Journal of Network Intelligence ©2025 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **10**, Number **2**, May **2025**

An IoT Intrusion Detection Method Combining GAN and Transformer Neural Networks

Zhi-Xian Zheng
Fujian Chuanzheng Communications College, Fuzhou 350007, China
378889925@qq.com

Feng Chen*
Fuzhou Institute of Scientific and Technological Information, Fuzhou 350007, China
138849323@qq.com

*Corresponding author: Feng Chen

Received August 28, 2024; revised November 29, 2024; accepted March 3, 2025.

Abstract. *Tackling the security issues inherent in the Internet of Things (IoT) necessitates the critical deployment of Intrusion Detection Systems (IDS). Presently, machine learning and deep learning techniques are widely employed for IoT intrusion detection. However, the imbalance in relevant datasets leads models to predominantly learn the characteristics of majority class data, neglecting those of minority class data, thereby significantly impairing detection accuracy. Additionally, in deep learning-based IDS, attackers can deceive the system by adding perturbations, causing it to misidentify malicious traffic as benign. To counter this, adversarial sample generation methods based on Generative Adversarial Networks (GANs) are employed. GANs use neural ordinary differential equations as generators and Wasserstein distance as the loss function, allowing the generator to better learn the distribution of genuine samples and improve the quality of adversarial samples. This approach allows the model to better learn the characteristics of underrepresented data, improving overall detection accuracy. A model combining Transformers with Bidirectional Gated Recurrent Units (Bi-GRU) is proposed for IoT intrusion detection. The Transformer encoder captures global relationships and performs initial feature extraction on input data, while the Bi-GRU network extracts long-distance dependency features, preserving the sequential characteristics of the data. A Multi-Layer Perceptron (MLP) further extracts deep features, and a Softmax classifier ultimately provides the classification results. The efficacy of the proposed approach was substantiated through testing on the publicly available ToN IoT dataset for intrusion detection within IoT environments. Experimental results show that the GAN-Transformer framework achieved average precision, recall, and F1 score of 97.62%, 97.66%, and 97.64%, respectively. When compared with other cutting-edge techniques, the proposed approach exhibited markedly enhanced detection performance.*

Keywords: IoT; Intrusion Detection; Generative Adversarial Networks; Transformer; Multi-Layer Perceptron; Neural Ordinary Differential Equations; Wasserstein Distance

1. Introduction.

The rapid development of Internet of Things (IoT) technology has led to its widespread application in various aspects of social life, including smart homes, industrial control, smart vehicles, smart healthcare, and environmental monitoring [1]. The advantages of IoT primarily lie in easy data access, faster and more stable communication, effective information sharing, cost reduction, and automation. These factors have significantly advanced the growth of the Fourth Industrial Revolution [2]. By 2025, the total number of IoT-connected devices worldwide is expected to reach 24.6 billion. The integration of sensors and devices with the internet has greatly enhanced efficiency in both production and daily life. However, the proliferation of IoT devices on networks has highlighted numerous issues, particularly the massive information traffic generated by these devices, necessitating reliable intelligent processing technologies to ensure system security [3].

In terms of IoT security incidents, the Mirai botnet in 2016 launched a denial-of-service attack with data rates reaching 620 Gb/s by controlling a large number of IoT devices, causing widespread internet outages in the United States [4]. In 2017, the WannaCry ransomware spread globally, affecting users in over 100 countries, causing significant financial losses, and severely impacting sectors such as finance and healthcare [5]. In 2018, hackers exploited security vulnerabilities in remote key systems to steal vehicles, including Teslas, using radio equipment and Raspberry Pi devices. In 2019, security flaws in IoT smart cameras were disclosed, making them vulnerable to remote illegal control by attackers, leading to severe information security breaches. In 2020, hackers conducted man-in-the-middle attacks using smart terminal routers like D-Link and Linksys, hijacking users' network access and directing them to malicious COVID-19-themed websites, deceiving users into downloading malware through fake notifications [6].

Intrusion Detection Systems (IDS) represent a forward-looking security solution that operates without the need to forward any traffic. These systems gather network information from IoT devices, including system logs, audit records, and network behavior, to scrutinize and identify potential threats, enabling swift responses to mitigate damage [7]. With the proliferation of IoT technology, these integrated systems have also opened new avenues for cybercrimes. The vulnerabilities of IoT terminal devices, low complexity of attacks, high hazard ratings, and the high reward-to-cost ratio for attackers make these devices easy targets for botnets [8]. However, defenders face more daunting challenges. In recent years, the frequency of attacks and intrusions on IoT devices has been increasing due to their inherent security flaws.

Early intrusion detection methods included models based on traffic data rules. These methods relied on packet format and data distribution rules for training intrusion detection models. Their limitations were significant, and their detection performance was poor. In recent years, the domain of intrusion detection has seen extensive application of artificial intelligence (AI) methodologies, notably machine learning (ML) and deep learning (DL) models [9]. Traditional ML algorithms for IoT intrusion detection, such as Random Tree, K-Nearest Neighbors (KNN), and Naive Bayes (NB), have been commonly used. Cybersecurity researchers have shown that feeding small, low-dimensional data samples into ML-based IDSs can achieve good detection performance [10]. However, a major drawback of these traditional methods is their low detection performance when handling high-dimensional large datasets. As the volume and dimensionality of intrusion detection datasets increase, the feature learning capability of traditional machine learning models diminishes, resulting in poor performance.

AI-driven Intrusion Detection Systems (IDSs) have garnered significant attention in both industry and academia, owing to their ability to efficiently process extensive datasets and detect zero-day attacks. However, with the ever-expanding scale of the internet and the explosive growth in network traffic, coupled with increasing complexity, enhancing the accuracy of malicious traffic detection and effectively differentiating between various types of malicious traffic have emerged as urgent challenges. Furthermore, DL models themselves are vulnerable to security issues. Adversarial attacks, where attackers introduce small perturbations into input data to deceive DL models into making incorrect classifications, are a significant concern [11]. These perturbations are often imperceptible to humans. The vulnerability of DL models to adversarial examples is common, and most neural networks in practical applications are likely to encounter them. DL-based IDSs typically comprise two components: one for feature extraction and another for classification. Existing IDSs often face two main problems: incomplete and inaccurate datasets, and vulnerability to attacks. Adversarial sample generation techniques can effectively address these issues [12].

IoT traffic is characterized by high dimensionality, temporal nature, and severe class imbalance. Current IoT intrusion detection methods have the following shortcomings: insensitivity to class-imbalanced samples, leading to weak generalization performance in handling class imbalance; weak modeling capability for global features of long sequences, preventing effective extraction of highly distinguishable IoT traffic features. To address the class imbalance issue and improve IoT intrusion detection accuracy, a framework combining Generative Adversarial Networks (GAN) and Transformer for IoT IDS is proposed. The Wasserstein GAN (WGAN) is used to learn the distribution of minority class data, avoiding the marginal distribution problem of generated data. A Transformer-based intrusion detection model is constructed to maintain connections between local features while focusing on important features. By introducing a multi-head attention (MHA) mechanism, the attention to local and global features is enhanced, uncovering the intrinsic relationships between different features. Additionally, Bidirectional Gated Recurrent Unit (Bi-GRU) is used to retain temporal features, and Multilayer Perceptron (MLP) is employed to extract deep features, achieving accurate IoT intrusion detection.

The structure of this paper is as follows. Section 1.1 presents an overview of related research on IoT intrusion detection. Chapter 2 delves into the details of the proposed IoT intrusion detection framework based on GANs and Transformers. Chapter 3 assesses the performance of the proposed methodology utilizing publicly available experimental datasets. Lastly, Chapter 4 concludes the paper and highlights potential avenues for future research.

1.1. **Related work.**

The operational landscape of IoT devices is susceptible to a multitude of threats, including those posed by hackers, malware, and viruses. These intruders aim to compromise data integrity in the network through various forms of attacks. Additionally, denial-of-service (DoS) attacks may be launched, depleting network and device resources [13]. The design and deployment of IDS can effectively address this issue. Thus, IDS has been widely used to monitor attacks in IoT environments, ensuring secure IoT communications.

Early intrusion detection models often relied on traditional ML methods. Deepa et al. [14] used the Random Tree algorithm on the KDD 99 dataset to develop an efficient IDS. Lin et al. [15] proposed a feature representation method based on KNN for traffic detection of intrusion behaviors. Bhosale et al. [16] introduced hybrid feature extraction into an improved Naïve Bayes algorithm to enhance the accuracy and performance of intrusion detection systems. Pavaiyarkarasi et al. [17] compared classic ML algorithms, such as decision trees (DT) and random forests (RF), on the Bot-IoT dataset, finding that the RF algorithm performed best.

Despite the strong performance of traditional ML methods in intrusion detection, their inadequacy in handling high-dimensional, nonlinear IoT data has become evident as networks have evolved. Fan et al. [18] developed a network intrusion detection model based on one-dimensional Convolutional Neural Networks (CNN) structure, using the KDD99 data sequence. This model utilized a cascade of "convolution–subsampling" layers to effectively characterize normal and abnormal data features, followed by multiple fully connected layers for classification. El-Sayed et al. [19] proposed an improved deep neural network (DNN) algorithm combined with feature engineering to select an optimal subset for training, improving the detection of anomalous traffic. Shams et al. [20] focused on context-aware feature extraction, proposing a method that served as a preprocessing step for CNN-based multi-class intrusion detection. Their experiments on various cybersecurity datasets yielded effective results. These methods primarily employed a two-stage process for feature selection and extraction, with model performance limited by complex preprocessing techniques. The development of Transformers, utilizing self-attention mechanisms to dynamically calculate global correlations in sequence data, has shown the ability to adaptively adjust attention weights and effectively extract deep global features from sequence data.

Scholars have extensively researched the class imbalance issue and achieved some success. Fernando et al. [21] designed a dynamically weighted balanced loss function to address class imbalance, effectively handling the issue and improving calibration performance. However, this approach did not analyze the data distribution itself and could not generate minority class data. Liu et al. [22] introduced an innovative difficult set sampling algorithm to tackle class imbalance. This method partitions the training set into difficult and easy subsets using the nearest neighbor algorithm. Subsequently, the K-Means algorithm is employed to condense the majority class samples within the difficult subset while amplifying the minority class samples. Joloudari et al. [23] combined over-sampling and under-sampling techniques to classify imbalanced data, using the Synthetic Minority Over-Sampling Technique (SMOTE) algorithm for over-sampling and complementary neural networks for under-sampling.

However, intrusion detection datasets perform poorly against adversarial attacks. Goodfellow et al. [24] put forward the rapid gradient sign approach, which adds symbolized noise to original samples based on loss function gradients, generating adversarial samples that mislead models into incorrect classifications. Xiao et al. [25] introduced a new adversarial attack method using auxiliary classifiers and GANs to model input samples and identify misclassified samples. Hu et al. [26] proposed an adversarial malware generation algorithm using GANs, maximizing the concept of maliciousness in generated adversarial samples. Han et al. [27] presented a particle swarm optimization-based GAN architecture to improve adversarial robustness in network intrusion detectors. This method significantly enhanced the evasion of adversarial samples but required considerable time and computational resources. Yan et al. [28] employed WGAN with a gradient penalty to evade detection by intrusion detection classifiers, synthesizing DoS traffic features based on normal traffic probability distributions.

Moreover, some methods focused on the power and storage limitations of IoT devices. Roy et al. [29] designed a lightweight IDS based on DL models to detect various attacks and anomalies in resource-constrained IoT systems. This model optimized for multicollinearity elimination, sampling, and dimensionality reduction, required minimal training data and time, and exhibited a low rate of false alarms coupled with a high detection rate across various datasets. Notably, unlike traditional resource-intensive IDS, this streamlined model is suitable for deployment on IoT devices constrained by limited power and storage capacity. Li et al. [30] introduced a collaborative IDS framework based on semi-supervised learning, employing a disagreement-driven algorithm. This approach aims to enhance the detection performance of individual detectors in IoT networks through data exchange and sharing. The evaluation results indicated that compared to traditional supervised ML classifiers, this approach enhances intrusion detection and minimizes false positives by autonomously utilizing unlabeled data. Moizuddin et al. [31] designed an IDS on the foundation of the Generalized Mean Grey Wolf Optimization (GMGWO) algorithm and the Elastic Contractive Auto Encoder (ECAE). This model demonstrated its ability to learn features from unlabeled data and generalize effectively to arbitrary test data without requiring additional training.

2. **Proposed Method.**

2.1. **Method Overview.** An IoT intrusion detection framework combining GAN and Transformer models was proposed. First, initial data were preprocessed and balanced with WGAN. The WGAN model employed neural ordinary differential equations (NODE) as the generator and used the Wasserstein distance as the loss function. This enabled the generator to more accurately grasp the distribution of genuine samples and to improve the quality of generated adversarial samples, creating a balanced dataset. The balanced data were subsequently input into a Transformer module, where connections between various features were established and more nuanced feature information was extracted through MHA. Subsequently, a BiGRU neural network was employed to elucidate the interconnections between prior and subsequent features, thereby preserving temporal dynamics. Finally, features were further extracted through an MLP and classified using a Softmax classifier to obtain the final results. The proposed model leveraged the advantages of various models while considering the relationships between different features and their temporal information. The overall structure is shown in Figure 1.

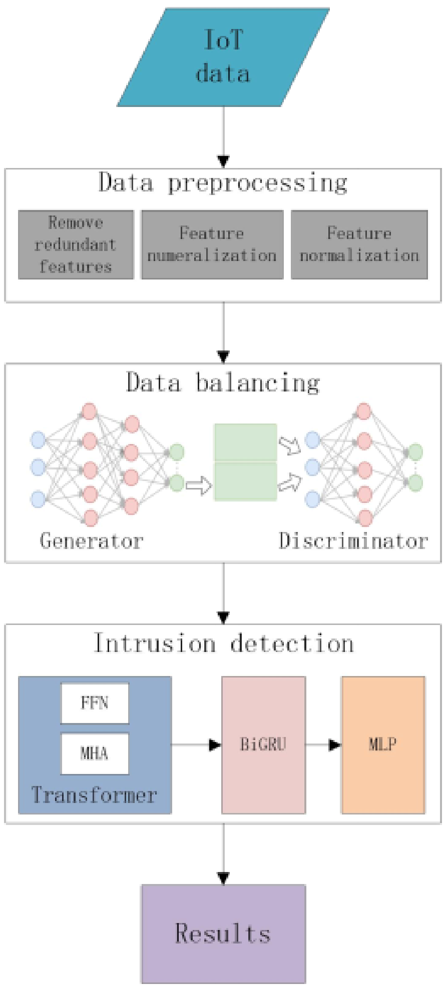

Figure 1. IoT intrusion detecto framework based on WGAN and transformer model

2.2. Data Preprocessing.

The original features in the dataset can significantly impact the classification performance of the neural network, and certain string-based features cannot be directly input into the network. String-based features refer to data features that consist of textual or categorical information, typically represented as sequences of characters or words. Unlike numerical features, these cannot be directly used by ML models like neural networks without first being converted into a numerical format. Therefore, it is crucial to effectively transform and process these features before feeding the data into the neural network for iterative training to ensure better classification outcomes.

In the experimental dataset, there are redundant features such as serial numbers and status codes that are either useless or detrimental to classification. These features need to be removed. Additionally, the dataset contains features represented as strings, which must be converted to numerical values using one-hot encoding. Numerical features can be further divided into discrete and continuous types. Continuous features, in particular, may have large value ranges that could adversely affect the decision boundaries of the neural network's classification. Hence, these features should be normalized. In summary, data preprocessing involves three main steps: removing redundant features, converting features to numerical values, and normalizing the features.

The scales and magnitudes of some continuous features vary significantly, resulting in large differences in their values. This can greatly affect the convergence of model weights during training. Therefore, it is essential to normalize continuous numerical features to mitigate the impact of these scale differences on the model. This study employs Min-Max normalization, which maps the values of continuous features to the [0,1] range. The Min-Max normalization formula is given as [32]:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

where x denotes the present magnitude of a specific attribute within the data instance. $x_{\min}$ signifies the minimal extent of that attribute, while $x_{\max}$ represents its maximal extent.

2.3. Data Balancing with WGAN.

Adversarial samples add subtle, human-imperceptible noise to the original samples, causing ML algorithms to misclassify them. These samples are denoted as x^*, where the disparity between x^* and the original input x is minimized according to a distance metric $d(\cdot)$, while still deceiving the target model f. Generally, the generated adversarial samples must satisfy the following constraints:

$$d(x^*, x) < \epsilon \quad \text{s.t.} \quad y(x^*) \neq y(x) \tag{2}$$

where ϵ denotes a constant constraining the perturbation size, and $y(\cdot)$ represents the model's predicted label. Attackers add slight perturbations to generate adversarial samples, causing the trained model to output incorrect labels. For IDSs, adversarial samples can significantly increase false alarm rates. These attacks have become increasingly sophisticated, posing a major threat to DL-based IDSs.

GAN consists of a pair of neural networks: the generator and the discriminator. The generator's objective is to synthesize data that mimics authentic data, whereas the discriminator's role is to differentiate between genuine and synthetic data. During GAN training, the generator receives random noise as input and generates data through successive neural network layers, producing data similar to genuine data. The discriminator classifies the genuine and synthetic data, treating the process as a binary classification problem. The GAN loss function can be expressed as [33]:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_r}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{3}$$

Where G represents the generator, D stands for the discriminator, p_r denotes the distribution of genuine data, and p_z signifies the distribution of noise. Both networks are trained simultaneously, continually optimizing to make the generated data indistinguishable from real data. The generator endeavors to mislead the discriminator, whereas the discriminator strives to precisely categorize authentic and fabricated data, akin to a gambling game where both players continually enhance each other's capabilities until the generator yields data that is indistinguishable from authentic data.

The proposed method introduced NODE and Wasserstein distance. NODE accelerates training and improves the quality and diversity of synthetic samples. The Wasserstein distance addresses training instability issues. Incorporating these techniques makes GAN training more efficient and stable.

2.3.1. *Neural Ordinary Differential Equations (NODE).* NODEs are used to describe the dynamic evolution of neural networks, modeling the continuous nature of data more naturally. In traditional neural networks, input data undergoes a series of linear and nonlinear transformations to produce the final output. Each layer corresponds to a fixed parameter matrix, learned through backpropagation. NODE, however, treats deep neural networks (DNN) as discrete approximations of ordinary differential equations (ODEs). Each neuron's state change over time is seen as the solution to an ODE. An ODE solver progressively advances the ODE solution, adjusting precision as needed to enhance the model's flexibility and accuracy.

In NODE, any DNN can be viewed as the initial condition of an ODE, representing the network's initial state. The ODE solver advances the ODE solution step by step to reach the final network state. This process leverages the solver's adaptive characteristics to adjust precision and step size. In NODE, forward propagation is transformed into solving a differential equation. Given an initial state (input), the differential equation is solved iteratively to find the next time step's state, which then becomes the initial state for the next step [34]. This iteration continues until the final state (output) is obtained, represented as:

$$y(t + \Delta t) = y(t) + f\big(y(t), t\big)\,\Delta t \tag{4}$$

Where $y(t)$ is the initial state, $f\big(y(t), t\big)$ is the differential equation, and Δt is the time step.

In NODE, backpropagation is performed using the adjoint method. This involves using an ODE solver to compute gradients concerning model parameters. The adjoint state is a vector of the same size as the original ODE system, with initial values being the negative gradient vector. During the adjoint ODE solution, backpropagation iterates from time T to t_0, updating the adjoint state at each solver step. The gradient vector is then used to update model parameters, enabling backpropagation.

By incorporating the framework of ODE, NODE enables more precise modeling of the generative process, thereby improving the quality of samples produced by GANs. Introducing more complex dynamics into the sample generation process enhances the diversity of the generated samples. Furthermore, when combined with GANs, NODE offers a flexible means to represent dynamic changes within the generative model, boosting the model's expressiveness and improving its ability to fit complex data distributions.

2.3.2. *WGAN.* The original GAN uses a cross-entropy loss function, while WGAN employs the Wasserstein distance, a smoother distance metric measuring the difference between two distributions. In WGAN, the discriminator outputs a real number rather than a probability between 0 and 1. The larger this value, the closer the sample is to the real distribution. The objective function for both the generator and discriminator is the

Wasserstein distance, not cross-entropy. This improved loss function stabilizes the training process and enhances sample quality. Compared to GANs, WGAN enhances training stability by optimizing the Wasserstein distance rather than the Jensen–Shannon divergence. The Wasserstein distance offers more meaningful gradient information and is less dependent on hyperparameters. Consequently, this leads to a more stable training process and more consistent gradient information, resulting in a greater diversity of generated adversarial samples.

WGAN introduces weight clipping to restrict the discriminator's weight range, preventing gradient explosion during training. This ensures the discriminator's parameters remain Lipschitz continuous, crucial for the convergence and stability of the WGAN loss function. Specifically, all discriminator parameters are clipped within a certain range, such as $[-c, c]$, where c is a constant. This operation maintains the discriminator function's gradient within reasonable bounds, preventing training instability and divergence. Theoretical analysis shows this approach significantly contributes to WGAN's convergence.

Adversarial training of NODE–WGAN can be framed as a minimax game:

$$\min_{G} \max_{D} J(\theta, x', y) \tag{5}$$

where $J(\theta, x', y)$ is the loss function, θ represents network parameters, x' denotes adversarial samples, y is the true label, and $D(x, x')$ represents the distance between original and adversarial samples.

2.3.3. *Generation of Adversarial Samples.* In the NODE–WGAN-based method for generating adversarial samples, a generator based on NODE is constructed. This generator consists of five NODE layers, each comprising a NODE and a linear layer. The NODE layer takes the output of the previous layer as its input and uses the NODE solver for forward propagation. The adjoint state method is employed for gradient calculation during backpropagation. Assuming the input to the l-th layer is h_{l-1} and the output is h_l, the forward propagation of the Generator can be expressed as:

$$h_l = \text{Linear}\big(\text{NODE}(h_{l-1})\big) \tag{6}$$

where $\text{NODE}(h_{l-1})$ represents the result of forward propagation of h_{l-1} through the NODE solver.

The discriminator is a neural network model structured as an MLP, tasked with classifying abnormal and normal traffic. Its aim is to reduce the Wasserstein distance separating genuine samples from those generated. Given a genuine sample x and a generated sample $\tilde{x}$, the discriminator loss function can be expressed as:

$$L_D = \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] \; - \; \mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})] \tag{7}$$

where p_{data} signifies the distribution of empirical data, and p_g denotes the distribution of synthetic data.

In each training iteration, the generator produces a set of adversarial samples. The discriminator is conditioned to differentiate these adversarial samples from authentic examples within the training dataset. The discriminator aims to minimize the Wasserstein distance between genuine and synthetic samples. Conversely, the generator strives to maximize the Wasserstein distance output by the discriminator:

$$L_G = -\,\mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})] \tag{8}$$

This loss function effectively evaluates the generator's performance and enhances training stability.

Adversarial training incorporates generated adversarial samples and original samples to train the classifier, thereby improving its robustness against adversarial attacks. Initially, the generator and discriminator in NODE–WGAN work in tandem. The generator creates adversarial samples similar to original ones but with perturbations. The discriminator determines whether a sample is original or adversarial. Specifically, through alternating training, the generator continually improves the quality of adversarial samples, making them more akin to real adversarial examples, while the discriminator enhances its ability to recognize these samples.

Subsequently, the classifier C is trained with both generator-produced adversarial samples $\tilde{x}$ and original samples x. This allows the classifier to learn to differentiate between the two and classify them correctly. Since adversarial samples are inherently complex, incorporating them into the training set helps the classifier learn intricate decision boundaries and increases the model's robustness. The classifier's loss function is given as:

$$L_C = -\mathbb{E}_{(x,y)\sim\text{data}}\big[y\log C(x)+(1-y)\log(1-C(x))\big] \quad - \mathbb{E}_{(\tilde{x},y)\sim\text{data}}\big[y\log C(\tilde{x})+(1-y)\log(1-C(\tilde{x}))\big] \tag{9}$$

where y is the true label of the samples. $C(x)$ and $C(\tilde{x})$ are the classifier's output probabilities for the original and adversarial samples, respectively.

2.4. **Classification Module.** The original Transformer architecture encompasses both encoder and decoder components. Due to the specific requirements of intrusion detection tasks and the fixed length of data in the dataset, only the encoder part of the Transformer is used in the proposed model, with certain parameters fine-tuned. Given that each data entry in the dataset has a fixed length, the autoregressive generation capability of the decoder—i.e., its mechanism for producing output step-by-step—is unnecessary. The encoder can effectively handle fixed-length input data and extract relevant information without the need for the decoder's generative functions. This design simplifies the model architecture, reduces computational complexity, and enhances training speed and efficiency. The structure of the Transformer-Encoder module is shown in Figure 2. Residual connections are used to prevent the vanishing gradient problem [35].

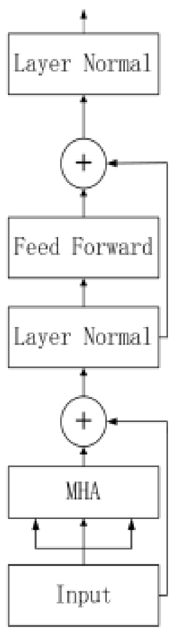

Figure 2. Encoder structure of the Transformer model

The encoder part includes an MHA mechanism and a feedforward neural network. Dot-product attention is used in the attention mechanism, involving three inputs: query Q, key K, and value V. The query and key are utilized to derive the weighting score allocated to each value. This score is subsequently employed to compute a weighted aggregate with the value, yielding the resultant output. Parallel computation with dot-product attention reduces training time. The MHA calculation is given as:

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \mathrm{head}_2, \ldots, \mathrm{head}_n)\, W^O \tag{10}$$

Therefore, Q, K, and V are computed using dot-product attention. W^O is a learnable matrix that linearly transforms the concatenated results of multiple attention heads.

Each attention head computes:

$$\mathrm{head}_i = \mathrm{Attention}(QW_i^Q,\, KW_i^k,\, VW_i^V) \tag{11}$$

The attention mechanism is then given as:

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

The data is then fed into BiGRU to capture temporal relationships between features. The BiGRU traverses the sequence in both forward and reverse orientations. The hidden states from both directions are amalgamated to encapsulate contextual insights from both preceding and forthcoming elements. The input layer feeds the input data separately into both networks.

The outputs from these networks are concatenated, and the result is shown as:

$$h_j = [\overrightarrow{h}_j;\ \overleftarrow{h}_j] \tag{13}$$

where h_j represents the final output after concatenating the forward and backward outputs for the j-th input data, and $j = 1, 2, \ldots, n$, with n being the total number of input data points. The concatenation of the last outputs from both networks serves as the input to the next layer. The output dimension is set to twice the input dimension to minimize model complexity as much as possible.

The feature vectors from the Transformer encoder are input into an MLP for further processing and feature extraction. The MLP is composed of one or more neuron layers, with each neuron in a given layer being interconnected with every neuron in the preceding layer. The hidden layer computation of the MLP is given as:

$$h = \sigma\big(W_1 x + b_1\big) \tag{14}$$

where x denotes input feature vector from the Transformer encoder. W_1 represents the weight matrix associated with the hidden layer, while b_1 denotes the bias vector pertinent to that layer. σ is an activation function. The output layer computation of the MLP is given as:

$$o = W_2 h + b_2 \tag{15}$$

where W_2 and b_2 are the weight matrix and bias vector of the output layer, respectively.

The MLP's output vector o is subsequently transformed by a Softmax function, yielding a probability distribution across the classes:

$$\hat{y}_i = \frac{e^{o_i}}{\sum_j e^{o_j}} \tag{16}$$

where o_i is the i-th element of the output vector o. $\hat{y}_i$ is the probability of the i-th class. The final output from the Softmax layer represents the probabilities of different classes (e.g., normal traffic, various types of intrusions). The class with the highest probability is selected as the predicted class for intrusion detection.

3. **Experiment.**

The experimental platform is configured with Windows 10 as the operating system, an Intel i5-12400 as the processor, 32.0 GB of memory, an NVIDIA GeForce RTX 3060 8 GB as the GPU, Python 3.8 as the programming language, and PyTorch 1.8.1 as the DL framework. The model's convergence speed and experimental results can be affected by different hyperparameter configurations. In this study, the hyperparameters are set as follows: a batch size of 512, a learning rate of 0.0001, a dropout rate of 0.5, an input dimension of 32, 8 attention heads, 64 hidden units in the FFN, 64 hidden units in the BiGRU, and an input dimension of 128 for the MLP.

3.1. **Dataset.**

The IoT intrusion detection dataset faces challenges such as data heterogeneity and severe class imbalance. To simulate real-world IoT and industrial IoT environments, the ToN IoT dataset was employed [36]. This dataset encompasses a range of authentic and malicious events, crafted across diverse operating systems, IoT environments, and industrial IoT services and networks. The training and test datasets comprise 461,043 IoT traffic entries (300,000 normal and 161,043 attack entries) with 44 features. The ToN IoT dataset includes nine types of anomalies: xss, DDoS, DoS, scanning, injection, ransomware, backdoor, password and mitm attacks. The dataset was partitioned into training and testing subsets in an 80:20 ratio, as outlined in Table 1.

Table 1. Statistical information of the experiment dataset

Category	Training samples	Testing samples	Total samples
normal	240,000	60,000	300,000
xss	16,000	4,000	2,000
DDoS	16,000	4,000	2,000
DoS	16,000	4,000	2,000
scanning	16,000	4,000	2,000
injection	16,000	4,000	2,000
ransomware	16,000	4,000	2,000
backdoor	16,000	4,000	2,000
password	16,000	4,000	2,000
mitm	834	209	1,043

3.2. **Evaluation Metrics.**

The experiment employs Precision, Recall, and F1-score as metrics to assess the model's classification performance. The true labels of samples and the model's predictions can result in four scenarios: true positives *TP*, true negatives *TN*, false positives *FP*, and false negatives *FN*. The aggregate sample count is the cumulative total of *TP*, *TN*, *FP*, and *FN*. In detail, *TP* refers to normal instances accurately classified as normal by the model. *TN* denotes anomalous instances accurately classified as anomalous. *FP* captures anomalous instances mistakenly identified as normal, while *FN* represents normal instances erroneously classified as anomalous.

Precision P is the proportion of correctly predicted abnormal samples among all samples predicted to be abnormal:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{17}$$

Recall R, also known as detection rate, is the proportion of correctly predicted abnormal samples out of all actual abnormal samples:

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{18}$$

The F1-score F_1 is the harmonic mean of P and R:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{19}$$

3.3. **Experiment Results.** Hyperparameters play a crucial role in DL models. Figure 3 illustrates the changes in loss values of the proposed method on the training set under different dropout rates. It can be observed from the figure that when the dropout rate is 0.7, the loss value decreases and the model performs poorly due to excessive dropout during training, which hinders the model from adequately learning data features. When the dropout rate is 0.3 or 0.5, the deep learning model exhibits varying degrees of overfitting after a certain number of training epochs. However, with a dropout rate of 0.5, the model fits the test set better than with a dropout rate of 0.3. Moreover, early stopping can be employed to mitigate the overfitting issue in DL models. Therefore, a dropout rate of 0.5 is chosen for the proposed method.

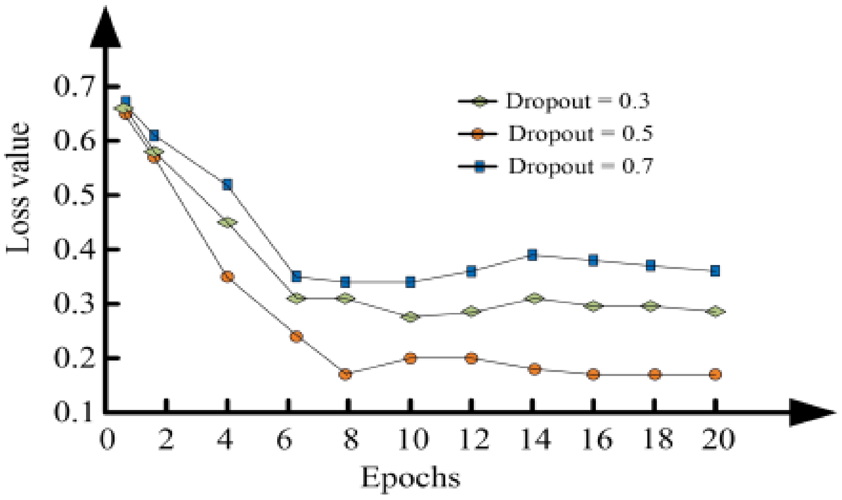

Figure 3. Training loss curves of the proposed framework.

In the proposed method, the number of encoder stacking layers of the Transformer model influences the model's depth. As illustrated in Table 2, with different number of encoder stacking layers, performance metrics initially improve with an increase in stacking layers but subsequently decline. Optimal performance is achieved at 3 layers. An appropriate increase in the number of encoder layers enhances the model's depth, enabling the extraction of abstract and highly discriminative features. However, excessive depth can lead to overfitting, which adversely affects detection performance.

Table 2. Encoder stacking layer number of the Transformer model on the detection performance (%).

Number of layers	P	R	F_1
1	88.97	92.47	90.69
3	97.62	97.66	97.64
5	97.58	96.85	97.21
7	97.55	93.28	95.37

The quantity of heads within the MHA mechanism influences the pattern extraction from network traffic, as illustrated in Table 3. For the MHA hyperparameter, the P, R, and F_1 of the proposed approach initially increase with the number of attention heads and then gradually decrease. Increasing the number of attention heads allows for the extraction of more sequence data feature patterns. However, as the number of feature patterns increases, the limited extraction capability of the feedforward neural network can lead to overfitting, causing a gradual decline in detection performance. Therefore, we decide to set the number of attention heads as 8.

Table 3. The number of heads of within the MHA on the detection performance (%).

Number of heads	P	R	F_1
2	90.73	91.44	91.08
4	94.86	92.81	93.82
8	97.62	97.66	97.64
10	97.89	95.03	96.44
12	98.63	93.92	96.22

The detection results of the proposed method on the experiment dataset for the 10 types of samples is given in Table 4. The proposed method achieved average P, R, and F_1 of 97.62%, 97.66%, and 97.64%, respectively, on the ToN IoT dataset. The method showed high performance in detecting Normal, Ransomware, XSS, and Backdoor samples, with P, R, and F_1 all above 98.88%. For DDoS, Password, and Scanning attacks, these metrics were above 96.35%. For DoS attacks, the P, R, and F_1 were 98.94%, 97.99%, and 98.46%, respectively, with recall being relatively lower. Injection attacks had P, R, and F_1 of 96.83%, 98.44%, and 97.62%, respectively, with precision being relatively lower. The MITM attack detection had P, R, and F_1 of 87.45%, 78.99%, and 82.78%, respectively, due to the small quantities of MITM instances (1,043, or 0.23% of the dataset). MITM attacks were often misclassified as Password attacks, which numbered 20,000. This misclassification is not solely due to class imbalance but also the inherent difficulty in detecting certain attack types.

Table 4. Detection results with the proposed method (%)

Category	P	R	F_1
normal	99.99	100.00	99.99
xss	98.88	99.81	99.34
DDoS	98.87	98.62	98.75
DoS	98.94	97.99	98.46
scanning	96.35	97.82	97.08
injection	96.83	98.44	97.62
ransomware	100.00	100.00	100.00
backdoor	100.00	99.97	99.99
password	98.91	98.90	98.90
mitm	87.45	78.99	82.78
Average	97.62	97.66	97.64

To further validate the superiority of the proposed method, comparative experiments were conducted against other intrusion detection methods. The methods compared include CANN [15], NB [16], CNN [18], DNN [19], and GMGWO-EACE [31]. Model architectures and parameters were designed based on references to achieve optimal performance. The detection performance was evaluated on the ToN IoT dataset using P, R, and F_1, the results are given in Table 5. The proposed method achieved the best detection results among all compared methods, with P, R, and F_1 of 97.62%, 97.66%, and 97.64%, respectively. Compared to the second-best GMGWO-EACE, the proposed method improved these metrics by 1.30%, 3.49%, and 1.89%, respectively. Compared to the traditional Transformer model, the proposed method improved P, R, and F_1 by 2.87%, 2.58%, and 3.02%, respectively. Against the traditional NB methods, the proposed method improved P, R, and F_1 by 27.77%, 12.23%, and 20.45%, demonstrating its superiority and effectiveness.

Table 5. Comparative experiment results of IoT intrusion detection (%)

Methods	P	R	F_1
CANN [15]	72.33	82.74	77.32
NB [16]	69.85	85.43	76.84
CNN [18]	90.87	91.23	91.17
DNN [19]	92.75	93.49	93.20
GMGWO-EACE [31]	96.32	94.18	95.40
original Transformer	94.75	95.08	94.27
Proposed method	**97.62**	**97.66**	**97.64**

4. **Conclusion.** This study has explored novel strategies for improving the performance of IDS in the IoT domain. The integration o GAN with DL models addresses several critical challenges. Firstly, GANs have been successfully employed to generate high-quality adversarial samples, which helps in training more robust IDS models. By using NODE and Wasserstein distance, the GAN framework enhances the generation of adversarial examples and improves the model's ability to handle perturbations. Secondly, the combination of Transformer and Bi-GRU in the proposed framework effectively captures both global and sequential dependencies within the data. This hybrid approach allows for a more comprehensive feature extraction process, improving the model's overall performance in detecting anomalies. The validation results on the ToN IoT dataset underscore the efficacy of the proposed method. The GAN-Transformer approach demonstrates superior performance metrics, achieving high precision, recall, and F1 scores, which signifies a significant advancement over existing methods. This improvement can be attributed to the effective handling of dataset imbalance and the mitigation of adversarial attacks. While this study represents a significant advancement in IDS for IoT systems, ongoing research is essential to address emerging challenges and opportunities. Several key areas include investigating advanced GAN architectures,expanding the framework to handle multi-modal data, eveloping methods to adapt the models in real-time to evolving threats and changing network conditions, explore optimization techniques and hardware accelerations to make these models more scalable and efficient for large-scale IoT networks, and investigate how these techniques impact user privacy and data security.

REFERENCES

[1] J. A. Heidari and M. A. Jabraeil Jamali, "Internet of Things intrusion detection systems: a comprehensive review and future directions," *Cluster Computing*, vol. 26, no. 6, pp. 3753–3780, 2022.

[2] S. N. Swamy and S. R. Kota, "An Empirical Study on System Level Aspects of Internet of Things (IoT)," *IEEE Access*, vol. 8, pp. 188082–188134, 2020.

[3] B. Chander, S. Pal, D. De, and R. Buyya, "Artificial Intelligence-based Internet of Things for Industry 5.0," in *Artificial Intelligence-based Internet of Things Systems*, pp. 3–45, 2022.

[4] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[5] M. Akbanov, V. G. Vassilakis, and M. D. Logothetis, "WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms," *Journal of Telecommunications and Information Technology*, vol. 1, no. 2019, pp. 113–124, 2019.

[6] A. Alanazi and A. Gumaei, "A Decision-Fusion-Based Ensemble Approach for Malicious Websites Detection," *Applied Sciences*, vol. 13, no. 18, p. 10260, 2023.

[7] P. R. Maidamwar, M. M. Bartere, and P. P. Lokulwar, "Intrusion Detection Systems in IoT: Techniques, Datasets, and Challenges," in *Computing Technologies and Applications*, pp. 103–142, 2021.

[8] X. Zhang, O. Upton, N. L. Beebe, and K.-K. R. Choo, "IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers," *Forensic Science International: Digital Investigation*, vol. 32, p. 300926, 2020.

[9] A. Thakkar and R. Lohiya, "A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2020.

[10] N. Saran and N. Kesswani, "A comparative study of supervised Machine Learning classifiers for Intrusion Detection in Internet of Things," *Procedia Computer Science*, vol. 218, pp. 2049–2057, 2023.

[11] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[12] A. Alotaibi and M. A. Rassam, "Adversarial Machine Learning Attacks against Intrusion Detection Systems: A Survey on Strategies and Defense," *Future Internet*, vol. 15, no. 2, p. 62, 2023.

[13] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Express*, vol. 7, no. 2, pp. 177–181, 2021.

[14] A. J. Deepa and V. Kavitha, "Efficient intrusion detection system using random tree," *International Journal of Enterprise Network Management*, vol. 6, no. 4, p. 275, 2015.

[15] W. Lin, S. Ke, and C. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015.

[16] K. S. Bhosale, M. Nenova, and G. Iliev, "Modified Naive Bayes Intrusion Detection System (MNBIDS)," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, IEEE, 2018, pp. 291–296.

[17] R. Pavaiyarkarasi, T. Manimegalai, S. Satheeshkumar, K. Dhivya, and G. Ramkumar, "A Productive Feature Selection Criterion for Bot-IoT Recognition based on Random Forest Algorithm," in *2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE, 2022, pp. 539–545.

[18] J. Fan and L. Kong, "Intrusion Detection Algorithm Based on Convolutional Neural Network," *Transactions of Beijing Institute of Technology*, vol. 37, no. 12, pp. 1271–1275, 2017.

[19] O. A. El-Sayed, S. K. Fawzy, S. H. Tolba, R. S. Salem, Y. S. Hassan, A. M. Ahmed, and A. Khattab, "Deep Learning Framework for Accurate Network Intrusion Detection in ITSs," in *2021 International Conference on Microelectronics (ICM)*, IEEE, 2021, pp. 212–215.

[20] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13647–13665, 2021.

[21] K. R. M. Fernando and C. P. Tsokos, "Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Confidence Calibration of Deep Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2940–2951, 2022.

[22] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

[23] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, "Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks," *Applied Sciences*, vol. 13, no. 6, p. 4006, 2023.

[24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[25] C. Xiao, B. Li, J. Zhu, W. He, M. Liu, and D. Song, "Generating Adversarial Examples with Adversarial Networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IEEE, 2018, pp. 3905–3911.

[26] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," in *International Conference on Data Mining and Big Data*, Singapore: Springer Nature Singapore, 2022, pp. 409–423.

[27] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, and X. Yin, “Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2632–2647, 2021.

[28] Q. Yan, M. Wang, W. Huang, X. Luo, and F. R. Yu, “Automatically synthesizing DoS attack traces using generative adversarial networks,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 12, pp. 3387–3398, 2019.

[29] 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.

[30] W. Li, W. Meng, and M. H. Au, “Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments,” *Journal of Network and Computer Applications*, vol. 161, p. 102631, 2020.

[31] M. Moizuddin and M. V. Jose, “A bio-inspired hybrid deep learning model for network intrusion detection,” *Knowledge-Based Systems*, vol. 238, p. 107894, 2022.

[32] M. Shantal, Z. Othman, and A. A. Bakar, “A Novel Approach for Data Feature Weighting Using Correlation Coefficients and Min–Max Normalization,” *Symmetry*, vol. 15, no. 12, p. 2185, 2023.

[33] Li and Y. Sung, “INCO-GAN: Variable-Length Music Generation Method Based on Inception Model-Based Conditional GAN,” *Mathematics*, vol. 9, no. 4, p. 387, 2021.

[34] C. Filici, “Error estimation in the neural network solution of ordinary differential equations,” *Neural Networks*, vol. 23, no. 5, pp. 614–617, 2010.

[35] H. Chen, D. Jiang, and H. Sahli, “Transformer Encoder With Multi-Modal Multi-Head Attention for Continuous Affect Recognition,” *IEEE Transactions on Multimedia*, vol. 23, pp. 4171–4183, 2021.

[36] A. R. Gad, A. A. Nashat, and T. M. Barkat, “Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset,” *IEEE Access*, vol. 9, pp. 142206–142217, 2021.