402	0.7220	0.7016	0.7620	0.5035	0.9721
	0.7709	0.7723	0.8696	0.7062	0.7034	0.7141	0.5035	0.9721
Second	0.7598	0.9042	0.8676	0.6619	0.7378	0.6962	0.5035	0.9721
	0.7764	0.9095	0.7866	0.7450	0.7247	0.7002	0.5035	0.9721
Third	0.8381	0.9150	0.7931	0.7394	0.7168	0.7409	0.5035	0.9721
	0.8407	0.8881	0.7529	0.7540	0.7036	0.7313	0.5035	0.9721
CICDDOS2019								
	500step	1000step	1500step	2000step	SMOTE	WCGAN-GP	5True	55True
First	0.6356	0.7084	0.6752	0.7281	0.5489	0.6061	0.2944	0.8905
	0.7207	0.6946	0.6618	0.7517	0.5211	0.5552	0.2944	0.8905
Second	0.6870	0.7040	0.7115	0.7452	0.5746	0.5171	0.2944	0.8905
	0.7402	0.6734	0.6953	0.7855	0.6080	0.5108	0.2944	0.8905
Third	0.7424	0.6834	0.6713	0.8152	0.7021	0.5251	0.2944	0.8905
	0.7306	0.6713	0.6859	0.7682	0.6642	0.5428	0.2944	0.8905

TABLE VI: Resource consumption comparison

	Period	Time(s)	CPU(%)	GPU(%)
CDDPM	Training	16.57	1.92	46.75
	Generation	33.00	3.22	90.31
WCGAN-GP	Training	79.51	3.14	30.76
	Generation	2.08	0.90	9.00
SMOTE	Training and Generation	2.07	2.20	0.00

Although CNNBiLSTM had better prediction performance in this category than the proposed method, its prediction of DoS_SNMP was far inferior to the proposed method. In summary, the CNNBiGRU method proposed in this article can make relatively accurate predictions even with relatively few samples, making it an excellent intrusion detection method.

*2) **Conditional Diffusion Model Data Generation Evaluation**:* Table V and Fig. 7 comprehensively display the results of the experiment, where "5 True" represents training with 5 real samples per category, and "55 True" represents training with 55 real samples per category.

We can see that on the CICIDS2017 dataset, when the CDDPM step is 1000 steps, the model detection performance reaches its optimal level with an accuracy of 0.9150. On the CICDDOS2019 dataset, the optimal result was achieved with an accuracy of 0.8152 when the CDDPM step was 2000 steps. This indicates that the training steps of the CDDPM have a significant impact on the effectiveness of generating samples. When the number of steps is small, the model may not be able to fully predict the noise at each step, resulting in insufficient generated samples to support efficient learning. When the number of steps is high, the model may reduce sample diversity due to excessive learning, which may also affect the model's generalization ability. However, in any case, the prediction performance of the samples generated by CDDPM is better than other methods and the actual case of few-shot direct training.

Table VI shows the performance consumption of the three methods in model training and sample generation, and records the performance consumption of generating each class from 5 samples to 50 samples per class on the CICDDOS2019 dataset, where CDDPM and WCGAN-GP are trained for 1000 rounds, and the number of CDDPM diffusion steps is set to 1000. Smote is a linear interpolation method for sample generation, and its performance consumption is the lowest, which can be predicted. The performance of CDDPM model proposed in this paper is mainly consumed in the sample generation stage, and WCGAN-GP is mainly consumed in the model training stage, but the overall time is still dominated by CDDPM.

Based on the comprehensive analysis of the results, we can

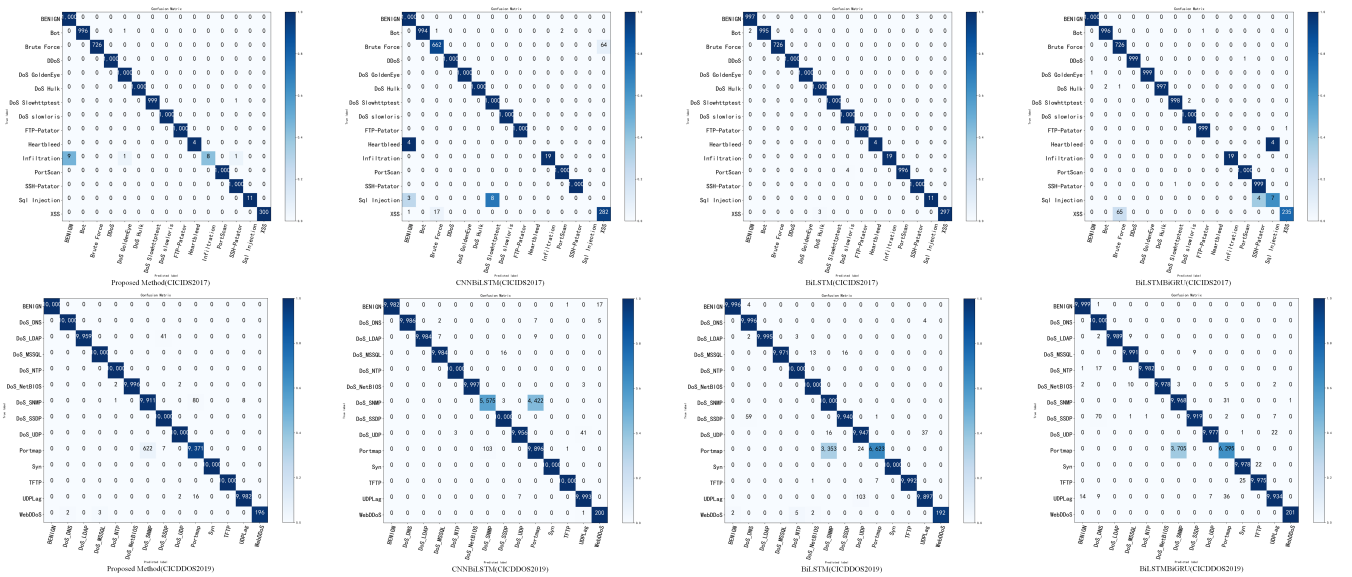


Fig. 6: Confusion Matrixs for Different Methods

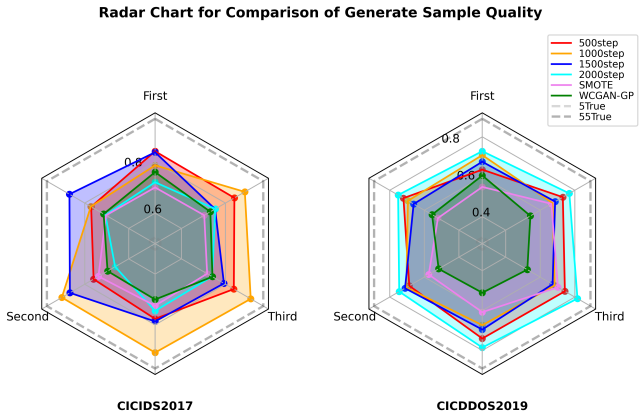


Fig. 7: Radar Charts for Comparison of Generated Samples Quality

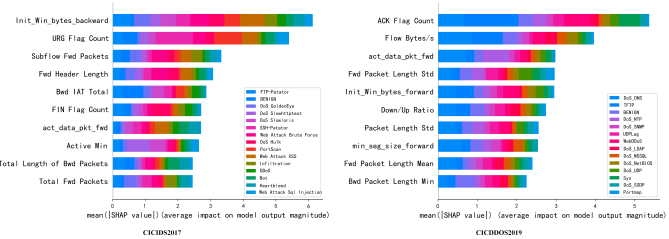


Fig. 8: Explanatory Diagram for CNNBiGRU

conclude that the CDDPM proposed in this study not only generates high-quality data under limited sample conditions, but also demonstrates the potential to enhance the detection performance of the model in experiments. This also provides evidence for the practicality and effectiveness of our proposed consumer electronic device network intrusion detection system, proving that the system can efficiently detect network attack traffic under limited sample resources, thereby reducing the need for a large amount of training data while ensuring security.

*3) **Interpretability Analysis of the CNNBiGRU Model**:* Illustrated by SHAP, Fig. 8 displayed the ten features contributing the most to the decision-making. During the decision analysis, we observed significant differences in the emphasis on features by the CNNBiGRU model across the CICDDOS2019 and CICIDS2017 datasets. This finding was as expected, as the CICDDOS2019 dataset focuses on recording DOS and DDOS attacks. In contrast, the CICIDS2017 dataset contains a wider variety of attacks, such as Web attacks and SSH attacks. The types of attacks differ, as do the bases on which the model makes its decisions. Specifically, for the CICDDOS2019 dataset, the ACK Flag Count and Flow Bytes/s were the most significant contributors to the model's decision-making. ACK Flag Count reflects the number of data packets acknowledged as received, and in DDoS attacks, attackers might abuse this mechanism to generate a large number of packets with ACK flags. Flow Bytes/s represents the density and rate of network traffic, which is critical in DDoS attacks, as the goal is often to consume network resources through a high traffic rate. In contrast, in the CICIDS2017 dataset, Init_Win_bytes_backward and URG Flag Count had a more significant impact. Init_Win_bytes_backward represents the initial window size when establishing a TCP connection. DDoS and DoS attacks might adjust the window size to control the volume of data sent to the target. PortScan attackers might alter the TCP window size to probe the system response, thereby identifying open ports, etc., all of which would correspondingly affect the TCP connection's window size, making this feature more effective in decision-making. URG Flag Count indicates that some data in the packet is urgent and needs to be processed as a priority. In some types of attacks,

such as buffer overflow attacks or certain scanning techniques, the URG flag might be used abnormally.

These findings not only validate the consistency between the model's decision logic and known attack behaviors but also deepen our understanding of how the model parses and responds to various cybersecurity threats. They provide support for the model's fairness, reliability, and transparency, better meeting user needs.

V. Conclusion

In this study, we focus on the challenges faced by smart consumer electronics in the field of security defense, proposing a new few-shot network intrusion detection system. This system includes a sample generator with CDDPM and an intrusion classifier CNNBiGRU that integrates CNN and BiGRU for network traffic. This system reduces the reliance on real anomaly data by generating high-quality samples, thereby improving the accuracy and efficiency of network intrusion detection. CDDPM is used to enrich the training dataset, enabling the model to effectively learn in environments with scarce samples. CNNBiGRU, leveraging its advantages in processing spatial and temporal sequence data, significantly enhances the performance of network intrusion detection.Furthermore, to improve the model's interpretability and meet users' needs for fairness, reliability, and transparency, we integrated the SHAP method to explain the model's decision-making process. The evaluation conducted on the publicly available datasets CICDDOS2019 and CICIDS2017 verified the effectiveness of the proposed method, showing that our system maintains high detection accuracy while also ensuring operational efficiency and system credibility.

Overall, the outcomes of this research contribute to building an efficient and reliable network intrusion detection system for consumer electronics, significantly improving the level of network intrusion detection in consumer electronics. We remain excited about future research directions, aiming to further advance the field of network network intrusion detection in consumer electronics. Although the CDDPM is efficient during model training, optimizing the time consumed to generate large volumes of samples remains necessary. Future research will focus on enhancing the efficiency of sample generation to better adapt to practical application needs.

VI. Acknowledgments

The paper is sponsored by Foundation of State Key Laboratory of Public Big Data under Grant No. PBD2022-10, Future Network Innovation Research and Application Projects under Grant No.2021FNA02006, and Development of an Ultra-large-scale Ubiquitous Network Quality Monitoring System Based on Trusted Edge Intelligence under Grant SYG202311.

References

[1] S. Saha, "Consumer Electronics Market: Consumer Electronics Market Analysis by Smart Home Devices, Wearable Devices, and Consumer Electronic Devices Product for 2023 to 2033," https://www.futuremarketinsights.com/reports/consumer-electronics-market, November 2023, REP-GB-433, Consumer Product, Future Market Insights Inc.

[2] X. Zhou, W. Huang, W. Liang, Z. Yan, J. Ma, Y. Pan, I. Kevin, and K. Wang, "Federated distillation and blockchain empowered secure knowledge sharing for internet of medical things," *Information Sciences*, vol. 662, p. 120217, 2024.

[3] M. Sayad Haghighi, F. Farivar, A. Jolfaei, A. B. Asl, and W. Zhou, "Cyber attacks via consumer electronics: Studying the threat of covert malware in smart and autonomous vehicles," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 825–832, 2023.

[4] A. Parnami and M. Lee, "Learning from few examples: A summary of approaches to few-shot learning," *arXiv preprint arXiv:2203.04291*, 2022.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[6] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[8] X. Zhou, X. Ye, K. I.-K. Wang, W. Liang, N. K. C. Nair, S. Shimizu, Z. Yan, and Q. Jin, "Hierarchical federated learning with social context clustering-based participant selection for internet of medical things applications," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 1742–1751, 2023.

[9] J. Gu and S. Lu, "An effective intrusion detection approach using svm with naïve bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021.

[10] P. Shettar, A. V. Kachavimath, M. M. Mulla, G. Hanchinmani *et al.*, "Intrusion detection system using mlp and chaotic neural networks," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE, 2021, pp. 1–4.

[11] X. Chen, P. Wang, Y. Yang, and M. Liu, "Resource-constraint deep forest-based intrusion detection method in internet of things for consumer electronic," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 2, pp. 4976–4987, 2024.

[12] Z. LI, P. WANG, and Z. WANG, "Flowgananomaly: Flow-based anomaly network intrusion detection with adversarial learning," *Chinese Journal of Electronics*, vol. 33, no. 1, pp. 1–14, 2024.

[13] D. Javeed, T. Gao, P. Kumar, and A. Jolfaei, "An explainable and resilient intrusion detection system for industry 5.0," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1342–1350, 2024.

[14] P. Wang, Z. Wang, F. Ye, and X. Chen, "Bytesgan: A semi-supervised generative adversarial network for encrypted traffic classification in sdn edge gateway," *Computer Networks*, vol. 200, p. 108535, 2021.

[15] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30 387–30 399, 2020.

[16] R. Duan, D. Li, Q. Tong, T. Yang, X. Liu, and X. Liu, "A survey of few-shot learning: An effective method for intrusion detection," *Security and Communication Networks*, vol. 2021, no. 1, p. 4259629, 2021.

[17] C. Chadebec and S. Allassonnière, "Data augmentation with variational autoencoders and manifold sampling," in *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 1*. Springer, 2021, pp. 184–192.

[18] Y. Guo, G. Xiong, Z. Li, J. Shi, M. Cui, and G. Gou, "Combating imbalance in network traffic classification using gan based oversampling," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–9.

[19] X. Zhou, X. Zheng, T. Shu, W. Liang, K. I.-K. Wang, L. Qi, S. Shimizu, and Q. Jin, "Information theoretic learning-enhanced dual-generative adversarial networks with causal representation for robust ood generalization," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2023.

[20] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[21] R. Ni, M. Goldblum, A. Sharaf, K. Kong, and T. Goldstein, "Data augmentation for meta-learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8152–8161.

[22] X. Zhou, X. Zheng, X. Cui, J. Shi, W. Liang, Z. Yan, L. T. Yang, S. Shimizu, and K. I.-K. Wang, "Digital twin enhanced federated reinforcement learning with lightweight knowledge distillation in mobile

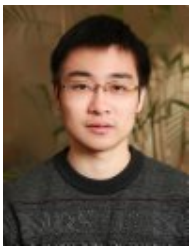
networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3191–3211, 2023.
[23] X. Zhou, Q. Yang, Q. Liu, W. Liang, K. Wang, Z. Liu, J. Ma, and Q. Jin, "Spatial–temporal federated transfer learning with multi-sensor data fusion for cooperative positioning," *Information Fusion*, vol. 105, p. 102182, 2024.
[24] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for iot networks," *Future Generation Computer Systems*, vol. 127, pp. 276–285, 2022.
[25] Y. Chen, J. Wang, T. Yang, Q. Li, and N. A. Nijhum, "An enhancement method in few-shot scenarios for intrusion detection in smart home environments," *Electronics*, vol. 12, no. 15, p. 3304, 2023.
[26] S. De, M. Bermudez-Edo, H. Xu, and Z. Cai, "Deep generative models in the industrial internet of things: a survey," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 5728–5737, 2022.
[27] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.
[28] Z. Li, Y. Zhu, and M. Van Leeuwen, "A survey on explainable anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 1, pp. 1–54, 2023.
[29] C. Wu, S. Shao, C. Tunc, P. Satam, and S. Hariri, "An explainable and efficient deep learning framework for video anomaly detection," *Cluster computing*, pp. 1–23, 2022.
[30] K. Böhmer and S. Rinderle-Ma, "Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users," *Information Systems*, vol. 90, p. 101438, 2020.
[31] I. B. Kraiem, F. Ghozzi, A. Péninou, G. Roman-Jimenez, and O. Teste, "Human-interpretable rules for anomaly detection in time-series," in *INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY*. OpenProceedings. org, 2021, pp. 457–462.
[32] T. Schlegl, S. Schlegl, N. West, and J. Deuse, "Scalable anomaly detection in manufacturing systems using an interpretable deep learning approach," *Procedia CIRP*, vol. 104, pp. 1547–1552, 2021.
[33] A. Barbado, Ó. Corcho, and R. Benjamins, "Rule extraction in unsupervised anomaly detection for model explainability: Application to oneclass svm," *Expert Systems with Applications*, vol. 189, p. 116100, 2022.
[34] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *International Conference on Information Systems Security and Privacy*, vol. 2. SciTePress, 2017, pp. 253–262.
[35] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway," *IEEE Access*, vol. 6, pp. 55 380–55 391, 2018.
[36] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 international carnahan conference on security technology (ICCST)*. IEEE, 2019, pp. 1–8.
[37] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.
[38] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential lstm neural networks autoencoders," in *2018 26th signal processing and communications applications conference (SIU)*. IEEE, 2018, pp. 1–4.
[39] J. Sinha and M. Manollas, "Efficient deep cnn-bilstm model for network intrusion detection," in *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, 2020, pp. 223–231.
[40] K. S. Ngwenduna and R. Mbuvha, "Alleviating class imbalance in actuarial applications using generative adversarial networks," *Risks*, vol. 9, no. 3, p. 49, 2021.

Ze Zhang is currently pursuing the Master's degree in Information Network at Nanjing University of Posts and Telecommunications. He was born in Zhenjiang, Jiangsu, China, in 2000. He received his Bachelor's degree in Network Engineering in 2022. His research interests include computer networks, network security, network intrusion detection and analysis.

Pan Wang received the BS degree from the Department of Communication Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001, and the PhD degree in Electrical and Computer Engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2013. He is currently a Professor in the School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests include cyber security and communication network security, network measurements, Quality of Service, Deep Packet Inspection, SDN, big data analytics and applications. From 2017 to 2018, he was a visiting scholar of University of Dayton (UD) in the Department of Electrical and Computer Engineering.

Tianqi Zhang is an undergraduate student at Nanjing University of Posts and Telecommunications. His research areas include intrusion detection and deep learning.

Minyao Liu is currently pursuing her master's degree at Nanjing University of Posts and Telecommunications. She received bachelor's degree in Management from NJUPT in 2022. Her research areas include traffic identification, deep learning and anomaly detection.



Xiaokang Zhou (M'12) is currently an associate professor with the Faculty of Business Data Science, Kansai University, Japan. He received the Ph.D. degree in human sciences from Waseda University, Japan, in 2014. From 2012 to 2015, he was a research associate with the Faculty of Human Sciences, Waseda University, Japan. He was a lecturer/associate professor with the Faculty of Data Science, Shiga University, Japan, from 2016 to 2024. He also works as a visiting researcher with the RIKEN Center for Advanced Intelligence Project (AIP), RIKEN, Japan, since 2017. Dr. Zhou has been engaged in interdisciplinary research works in the fields of computer science and engineering, information systems, and social and human informatics. His recent research interests include ubiquitous computing, big data, machine learning, behavior and cognitive informatics, cyber-physical-social systems, and cyber intelligence and security. Dr. Zhou is a member of the IEEE CS, and ACM, USA, IPSJ, and JSAI, Japan, and CCF, China.