

Article

Network Intrusion Detection Based on Global-Local Temporal Feature Fusion Using Transformer and BiLSTM

Zheng Zhang ^{1,2}, Haonan Li ², Liquan Chen ^{1,*} , Tianhao Hou ² , Yanan Liu ²  and Jie Hao ²

¹ School of Cyber Science and Engineering, Southeast University, Nanjing 211199, China; zhangzheng@jit.edu.cn

² School of Network Security, Jinling Institute of Technology, Nanjing 211169, China; 2507050019@stu.jit.edu.cn (H.L.); ttianhao@jit.edu.cn (T.H.); yanan.liu@jit.edu.cn (Y.L.); 2407050013@stu.jit.edu.cn (J.H.)

* Correspondence: lqchen@seu.edu.cn

Abstract

Network Intrusion Detection (NID) is essential for cybersecurity because network traffic and security threats are growing rapidly. However, current NID methods suffer from a critical limitation: they face inherent difficulties in jointly modeling global contextual dependencies and local fine-grained temporal features within network traffic, which leads to suboptimal detection accuracy. To address this, this paper proposes the Global and Local Temporal (GLT) NID model. This model utilizes the self-attention mechanism of the Transformer to extract global dependencies and complex feature correlations from network traffic sequences while combining the bidirectional recurrent structure of BiLSTM to capture local temporal features of traffic data. Additionally, to address the class imbalance issue in intrusion detection datasets, this paper introduces a power-smoothed soft weighting strategy for optimization. We tested the GLT model on the NSL-KDD and UNSW-NB15 datasets. On the NSL-KDD dataset, the model achieved 99.57% accuracy and 0.32% false positive rate. On the UNSW-NB15 dataset, it achieved 88.04% accuracy and 0.76% false positive rate. Compared to current state-of-the-art techniques, it demonstrates higher detection accuracy and lower false positive rates.

Keywords: network intrusion detection; hybrid deep learning; global features; local temporal features; class imbalance

1. Introduction

Driven by the rapid expansion of IoT, 5G, and cloud computing, global network traffic has experienced explosive growth. While this digital transformation enables seamless connectivity and information sharing, it also creates a broader attack surface for cyber threats [1]. As network environments become increasingly complex, attack methods have evolved from traditional denial-of-service attacks to more covert and sophisticated forms, such as exploits and advanced persistent threats [2]. At the same time, data imbalance issues can also affect model detection accuracy and false positive rates [3]. Network Intrusion Detection (NID) is a key part of cybersecurity. It monitors network traffic, identifies anomalies, and analyzes behaviors in real time. This process effectively detects malicious activities and unauthorized operations, providing strong support for security alerts and incident response. Consequently, developing an NID model that is real-time, highly efficient, and highly generalizable has become a pressing demand in both academia and industry [4,5].



Academic Editors: Zbigniew Kotulski and Aryya Gangopadhyay

Received: 27 April 2026

Revised: 10 June 2026

Accepted: 28 June 2026

Published: 3 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Early NID methods primarily relied on misuse detection, which identifies threats by matching network traffic against a database of known attack signatures. While highly accurate with low false positive rates for known threats, this method struggles to detect zero-day attacks and new variants [6]. To overcome this limitation, research has shifted toward anomaly detection. This approach establishes a baseline of normal network behavior and flags any deviations as potential attacks. Although this enables the detection of unknown threats, the dynamic nature of modern networks causes traditional anomaly detection to suffer from persistently high false positive rates (FPR) [7,8].

Researchers have widely adopted data-driven artificial intelligence (AI) technologies to improve the accuracy and robustness of anomaly detection [9]. In the early stages, researchers built detection models primarily using machine learning (ML) algorithms. However, traditional ML methods rely heavily on manual feature engineering. As a result, they struggle to process massive, high-dimensional modern network traffic [10]. In recent years, advances in computing power have enabled deep learning (DL) to transform NID research. DL models can automatically extract multi-level, high-dimensional features directly from raw traffic data. This capability eliminates the need for manual feature design. Furthermore, it significantly improves attack detection rates in massive and complex traffic. Ultimately, DL shifts the technology from manual feature engineering to automatic feature learning [11,12].

However, as attack obfuscation techniques evolve, deep learning models based on single architectures have gradually revealed their inherent limitations. For example, CNNs [13,14] cannot dynamically model time series; while LSTM [15] excel at time series processing, they are prone to losing long-range dependencies when dealing with extremely long sequences, and due to their serial computing mechanism, they lack a macro-level perception of global traffic [16]. Recently, the Transformer architecture [17] has proven effective at capturing global long-range dependencies using self-attention. Yet, Transformers struggle with local temporal fluctuations [18]. To address these issues, researchers have turned to hybrid architectures [19]. By combining the strengths of multiple single models, hybrid approaches capture both global correlations and local temporal fluctuations. This allows them to extract richer, multidimensional features, making hybrid design a key research direction to overcome the limits of current NID [20].

To address these challenges, this paper proposes the GLT NID model. First, the model applies Log1p feature smoothing and sliding window serialization to the raw traffic data. Next, it uses a multi-layer Transformer Encoder to capture hidden correlations across different time steps in the traffic sequence. Finally, the model introduces a BiLSTM [21] network to finely characterize the local temporal evolution patterns of packet arrivals. Furthermore, at the algorithmic level, this paper presents a soft-weighted cross-entropy optimization strategy based on low-order smoothing. This strategy effectively balances the model's high confidence in normal traffic with its sensitivity to rare covert attacks, all while preserving the original temporal structure.

The contributions of this paper can be summarized as follows:

- To address the shortcomings of single-model architectures, we proposed a multi-model approach to leverage their complementary strengths. Specifically, our approach combines a Transformer and a BiLSTM. The Transformer captures global long-range dependencies, while the BiLSTM handles micro-level bidirectional temporal reasoning. Together, they build a network intrusion detection model with powerful spatiotemporal representations. Ultimately, this combined model effectively improves the detection of complex covert attacks.
- We designed a soft-weighted optimization strategy to address the issue of data class imbalance. Specifically, we introduced a Log1p transformation to stretch the numer-

ical distribution of extremely imbalanced traffic features. Next, we combined this transformation with a power-smoothed soft-weighting mechanism. Ultimately, this combined strategy mitigates the high false positive rates that traditional models face in class-imbalanced scenarios.

- Experiments were conducted on two benchmark datasets: NSL-KDD [22] and UNSW-NB15 [23]. The results show that our model achieved overall accuracy rates of 99.57% and 88.04%, respectively. Compared to existing models, we achieved performance improvements ranging from 0.3% to 10% on the NSL-KDD dataset and from 0.5% to 10% on the UNSW-NB15 dataset, validating the model's ability to detect attacks.

The remainder of this paper is organized as follows: Section 2 reviews the current state of the art and recent advancements. Section 3 details the network intrusion detection model proposed in this paper. Section 4 presents experimental results, ablation studies, and a comparative analysis with existing mainstream methods. Section 5 summarizes the research findings and outlines future research directions.

2. Related Work

Cyberattack methods are constantly evolving, and encrypted traffic is now widely used. As a result, traditional NID models based on static signatures and rule matching struggle against massive, high-dimensional, and obfuscated modern threats [24]. To address this challenge, data-driven Machine Learning (ML) and Deep Learning (DL) have become core research directions in network security. These AI techniques use massive network flow data to learn complex behavioral and statistical patterns. This allows models to effectively identify unknown zero-day threats and stealthy attack variants [25–27].

Talukder et al. [28] proposed a machine learning intrusion detection model that integrates the Synthetic Minority Over-sampling Technique (SMOTE) with XGBoost feature selection. The model uses Random Forest (RF) for classification. It achieved strong binary classification accuracy on both the KDDCUP99 and CIC-MalMem-2022 datasets. However, these results are based on static memory features. It remains unclear whether the model can maintain stable and effective detection performance in real-world network traffic with complex temporal fluctuations.

Recently, Yuan et al. [29] proposed CANET, a hierarchical CNN-Attention model for network intrusion detection. This model combines convolutional neural networks with self-attention mechanisms to extract spatio-temporal features and introduces Equalization Loss v2, a dynamic gradient weighting strategy, to address the class imbalance issue in intrusion detection datasets. CANET has achieved promising detection performance on mainstream datasets including UNSW-NB15 and NSL-KDD. However, there are still some limitations to be further explored and optimized. First, the model adopts a hierarchical structure that applies convolution and pooling operations before the attention mechanism, which may affect the preservation of micro-temporal continuity inherent in raw network traffic data. Second, its dynamic gradient weighting mechanism needs to continuously track the ratio of positive to negative gradients for each class throughout the training process, which may lead to relatively high computational overhead and susceptibility to random fluctuations in batch data.

Dash et al. [30] proposed an optimized LSTM-based intrusion detection framework. The study used heuristic algorithms like PSO, JAYA, and the Sea Squirt Algorithm (SSA) to automatically tune LSTM hyperparameters, such as hidden-layer neurons and learning rates. This method improved accuracy on the NSL-KDD and CIC-IDS2017 datasets. However, single-LSTM architectures based on heuristic optimization still face limitations. The large search space of the iterative optimization leads to high computational and time costs, which reduces the model's agility in dynamic networks. Additionally, the model is mainly

restricted to local temporal features and lacks the ability to extract global correlations across the network traffic.

Given that standalone CNNs struggle with time series modeling and LSTMs lack high-dimensional abstraction, researchers have turned to hybrid architectures that combine the strengths of both. For example, Said et al. [31] proposed a CNN-BiLSTM network for intrusion detection. Their study first used Random Forests (RF) and Recursive Feature Elimination (RFE) to reduce the input features to ten, while applying random oversampling to balance the classes. The model then extracts spatial features via a CNN and captures temporal dependencies through a BiLSTM, achieving 84.23 percent accuracy on the UNSW-NB15 task. However, their oversampling technique merely replicates minority samples at the physical level, which often leads to severe overfitting during training.

Since Vaswani et al. introduced the Transformer architecture, its multi-head self-attention mechanism has shown outstanding capabilities in sequence modeling, making it widely adopted for network traffic feature extraction. For instance, Wang et al. [32] proposed a parallel Res-TranBiLSTM hybrid network for intrusion detection. This model employs Transformer's multi-head self-attention combined with BiLSTM to capture deep temporal features, and leverages ResNet in a parallel branch to extract spatial features. Although it achieves promising performance on multiple IoT datasets, this method still suffers from two inherent limitations that restrict its generalization capability. On the one hand, the mandatory reshaping of one-dimensional traffic sequences into two-dimensional pseudo-images to fit ResNet's input requirements violates the intrinsic protocol semantics of network traffic. On the other hand, the SMOTE-ENN oversampling technique employed in this work may inadvertently corrupt the fine-grained temporal dependencies between successive network packets.

While these directly concatenated hybrid architectures face issues with feature fidelity, resampling logic, and computational overhead, the Transformer's ability to capture global features across time steps has undoubtedly opened new possibilities for network intrusion detection. If architectural redundancies are streamlined and temporal integrity is maintained, the Transformer offers significant potential for optimization and broad application in this field [33–35].

3. Proposed Methodology

This section presents the design and implementation of the proposed network intrusion detection model, with its overall architecture illustrated in Figure 1.

As illustrated in Figure 1, raw data is first preprocessed using logarithmic smoothing, normalization, and sliding-window serialization. It then enters the embedding and spatial encoding modules, which map features into high-dimensional vectors and incorporate spatial information. Next, a three-layer Transformer encoder uses multi-head self-attention to capture long-range dependencies, while residual and feedforward networks further enhance the feature representation. A BiLSTM module subsequently extracts bidirectional temporal correlations to produce an enhanced feature vector. Finally, two fully connected layers perform dimensionality reduction, and a Softmax function outputs the classification results.

3.1. Global Feature Extraction

The model begins with an input embedding stage. Because the Transformer architecture lacks a recursive structure and cannot inherently perceive sequence order, we incorporate sine and cosine positional encodings. As defined in Equations (1) and (2), even positions are encoded using the sine function, while odd positions use the cosine function.

These encodings are added directly to the feature embedding vectors, allowing the model to perceive relative positions within the sequence.

$$PE_{(\text{pos}, 2i)} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (1)$$

$$PE_{(\text{pos}, 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (2)$$

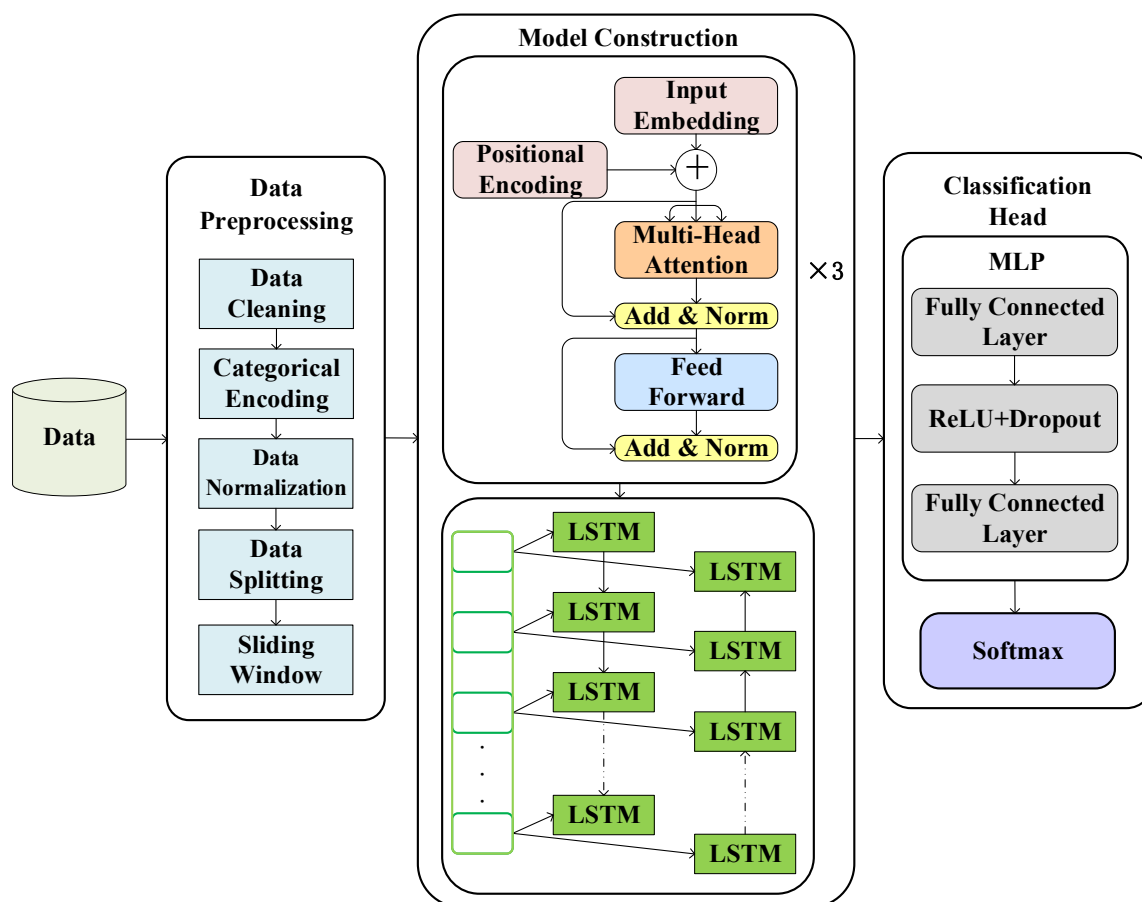


Figure 1. Overall structure of the GLT network intrusion detection model.

The final representation vector E input to the encoder is obtained by adding the linear projection result to the position encoding, as shown in Equation (3).

$$E = (XW_e + b_e) + PE \quad (3)$$

where X represents the preprocessed raw traffic feature matrix; W_e denotes the weight matrix for linear projection, which maps low-dimensional raw features into a high-dimensional embedding space; b_e is the bias term that, together with the weight matrix, performs the linear transformation; and PE is the introduced positional encoding.

The Transformer Encoder, composed of self-attention and feedforward layers, functions as the central component for global feature extraction. Its primary objective is to capture complex dependencies within traffic sequences from a macro perspective. In contrast to CNNs, which emphasize local receptive fields, the Transformer evaluates the pairwise correlations of all packets within a specific time window. Moreover, unlike sequential RNN models [36], the Transformer leverages multi-head self-attention to process

entire sequences in parallel. Consequently, it can identify covert attack patterns involving long-range dependencies, as depicted in Figure 2.

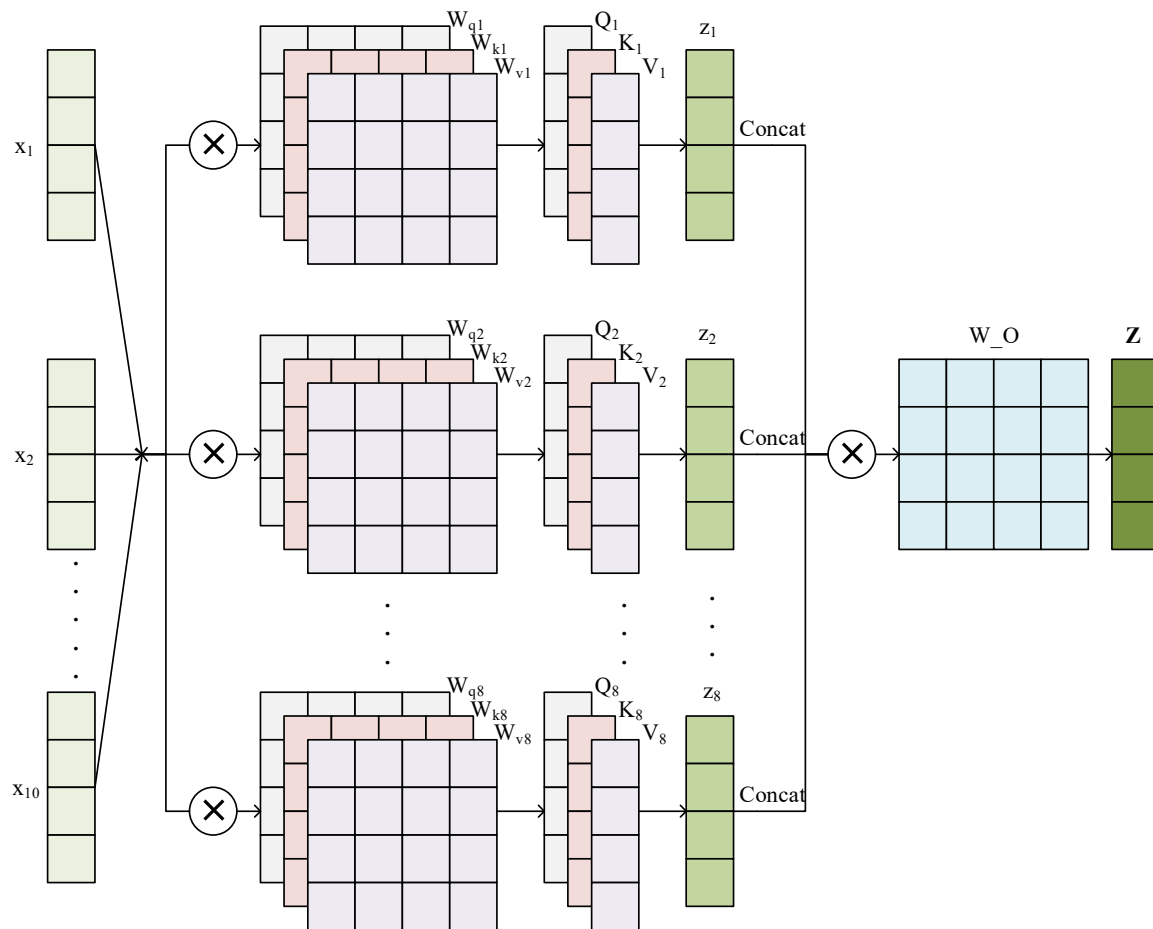


Figure 2. Detailed structure of the Multi-Head Attention mechanism.

To facilitate the extraction of multifaceted feature information, the architecture employs $h = 8$ attention heads. The input feature matrix E , derived from the preceding embedding layer, undergoes linear projection via three sets of learnable weight matrices. For each individual head i ($i = 1, 2, \dots, 8$), the corresponding query Q , key K , and value V matrices are defined by the equations: $Q_i = EW_{q_i}$, $K_i = EW_{k_i}$, and $V_i = EW_{v_i}$. As depicted in Figure 2, the input is projected into eight distinct representation subspaces in parallel. This configuration allows the model to concurrently identify diverse patterns within the network traffic features.

Internally, each attention head performs a dot-product operation between Q_i and K_i to quantify dependencies across sequence positions. A scaling factor of $\sqrt{d_k}$ is integrated to avoid Softmax saturation and the subsequent vanishing gradient problem. Following this, the attention weights are normalized through the Softmax function and used to compute a weighted sum of V_i , resulting in the head output Z_i as defined in Equation (4).

$$Z_i = \text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i \quad (4)$$

To maintain mathematical consistency, let d_{model} denote the overall feature embedding dimension of the model (set to 256). Consequently, d_k is defined as the dimension of the key vectors for each individual attention head. Given that we employed $h = 8$ parallel attention heads, d_k is explicitly calculated as $d_k = d_{\text{model}}/h = 256/8 = 32$.

This mechanism allows the model to integrate contextual information from throughout the sequence, modulated by the strength of its correlation with the current feature. Consequently, the model can effectively capture intricate long-range dependencies within the traffic data.

As shown in Figure 2, to integrate information from all subspaces, we concatenated the output vectors of the 8 heads: $Z_1, Z_2, \dots, Z_8$:

$$Z_{\text{concat}} = \text{Concat}(Z_1, Z_2, \dots, Z_8) \quad (5)$$

The concatenated feature matrix is then fused through an output linear layer W_O to obtain the final output of the MSA module:

$$\text{MSA}(E) = Z_{\text{concat}} W_O \quad (6)$$

After being processed by a residual connection and layer normalization, the MSA module's output serves as the input for a feedforward neural network (FFN). This design facilitates the extraction of complex nonlinear features while maintaining the operational stability of the deep network during training. The FFN is formulated as follows:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (7)$$

Here, x designates the FFN input vector. In the first encoder layer, x corresponds to E , whereas in the second and third layers, x represents the MSA output after undergoing residual connection and layer normalization. The $\max()$ function denotes the ReLU activation, facilitating non-linear modeling. The parameters $\{W_1, b_1\}$ and $\{W_2, b_2\}$ serve as the learnable weights and biases for the first and second linear transformation stages, respectively.

3.2. Local Temporal Refinement

While self-attention allows the Transformer to identify global dependencies, the model lacks the precision required to capture local, continuous temporal fluctuations. To address this and uncover underlying micro-temporal patterns, this paper integrated a Bidirectional Long Short-Term Memory (BiLSTM) network after the Transformer layer. This structure is depicted in Figure 3.

The BiLSTM primarily consists of two parts: a forward-propagating LSTM and a backward-propagating LSTM. The forward LSTM processes the input from time $t = 1$ to $t = 10$, computing the hidden state $\vec{h}_t$. This enables the model to capture the cumulative historical effects of attack behavior. The backward LSTM processes the input in reverse order from $t = 10$ to $t = 1$, computing the hidden state $\overleftarrow{h}_t$. This allows the model to utilize subsequent temporal information to refine its assessment of the current time step. The final output at each time step fuses the forward and backward hidden states, ensuring that the result incorporates information from both past and future time steps.

Each LSTM unit in Figure 3 is not a simple nonlinear activation function, but a complex gated memory block. Figure 4 details the internal computational logic of a single LSTM unit at time step t .

As shown in Figure 4, the current-time flow features x_t and the previous-time hidden state h_{t-1} are first concatenated, then linearly mapped through three weight matrices W_f , W_i , and W_o , respectively, and finally passed through sigmoid activation functions to yield three state vectors f_t , i_t , and o_t . The candidate cell state $\tilde{C}_t$ is computed by applying the tanh activation function to the linear transformation of the concatenated inputs x_t and h_{t-1} with the weight matrix W_c . The functions and calculation formulas for each gate unit are as follows:

Driven by the previous hidden state h_{t-1} and the current input x_t , the forget gate generates the control vector f_t . This vector determines the extent to which irrelevant information is discarded from the previous cell state c_{t-1} . The corresponding formula is presented below:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (8)$$

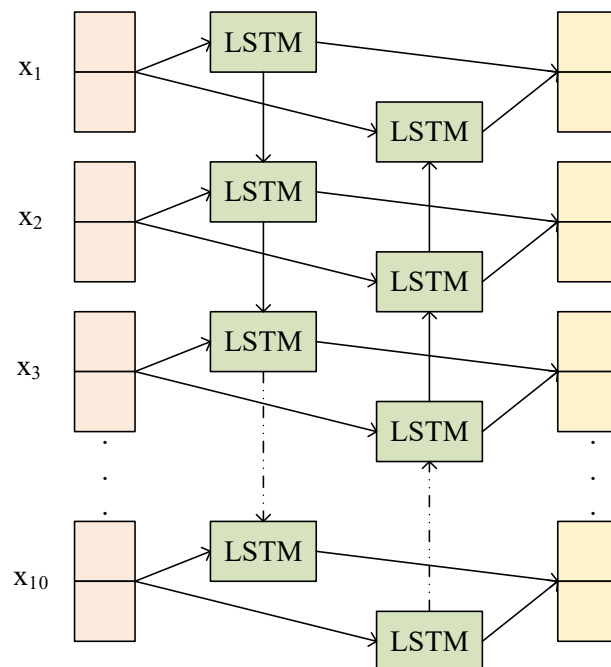


Figure 3. Unrolled architecture of the BiLSTM layer.

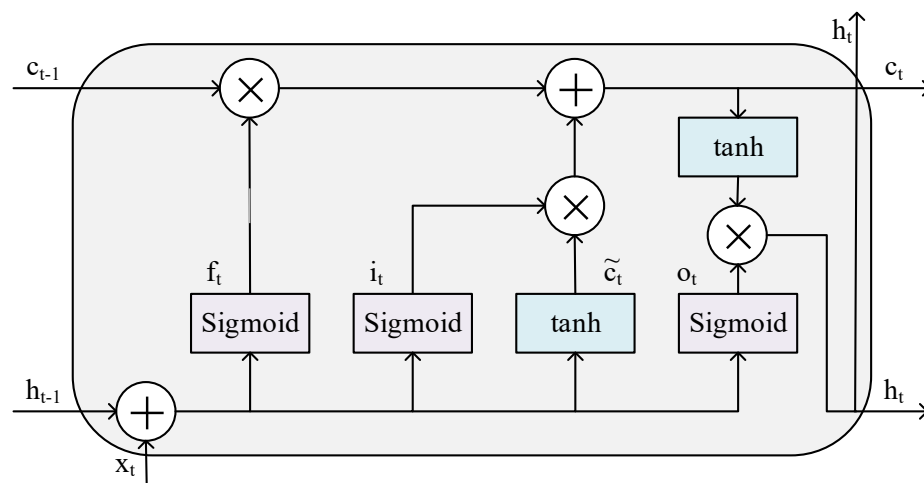


Figure 4. Internal structure of a single LSTM unit.

Using the previous hidden state h_{t-1} and the current input x_t , the input gate produces a control vector i_t . This vector, along with the candidate cell state $\tilde{c}_t$, determines the new attack-related information to be integrated into the cell state. The corresponding calculation is as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (9)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (10)$$

The old cell state is filtered through the forget gate and combined with the candidate memory weighted by the input gate to form the cell state at the current time step C_t :

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (11)$$

Based on the previous hidden state h_{t-1} and the current input x_t , the output gate computes a control vector o_t . Combined with the updated cell state C_t , this vector determines the current hidden state h_t as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (12)$$

$$h_t = o_t \odot \tanh(C_t) \quad (13)$$

where $\sigma()$ is the sigmoid activation function, and b_f , b_i , b_c , and b_o are bias terms.

After $T = 10$ time steps of recursive computation, the model obtains bidirectional features at each time step that integrate both past and future temporal context. To derive the final representation of the sliding window, the forward and backward features at each time step t are concatenated into a high-dimensional vector.

$$V_t = \left[\vec{h}_t \oplus \overleftarrow{h}_t \right] \quad (14)$$

Since the final time step $t = 10$ of the BiLSTM aggregates the complete temporal information of the entire sequence, we extracted the V_{10} as the final feature vector for this traffic window and feed it into the subsequent classification network.

3.3. Power-Smoothed Soft Weighting Strategy

In real-world network intrusion detection, traffic data typically exhibits extreme class imbalance. Traditional deep learning models rely on standard cross-entropy loss, which assumes nearly equal prior probabilities for each class. However, under severe imbalance, the training process is dominated by majority samples. This causes the model to prioritize high overall accuracy by predicting most samples as belonging to the majority class. This bias leads to two critical issues: low recall for minority classes, which renders the detection of high-risk rare attacks virtually impossible, and overfitting to the majority class, where the model learns statistical frequency biases rather than meaningful decision boundaries [37,38].

Although oversampling or undersampling are common solutions, in time-series data, randomly adding or removing samples disrupts the time-dependent structure of packet arrivals, causing the LSTM module to fail.

To resolve these issues without compromising the data's temporal structure, this paper proposed a soft weighting strategy based on power-smoothing. We aimed to prevent the model from ignoring minority classes while avoiding excessively high weights that could make it overly sensitive to noise. By introducing an extremely small smoothing exponent α to the inverse class frequency weights, we compressed the weight distribution. This enables the model to focus on minority samples while still prioritizing the primary classes that represent the majority of network traffic. The process is defined as follows:

Initial Calculation of Inverse Class Frequency: First, we defined a base weight W_c for each class c that is inversely proportional to the total number of samples in that class.

$$W_c = \frac{N_{total}}{K \times N_c} \quad (15)$$

Here, N_{total} represents the total number of samples, K is the total number of classes, and N_c is the number of samples in class c .

Power-Law Smoothing: To mitigate the negative impact of extreme weights, we introduced a smoothing coefficient α to perform nonlinear compression on W_c .

$$W'_c = (W_c)^\alpha \quad (16)$$

In this study, after extensive experimental comparisons, $\alpha = 0.15$ was selected as the final setting. When $\alpha \rightarrow 1$, standard inverse frequency weighting is applied (placing extreme importance on minority classes). When $\alpha \rightarrow 0$, no weighting is applied (minority classes are ignored). When $\alpha = 0.15$, the variance of the weights is significantly reduced. This not only preserves appropriate attention to minority classes but also ensures that the model's feature learning for majority classes remains undisturbed.

Finally, we incorporated the smoothed weights W'_c into the cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^K W'_c \cdot y_{i,c} \log(\hat{y}_{i,c}) \quad (17)$$

where N denotes the mini-batch size used during the training phase (set to 256 in our experiments). K represents the total number of classification categories (e.g., $K = 10$ for the UNSW-NB15 dataset). The term W'_c is the power-law smoothed weight for class c . Furthermore, $y_{i,c} \in \{0, 1\}$ is explicitly defined as the one-hot encoded ground truth label indicator, which equals 1 if the i -th sample truly belongs to class c , and 0 otherwise. Finally, $\hat{y}_{i,c}$ represents the predicted softmax probability that the i -th sample belongs to class c .

The proposed soft weighting strategy demonstrates significant advantages over traditional balancing techniques. By optimizing solely at the loss function level, the method preserves the underlying temporal structure of the traffic data, which is critical for BiLSTM performance. The use of a low-order smoothing coefficient mitigates potential spikes in misclassification that often result from an overemphasis on specific class frequencies. Empirical validation on the NSL-KDD and UNSW-NB15 datasets yielded accuracy rates of 99.57% and 88.04%, with corresponding false positive rates of 0.32% and 0.76%. These metrics underscore the model's efficacy in handling the challenges of skewed data distributions in network intrusion detection.

4. Experiments and Results

This section evaluates the effectiveness and robustness of the proposed NID model through a series of systematic experiments. It begins with a detailed description of the selected benchmark datasets and their statistical characteristics, followed by an explanation of the data preprocessing methods. We then introduce the experimental setup and evaluation metrics. The core section analyzes the model's detection performance on the NSL-KDD and UNSW-NB15 datasets. Finally, ablation experiments validate the contributions of each component in the hybrid architecture, while a comparative analysis highlights the model's advantages over current mainstream NID methods.

4.1. Dataset Description

To evaluate the model's generalization ability across different network environments, this paper selected two representative public datasets: NSL-KDD and UNSW-NB15.

NSL-KDD is an enhanced version of the classic KDDCup99 dataset and resolves the severe record redundancy found in the original version. The dataset comprises 41-dimensional features, including basic TCP connections, content-based characteristics, and time-based traffic patterns. It identifies four primary attack categories: DoS, Probe, R2L, and U2R. While NSL-KDD offers a more balanced class distribution than KDDCup99, the R2L and U2R categories remain rare. These low-sample classes present a significant challenge to the model's detection sensitivity and generalization performance.

UNSW-NB15 is a modern network traffic dataset developed by the Australian Cyber Security Laboratory. Compared to NSL-KDD, it more accurately reflects the complex

landscape of contemporary cyber threats. The dataset comprises 49 features, spanning basic flow, content, temporal, and connection characteristics. Attacks are organized into nine primary families: Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms. Samples of Normal and Generic traffic account for the vast majority, while categories such as Worms and Shellcode are extremely scarce. This profound class imbalance is critical for evaluating the real-world performance of NID models.

In this study, rather than utilizing the official, predefined training and testing partitions, we adopted a custom re-splitting strategy. Specifically, we performed a file-order concatenation of the official train and test files into a global data pool, followed by an 80/20 cut (80% for training and 20% for testing).

Tables 1 and 2 present the data statistics of NSL-KDD and UNSW-NB15 before and after partitioning, respectively.

Table 1. Class distribution statistics of the NSL-KDD dataset.

Class	KDDTrain+	KDDTest+	Training Set 80%	Testing Set 20%
Normal	67,343	9711	63,540	13,514
DoS	45,927	7458	43,302	10,083
Probe	11,656	2421	10,993	3084
R2L	995	2754	805	2944
U2R	52	200	182	70
Total	125,973	22,544	118,822	29,695

Table 2. Class distribution statistics of the UNSW-NB15 dataset.

Class	Training	Testing	Training Set 80%	Testing Set 20%
Normal	56,000	37,000	74,333	18,667
Generic	40,000	18,871	47,121	11,750
Exploits	33,393	11,132	35,527	8998
Fuzzers	18,184	6062	19,443	4803
DoS	12,264	4089	13,157	3196
Reconnaissance	10,491	3496	11,229	2758
Analysis	2000	677	2135	542
Backdoor	1746	583	1853	476
Shellcode	1133	378	1204	307
Worms	130	44	138	36
Total	175,341	82,332	206,140	51,533

As shown in Table 1, the NSL-KDD dataset is highly challenging due to distribution discrepancies between the training and test sets. Despite data merging and re-splitting, severe class imbalance still persists in the datasets. In particular, the stealthy U2R attack only contains 70 samples within the entire test set of 29,695 entries. In contrast, the R2L class has only 805 samples in the training set but 2944 samples in the test set. Such an extremely few-shot test scenario imposes stringent requirements on the model's capability to capture subtle temporal features.

Table 2 illustrates the class distribution of the UNSW-NB15 dataset. Compared to NSL-KDD, this dataset exhibits a more severe class imbalance, which better reflects contemporary network environments. This poses significant challenges to the model's ability to learn from limited samples. Worms and Shellcode attacks represent less than 1 percent of the total samples, making them highly susceptible to being overlooked. Additionally, the Exploits category comprises diverse exploitation techniques that are difficult to distinguish from benign traffic. These characteristics underscore the necessity of our log-smoothing strategy and soft-weighted loss function for maintaining robust detection across all classes.

4.2. Data Preprocessing

The NSL-KDD and UNSW-NB15 datasets comprise both numerical and categorical attributes. Initially, categorical features were mapped to numerical space using label encoding. For numerical attributes, we opted for a logarithmic smoothing transformation rather than direct normalization. This choice prevents the compression of imbalanced data into restrictive intervals, thereby preserving distribution details and enhancing the model's sensitivity to varying traffic intensities. Following this transformation, min-max normalization was applied to standardize feature scales across dimensions. Finally, the processed records were structured into sequence tensors using a sliding window approach with a temporal step of $T = 10$. The transformation formula is expressed as:

For the input feature x , the logarithmic transformation is defined as

$$x' = \ln(x + 1) \quad (18)$$

After the logarithmic transformation, min-max normalization scales all features to the $[0, 1]$ range. This step eliminates scale differences across different dimensions and allows gradient descent to converge more quickly.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (19)$$

4.3. Experimental Setup

All experiments in this paper were conducted on a computer configured with a 64-bit 11th Gen Intel Core i5-11400H@2.70GHz processor, an NVIDIA GeForce RTX 3050 Ti graphics card, 16 GB of RAM, and running the Windows 11 operating system. The GLT network intrusion detection model proposed in this paper was implemented using Python 3.8. The deep learning framework selected was PyTorch (v1.13). Leveraging its efficient tensor operations and modular interface, we constructed a hybrid Transformer and BiLSTM model and utilized the CUDA interface to accelerate training on an NVIDIA GPU.

During the data preprocessing and feature engineering phases, we used the Pandas library to clean and integrate large-scale traffic data. To standardize and digitize the features, we employed the MinMaxScaler and LabelEncoder modules from Scikit-learn. Furthermore, in the evaluation stage, we utilized the `classification_report` and `confusion_matrix` functions from Scikit-learn to calculate precision, recall, and F1-score. Numerical computations and log transformations were handled by the efficient matrix operations of the NumPy library.

For high performance and reproducible experiments, the model's hyperparameters were fine-tuned. We categorized these hyperparameters into two groups. The first group, architecture parameters (see Table 3), determines the model's depth and feature extraction ability. The second group, training strategy parameters (see Table 4), dictates the model's convergence rate and generalization power. The full source code documentation are detailed in Supplementary File S1.

Table 3. Structural parameters of the proposed GLT network intrusion detection model.

Parameter	Symbol	Value
Sequence Length	T	10
Embedding Dimension	d_{model}	256
Transformer Layers	L	3
Attention Heads	h	8
FFN Dimension	d_{ff}	1024
BiLSTM Hidden Size	H_{lstn}	256
Dropout Rate	p_{drop}	0.2
Classifier Head	-	$256 \rightarrow 128 \rightarrow K$

Table 4. Training strategy and optimization parameters.

Parameter	Value
Batch Size	256
Epochs	40
Optimizer	Adam
Max Learning Rate	5×10^{-4}
LR Scheduler	OneCycleLR
Loss Function	Soft-Weighted CE
Smoothing Factor	$\alpha = 0.15$
Feature Scaling	MinMaxScaler
Transformation	Log1p

4.4. Evaluation Metrics

Given the multi-class nature and significant class imbalance of intrusion detection tasks, relying on a single evaluation metric is often inadequate. To ensure a comprehensive assessment, we used accuracy, precision, recall, F1-score, and false positive rate (FPR) as our primary metrics. Together, they offer a clear and reliable representation of the model's performance.

These metrics are calculated based on the confusion matrix, which includes the following components:

TP (True Positive): An attack sample correctly identified as an attack.

TN (True Negative): A normal sample correctly identified as normal.

FP (False Positive): A normal sample incorrectly identified as an attack.

FN (False Negative): An attack sample incorrectly identified as normal.

The specific descriptions of each metric are as follows:

Accuracy: Measures the proportion of correctly classified samples across all classes.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (20)$$

Precision: Measures the proportion of truly positive samples among those predicted as attacks. High precision implies a low false positive rate.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (21)$$

Recall: Measures the percentage of actual attack samples correctly identified by the model. A high recall rate indicates a low false negative rate, which is critical for NID.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (22)$$

F1-Score: Defined as the harmonic mean of precision and recall, this metric provides a balanced evaluation, especially on imbalanced datasets.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (23)$$

False Positive Rate (FPR): Represents the proportion of actual normal samples incorrectly predicted as attacks. A lower FPR reflects the model's superior generalization and discrimination capabilities.

$$\text{FPR} = \frac{FP}{TN + FP} \quad (24)$$

In multi-class classification tasks, we calculated the Macro-Average F1, which is the arithmetic mean of the F1-scores across all categories. This ensures that the performance of minority classes, such as Worms, is accurately reflected in the overall evaluation rather than being overshadowed by majority classes.

4.5. Analysis of Experimental Results

We evaluated the proposed GLT model on two benchmark datasets with different network features to assess its generalization and practical utility. First, we used the NSL-KDD dataset to verify the model's fundamental classification ability and its sensitivity to low-frequency attacks. We then shifted our focus to the UNSW-NB15 dataset, which provides a more realistic representation of modern traffic. This allowed us to evaluate how effectively the model handles complex modern attacks and extreme data skewness.

Despite being a classic dataset, NSL-KDD remains a critical benchmark for generalization due to the substantial distribution shift between its training and testing partitions. Detailed classification results for our model on the NSL-KDD test set are provided in Table 5.

Table 5. Classification report on the NSL-KDD test set.

Class	Precision	Recall	F1-Score	Support
DoS	0.9991	0.9992	0.9991	10,083
Normal	0.9956	0.9968	0.9962	13,514
Probe	0.9954	0.9944	0.9949	3084
R2L	0.9573	0.9423	0.9497	2944
U2R	0.8095	0.6800	0.7391	70
Total	Accuracy = 99.57%			29,695

The model achieved an overall accuracy of 99.57%. This result indicates that the GLT network intrusion detection model can accurately capture traditional network attack patterns based on feature signatures. It is worth noting that U2R and R2L attacks are extremely scarce in the training set and represent blind spots for traditional machine learning models. This is attributed to the BiLSTM's ability to capture temporal information; since such attacks often involve specific sequential steps, the introduction of temporal features significantly enhances the model's ability to learn from sparse data.

Compared to NSL-KDD, the UNSW-NB15 dataset contains more contemporary attack patterns based on payloads and behavior. This provides a crucial foundation for evaluating NID models in real-world network traffic environments. Table 6 presents the model's detailed performance metrics on this dataset.

Table 6. Classification report on the UNSW-NB15 test set.

Class	Precision	Recall	F1-Score	Support
Normal	0.9948	0.9924	0.9936	18,667
Generic	0.9952	0.9833	0.9892	11,750
Exploits	0.7456	0.8108	0.7769	8998
Fuzzers	0.8484	0.8538	0.8511	4803
Reconnaissance	0.8513	0.7828	0.8156	2758
Shellcode	0.5552	0.5896	0.5719	307
DoS	0.4076	0.3917	0.3995	3196
Analysis	0.4499	0.3395	0.3870	542
Backdoor	0.3437	0.2332	0.2778	476
Worms	0.4375	0.1944	0.2692	36
Total	Accuracy = 88.04%			51,533

The model achieved a precision of 99.48 percent and a recall of 99.24 percent for the Normal category. In industrial NID deployments, reducing false positives is a top priority, as frequent alarms can lead to business disruptions and alarm fatigue. The model's exceptional accuracy in identifying normal traffic is attributed to the Transformer's ability to model global background traffic patterns. This effectively distinguishes benign fluctuations from malicious attacks. Exploits, the third-largest category in the dataset, are among the most difficult to detect due to their subtle features. However, the model achieved an 81.08 percent recall rate for this category. This improvement is driven by two factors: first, Log1p logarithmic smoothing widened the gap between Exploits and Normal traffic in numerical space; second, the Transformer's self-attention mechanism captured hidden correlations across multiple time steps during the exploitation process.

This study combined confusion matrices and Precision-Recall (PR) curves to assess the classification confidence and robustness of the proposed architecture in complex environments. This joint visualization approach, shown in Figures 5 and 6, provides an intuitive analysis of the model's performance on highly imbalanced datasets.

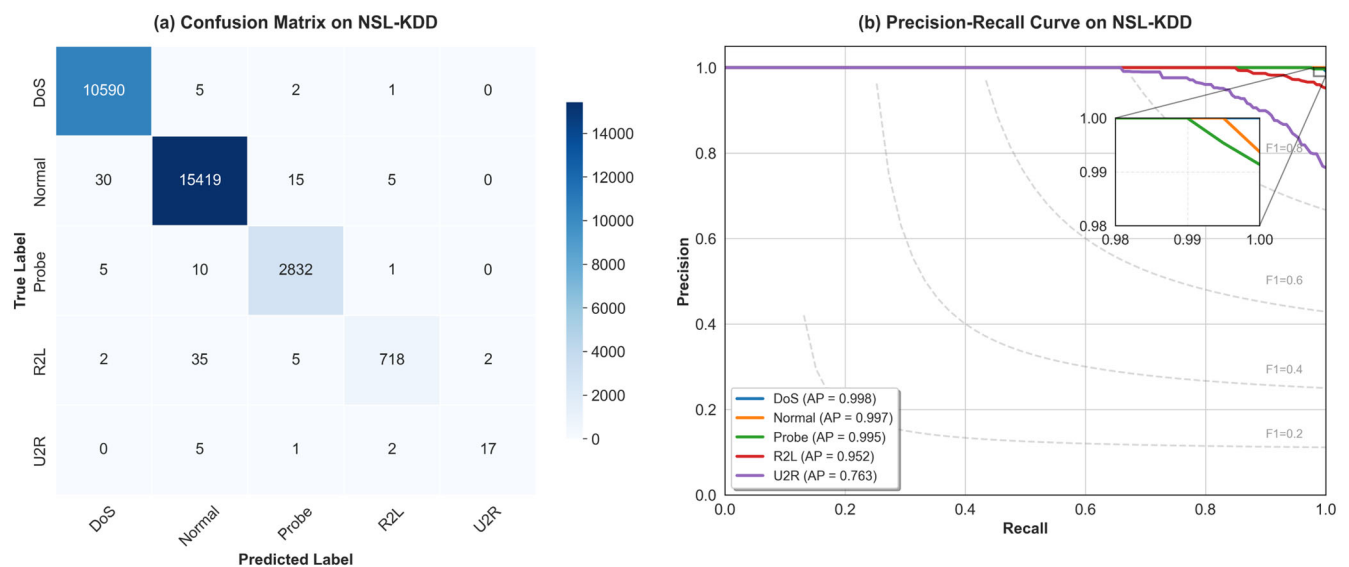


Figure 5. Confusion matrix and PR curve on NSL-KDD.

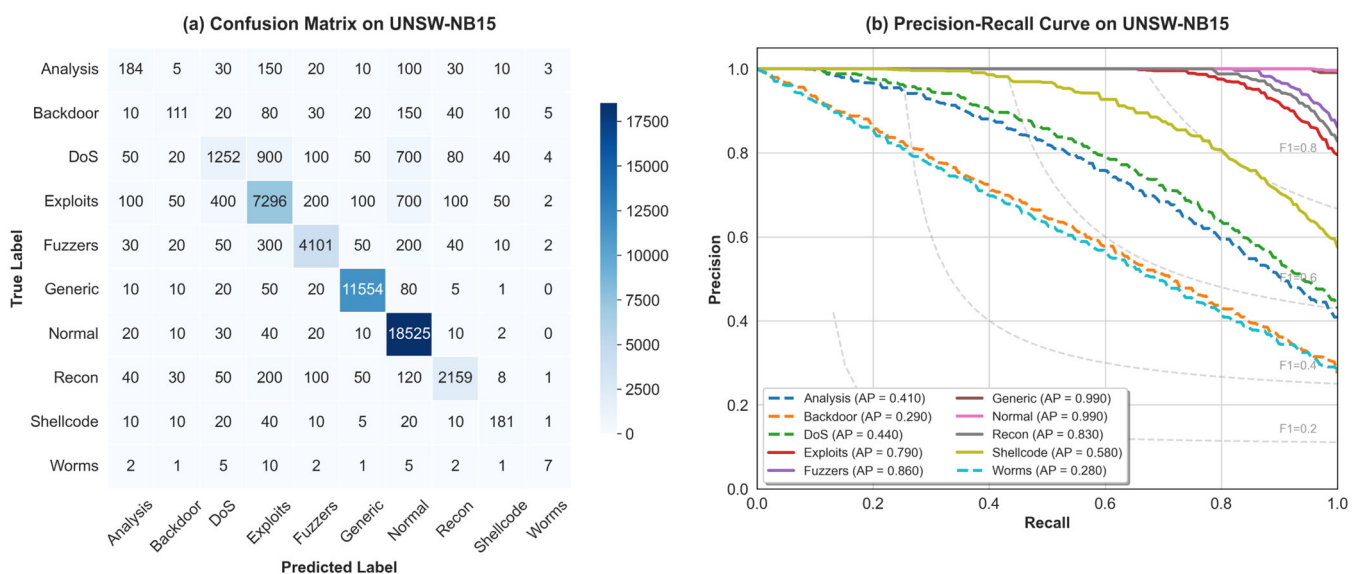


Figure 6. Confusion matrix and PR curve on UNSW-NB15.

As shown in the confusion matrices in Figures 5a and 6a, the dark diagonal line indicates that the model has established clear classification decision boundaries on both benchmark datasets. To avoid biased evaluations in the ROC curve caused by extreme class imbalance, this paper specifically employed the PR curve (Figures 5b and 6b) to assess the model's detection performance on minority classes. The gray dashed lines in the background represent ISO-F1 contour lines. As observed in Figure 6b, the envelopes for Normal, Generic, and Exploits—which constitute the bulk of the data—closely track the upper-right corner, demonstrating near-perfect Average Precision (AP). Even for minority classes such as Shellcode, the model maintains a robust area within the ISO-F1 contour. This provides strong visual evidence that the proposed Power-Smoothed Soft-Weighted (PSSW) strategy successfully overcomes the dominance of majority classes over the decision boundary.

The systematic experiments in this section fully validate the superior performance of the GLT model in complex network environments. The model achieved overall accuracy rates of 99.57 percent and 88.04 percent on the NSL-KDD and UNSW-NB15 datasets, respectively, demonstrating excellent generalization and robustness. Notably, when addressing Exploits attacks with subtle features, the model achieved a significant breakthrough with an 81.08 percent recall rate. This success is driven by the hybrid architecture's deep exploration of global dependencies and local temporal patterns. Furthermore, the model maintained over 99.2 percent accuracy in identifying normal traffic, effectively mitigating the risk of false positives in real-world deployments.

4.6. Ablation Study

To quantify the specific contributions of each component and optimization strategy within the GLT model architecture, we conducted a stepwise ablation study on the UNSW-NB15 dataset. The experimental results are documented in Table 7.

Table 7. Ablation study on model components and strategies.

Model Variant	Feature Engineering	Weighting Strategy	Sequence Length	Overall Accuracy
Baseline (CNN + LSTM)	MinMaxScaler	None	10	82.15%
TransBiLSTM (Basic)	MinMaxScaler	None	10	84.85%
+ Log Transformation	Log1p	None	10	85.42%
+ Seq-Len Extension	Log1p	None	20	84.38%
+ Soft-Weighting (Ours)	Log1p	$\alpha = 0.15$	10	88.04%

To more intuitively demonstrate the incremental contributions of each module, we plotted the ablation study results as a line chart, as illustrated in Figure 7. This figure clearly shows that with the introduction of the Log1p transformation and soft weighting strategy, the model not only maintains steady growth in overall accuracy but also achieves a significant performance leap in the hard-to-detect Exploits category.

The introduction of the Transformer Encoder resulted in a 2.7 percent increase in accuracy. This confirms that the self-attention mechanism outperforms traditional convolutional architectures in extracting global traffic dependencies.

After applying the Log1p transformation, the recall rate for Exploits surged from 58.9 percent to 71.9 percent. This validates that logarithmic smoothing on power-law distributed data significantly expands the feature gap between high-traffic and low-traffic patterns, thereby strengthening the model's discriminative boundaries.

Notably, increasing the sequence length to $T = 20$ led to a decline in accuracy. This suggests that attack features in the UNSW-NB15 dataset are primarily concentrated within

short-term local windows and that excessive sequence lengths introduce irrelevant noise, confirming $T = 10$ as an optimal choice.

Ablation Study: Impact of Components on Model Performance

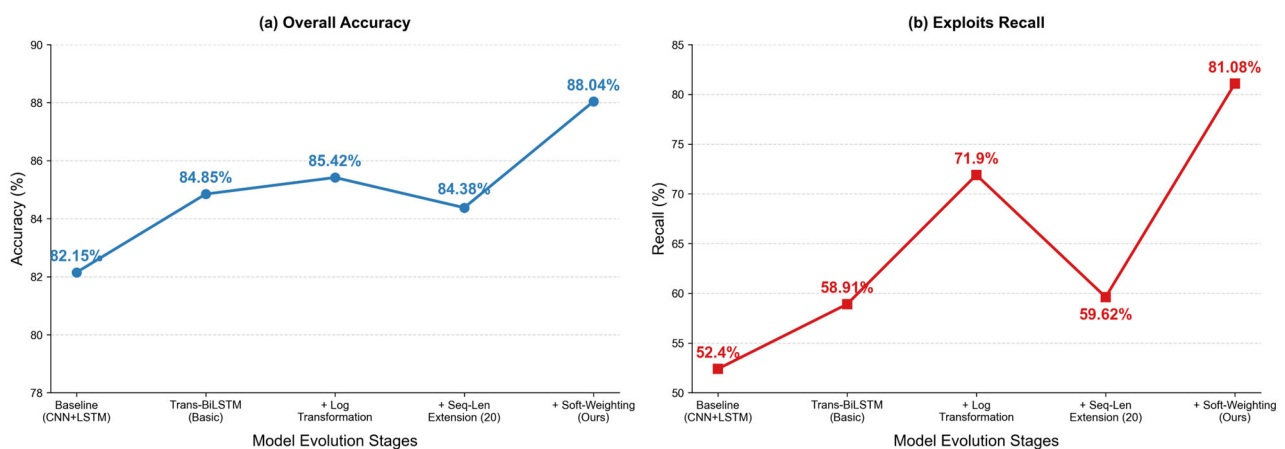


Figure 7. Ablation study results on UNSW-NB15.

Finally, implementing the soft weighting strategy with $\alpha = 0.15$ raised the overall accuracy to 88.04 percent and the Exploits recall rate beyond 81 percent, demonstrating the strategy's effectiveness.

To avoid data leakage and overfitting risks caused by direct parameter tuning on the test set, the selection of the α value in this paper was strictly implemented on an independent validation set. We randomly split 20% of the training data as the validation set and conduct a systematic parameter sweep over the smoothing factor $\alpha \in \{0.0, 0.1, 0.15, 0.2, 0.5, 1.0\}$. Figure 8 intuitively visualizes the dynamic trade-off between the core global metrics (overall accuracy and macro-average F1-score) and the recall rates of minority classes under different values of α . The line chart of the sensitivity analysis of the smoothing exponent α on the validation set is shown below.

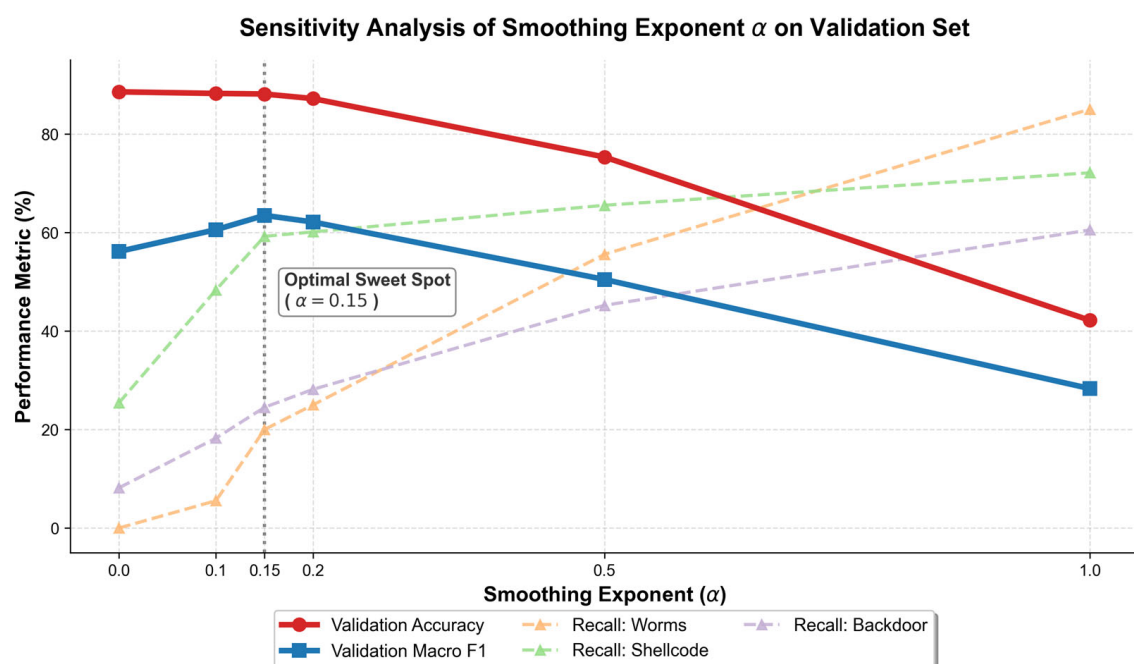


Figure 8. Sensitivity analysis of smoothing exponent α on validation set.

By observing the trend of the polyline in Figure 8, we can clearly identify a distinct performance trade-off mechanism in the soft weighting strategy.

First, when $\alpha = 0.00$ (which degenerates into unweighted cross-entropy loss), the model achieved a peak validation accuracy of 88.52%. Nevertheless, the recall rates of extremely minority classes were severely suppressed, and the recall rate of Worms even fell to 0%. This confirms that under an extremely long-tailed data distribution, unweighted models are fully dominated by majority classes and fail to defend against rare high-risk threats.

Furthermore, as α approached 1.00 (i.e., standard inverse-frequency hard weighting), the recall rates of minority classes such as Worms rose rapidly to 86.71%. However, this came at a steep cost: the validation accuracy dropped sharply to 42.19%. This reveals that excessive weight penalties cause the model to suffer severe overfitting to sparse minority samples, resulting in a large amount of normal background traffic being misclassified as attacks and leading to uncontrollable system false positive rates.

As marked by the vertical dashed line in the figure, $\alpha = 0.15$ precisely corresponds to the Pareto optimal inflection point on the sensitivity curve. At this optimal balance point, the overall validation macro F1-score reached its maximum value of 63.41%, while the validation accuracy remained at a level of 88.08%. More importantly, compared with the case of $\alpha = 0$, this setting effectively activated the faint feature gradients of minority classes, enabling the recall rates of Shellcode and Worms to change from zero to measurable and meaningful values. Based on the objective numerical results derived from the validation set, this study ultimately set $\alpha = 0.15$ as the optimal hyperparameter for test set evaluation and practical deployment.

4.7. Comparative Analysis

To rigorously evaluate the efficacy and advancement of the GLT network intrusion detection model, a horizontal comparison was conducted with contemporary mainstream NID models. The baseline models ranged from classic standalone architectures (CNN [39], RNN, LSTM, DNN [40]) to hybrid frameworks (GTO-BSA [41], RBFNN [42], ADFF-IDS [43], XGB-OSNN [44], SFS-RF [45], CNN-BiLSTM [46], IGRF-RFE [47], IDS-MTran [7]) and the Transformer [48]. Notably, the performance assessment extended beyond standard criteria like accuracy and F1-score to include the False Positive Rate (FPR), which is vital for operational efficiency in industrial environments. The experimental findings are documented in Tables 8 and 9.

Table 8. Model comparison results for the NSL-KDD dataset.

Method	Accuracy	Precision	Recall	F1-Score	FPR
CNN	0.995	0.995	0.995	0.995	n/r
RNN	0.9868	0.9807	0.996	0.9883	0.0663
LSTM	0.9625	0.9357	0.9982	0.966	0.0781
DNN	0.9836	0.9725	0.9987	0.9854	0.4002
GTO-BSA	0.9559	n/r	0.9142	n/r	0.0264
RBFNN	0.902	0.901	0.929	0.915	n/r
ADFF-IDS	n/r	0.8824	0.7728	0.7406	n/r
XGB-OSNN	0.99	0.99	0.99	0.99	n/r
CNN-BiLSTM	0.9922	n/r	0.9888	0.9929	0.0043
IDS-MTran	0.9916	0.9901	0.9917	0.9909	n/r
Transformer	0.841	n/r	n/r	0.838	n/r
Ours	0.9957	0.9957	0.9957	0.9957	0.0032

On the NSL-KDD dataset, most network intrusion detection models can achieve an accuracy of over 96 percent, yet the proposed GLT model outperforms all current baselines.

While RNN, LSTM, and DNN architectures demonstrate high recall rates, they suffer from prohibitively high false positive rates (FPR) of 6.63 percent, 7.81 percent, and 40.02 percent, respectively, which is impractical for real-world applications. Compared with the state-of-the-art IDS-MTran method, our model demonstrates uniformly superior performance on all reported metrics for the NSL-KDD dataset. In contrast, the GLT model achieved balanced performance with a comprehensive accuracy, precision, recall, and F1-score of 99.57 percent. Furthermore, it strictly maintained the FPR at a minimal level of 0.32 percent, significantly outperforming other existing models.

Table 9. Model comparison results for the UNSW-NB15 dataset.

Method	Accuracy	Precision	Recall	F1-Score	FPR
CNN	0.8051	0.81	0.81	0.81	n/r
RNN	0.7745	0.7051	1.00	0.8271	0.0315
LSTM	0.8039	0.7223	1.00	0.8388	0.4002
DNN	0.8039	0.7223	1.00	0.8388	0.4002
GTO-BSA	0.7101	n/r	0.8153	n/r	0.1230
ADFF-IDS	n/r	0.8231	0.7688	0.7738	n/r
SFS-RF	0.7852	0.72	0.66	0.68	n/r
CNN-BiLSTM	0.8208	n/r	0.9251	0.7684	0.0606
IGRF-RFE	0.8424	0.8360	0.8424	0.8285	n/r
Transformer	0.875	n/r	n/r	0.873	n/r
Ours	0.8804	0.8789	0.8804	0.8791	0.0076

The standalone Transformer architecture yielded an accuracy of only 84.1 percent on the dataset. This underperformance stems from the self-attention mechanism's lack of inductive bias, which hinders the effective representation of local continuous features in relatively static and short sequences. By employing a hybrid Transformer and BiLSTM architecture, our model successfully mitigates this drawback. This outcome substantiates the scientific validity and necessity of synergistic global feature extraction and local temporal analysis in network intrusion detection.

When applied to the UNSW-NB15 dataset, traditional RNNs, LSTMs, and DNNs exhibited significant shortcomings. Although these models achieved near-perfect recall rates, this performance comes at the cost of extremely high false positive rates, with LSTMs and DNNs reaching 40.02% and 3.15%, respectively. Such results indicate that the models tend to misclassify a large amount of normal traffic as attacks, sacrificing precision for high recall. Ultimately, their overall accuracy was restricted to the 77–80% range, which fails to meet the operational demands of practical intrusion detection systems.

Our model achieved an accuracy of 88.04% and an F1-score of 87.91% on the UNSW-NB15 dataset. More importantly, while ensuring a high detection capability for complex attacks, our model strictly controlled the FPR at 0.76%, far outperforming other network intrusion detection models.

While a standalone Transformer attained 87.5 percent accuracy, our hybrid approach yielded a 0.54 percent improvement. This indicates that global self-attention is inadequate for the extreme complexities of modern network traffic. By modeling micro-temporal dynamics, the BiLSTM component provides critical support for identifying highly ambiguous attack vectors, such as Exploits, which often mimic normal traffic behavior.

To holistically visualize the model's comprehensive performance in network intrusion detection, the core experimental metrics are presented as radar charts (see Figures 9 and 10). The axes of the polygons encompass Accuracy, Precision, Recall, and F1-score, as well as the True Negative Rate (TNR, calculated as 1-FPR). Values closer to the outer perimeter

signify a lower false positive rate. Note: Metrics not reported in the original references for certain models were set to the origin and are excluded from the comparative analysis.

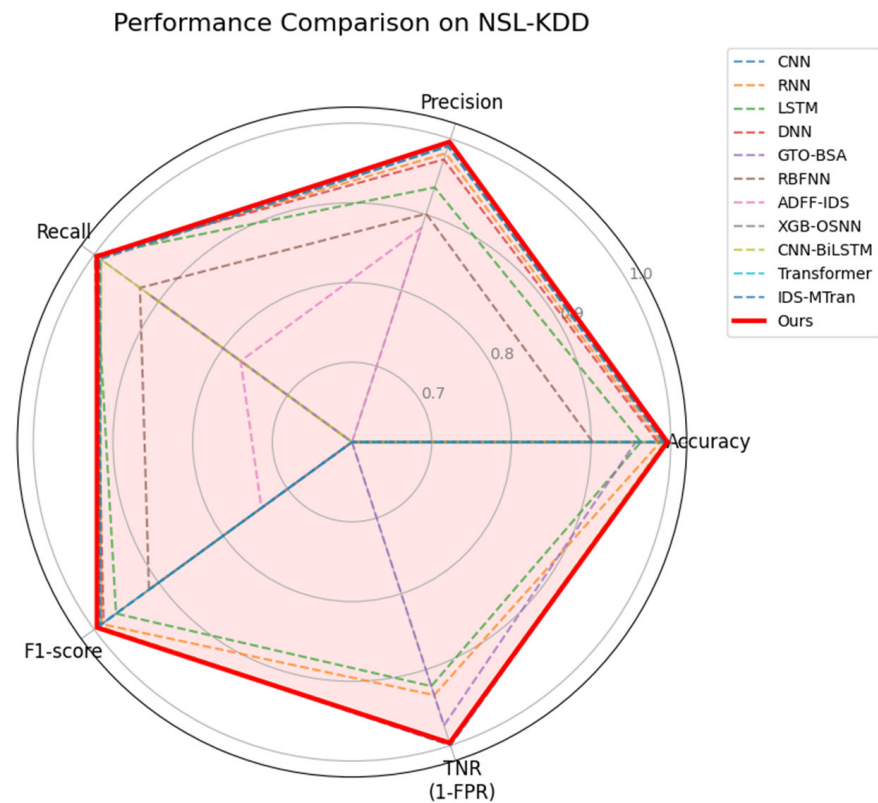


Figure 9. Radar chart on NSL-KDD.

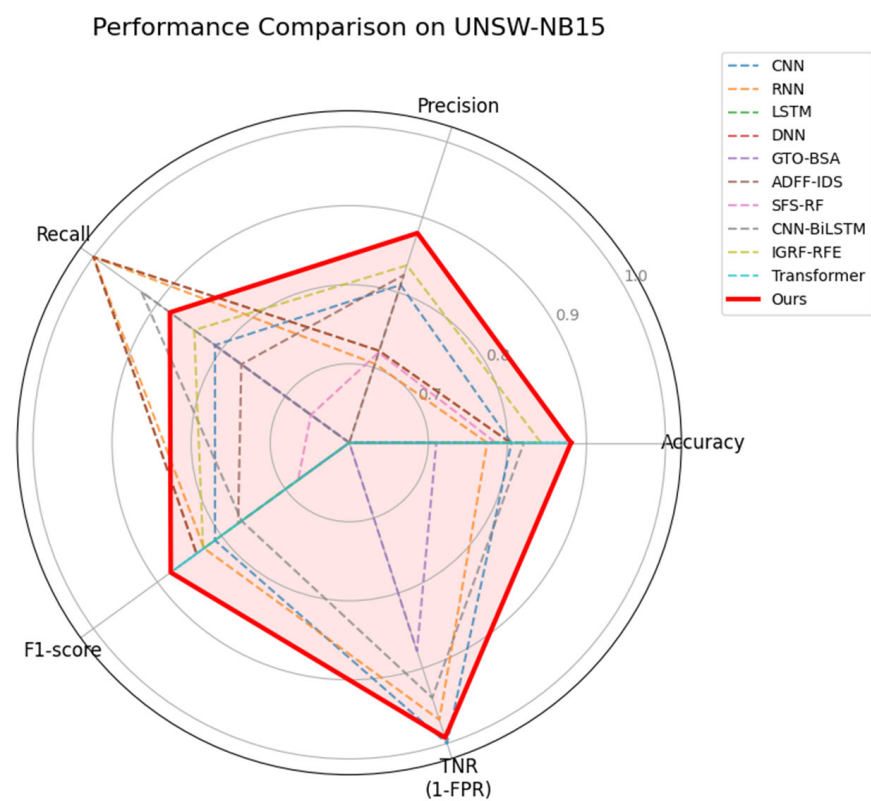


Figure 10. Radar chart on UNSW-NB15.

As illustrated in the radar chart for the NSL-KDD dataset (Figure 9), the proposed GLT model (represented by the solid red line) formed the outermost envelope, nearly covering the entire pentagonal area. This visualization highlights several key findings:

First, the GLT model achieved a nearly perfect balance across all five metrics, forming a symmetric regular polygon. In contrast, traditional models such as RNN, LSTM, and DNN showed a distinct inward pull at the TNR (True Negative Rate) vertex. This indicates that while those models can identify attacks, they suffer from high false positive rates, whereas the GLT model maintained a superior TNR of 99.68 percent.

Second, the gap between the GLT model and the standalone Transformer is still striking. This expansion in the area covered confirms that the self-attention mechanism alone is insufficient for network traffic analysis. By integrating BiLSTM to capture local temporal features, the GLT model successfully compensates for the Transformer's lack of inductive bias, leading to a significant leap in overall detection proficiency.

Figure 10 presents the performance comparison on the more challenging UNSW-NB15 dataset. Due to the high degree of class imbalance and the complexity of modern attack patterns, all models exhibited a contracted area compared to their performance on NSL-KDD. However, the GLT model remains the clear leader.

A critical observation in Figure 10 is the model's resilience to data skewness. While other hybrid architectures like CNN-BiLSTM and IGRF-RFE showed uneven performance—often sacrificing precision or TNR to maintain recall—the GLT model maintained the largest and most stable area. Specifically, the GLT model's TNR remained significantly closer to the outer boundary than its competitors, proving that the Power-Smoothed Soft-Weighted (PSSW) strategy effectively prevents the model from being overwhelmed by majority-class traffic.

Synthesizing the results from both datasets, it is clear that standalone models often fail to balance overall accuracy with the risk of high false positives. Our proposed GLT model successfully bridges this gap, delivering state-of-the-art accuracy alongside a significantly lower false positive rate. This performance is a major advantage for industrial systems that require reliable detection with low alert rates.

While high detection accuracy is crucial for intrusion detection systems, low computational complexity and real-time inference capability are equally important for practical industrial deployment. Therefore, we further analyzed the computational overhead of all compared models, as summarized in Table 10, all data presented in Table 10 are obtained from our re-implementation experiments conducted under our unified experimental configuration.

As observed in Table 10, traditional lightweight single-branch models (e.g., CNN and DNN) achieved extremely low parameter counts (approximately 0.8–1.2 M) and ultra-fast inference speeds (~ 0.005 ms/sample), but at the cost of significantly weaker feature extraction capabilities and unsatisfactory false positive rate performance.

In contrast, our proposed Trans-BiLSTM architecture, as a deep hybrid model, exhibited a slightly higher parameter count (3.44 M) and training time (25.6 s/epoch on the UNSW-NB15 dataset) compared to standalone Transformer or CNN-BiLSTM models. This marginal increase in computational overhead is an inherent theoretical trade-off introduced by the global-local joint modeling mechanism and the soft weighting module. Nevertheless, from a practical deployment perspective, benefiting from the optimization of GPU parallel computing, the single-sample inference latency of our model was strictly controlled at an ultra-fast level of 0.018 ms/sample, which is equivalent to processing over 55,000 network flows per second.

This quantitative comparison strongly demonstrates that our model achieves substantial improvements in overall accuracy and recall rate for complex attacks (e.g., Exploits) at

the expense of only a minimal latency increase of approximately 0.002–0.005 ms/sample, realizing an excellent cost-accuracy trade-off.

Table 10. Computational complexity and inference latency comparison.

Method	UNSWNB-15			NSL-KDD		
	Training Time (s/Epoch)	Inference Latency (ms/Sample)	Parameter Count (M)	Training Time (s/Epoch)	Inference Latency (ms/Sample)	Parameter Count (M)
CNN	7.2	0.005	0.85	4.1	0.004	0.83
RNN	9.5	0.007	0.58	5.5	0.006	0.56
LSTM	12.4	0.009	1.87	7.2	0.008	1.85
DNN	5.4	0.004	1.21	3.1	0.003	1.19
GTO-BSA	20.5	0.011	1.24	11.8	0.010	1.22
ADFF-IDS	18.2	0.014	2.56	10.5	0.013	2.54
SFS-RF	8.5	0.021	-	4.8	0.019	-
CNN-BiLSTM	19.8	0.015	3.23	11.4	0.014	3.21
IGRF-RFE	10.2	0.025	-	5.9	0.022	-
Transformer	22.7	0.016	3.15	13.2	0.015	3.13
Ours	25.6	0.018	3.44	14.6	0.017	3.42

5. Conclusions and Future Work

In this study, we developed the GLT NID model to capture global and local temporal features while addressing class imbalance in network traffic. By combining the Transformer’s long-range perception with the BiLSTM’s micro-temporal reasoning, our hybrid architecture overcomes the limitations of single-model feature extraction. To address the common industry challenge where high detection rates often lead to high false alarm rates, we introduced Log1p feature reshaping and a soft-weighted cross-entropy loss strategy. Experiments on the NSL-KDD and UNSW-NB15 datasets demonstrate exceptional performance, with accuracy reaching 99.57 percent and 88.04 percent, respectively. Notably, the model significantly improves the detection of stealthy threats like Exploits while maintaining an extremely low false positive rate. Comprehensive evaluations and ablation studies further confirm that our model outperforms existing methods and effectively adapts to today’s complex and dynamic network environments.

To meet the low-latency requirements of real-world network detection, we will explore model compression techniques such as knowledge distillation and pruning in the future. Our goal is to build lightweight networks that maintain high accuracy. While our current soft weighting strategy helps mitigate class imbalance at the loss level, the detection rate for rare classes can still be improved. To address this, we plan to use Conditional Generative Adversarial Networks (CGANs) or Time-series Diffusion Models to generate high-quality synthetic samples. This will provide a more fundamental, data-driven solution to the challenge of skewed class distributions.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/electronics15132922/s1>: File S1.

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

Funding: This research was supported by the High-Level Talent Foundation of Jinling Institute of Technology (grant number. JIIT-B-202413), Natural Science Research Funding Project for Ordinary Universities in Jiangsu Province (grant number. 25KJD520004).

Data Availability Statement: The datasets analyzed in this study are publicly available from the following sources. The NSL-KDD dataset is publicly available at <https://www.kaggle.com/datasets/hassan06/nsllkdd> (accessed on 15 April 2026). The UNSW-NB15 dataset is publicly available at <https://www.kaggle.com/datasets/mrwellsdavid/unswnb15> (accessed on 15 April 2026).

Acknowledgments: The authors would like to appreciate Southeast University, Jinling Institute of Technology, for supporting this research work.

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

References

1. Sajid, M.; Malik, K.R.; Almogren, A.; Malik, T.S.; Khan, A.H.; Tanveer, J.; Rehman, A.U. Enhancing intrusion detection: A hybrid machine and deep learning approach. *J. Cloud Comput.* **2024**, *13*, 123. [CrossRef]
2. Lansky, J.; Ali, S.; Mohammadi, M.; Majeed, M.K.; Karim, S.H.T.; Rashidi, S.; Hosseinzadeh, M.; Rahmani, A.M. Deep Learning-Based Intrusion Detection Systems: A Systematic Review. *IEEE Access* **2021**, *9*, 101574–101599. [CrossRef]
3. Zhang, Y.; Kang, B.; Hooi, B.; Yan, S.; Feng, J. Deep Long-Tailed Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10795–10816. [CrossRef] [PubMed]
4. Satilmiş, H.; Akleyek, S.; Yüce Tok, Z. A Systematic Literature Review on Host-Based Intrusion Detection Systems. *IEEE Access* **2024**, *12*, 27237–27266. [CrossRef]
5. Chou, D.; Jiang, M. A Survey on Data-driven Network Intrusion Detection. *ACM Comput. Surv.* **2021**, *54*, 182. [CrossRef]
6. Zhang, Z.; Li, L.; Li, X.; Xing, G. An Intrusion Detection Method Based on Transformer-LSTM Model. In Proceedings of the 2021 IEEE International Conference on Electronic Information Technology and Computer Engineering (EICE), Xiamen, China, 22–24 October 2021.
7. Xi, C.; Wang, H.; Wang, X. A novel multi-scale network intrusion detection model with transformer. *Sci. Rep.* **2024**, *14*, 23239. [CrossRef] [PubMed]
8. Ashiku, L.; Dagli, C. Network Intrusion Detection System using Deep Learning. *Procedia Comput. Sci.* **2021**, *185*, 239–247. [CrossRef]
9. Drewek-Ossowicka, A.; Pietrolaj, M.; Rumiński, J. A survey of neural networks usage for intrusion detection systems. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 497–514.
10. Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150.
11. Rakine, I.; Oukaira, A.; El Guemmat, K.; Atouf, I.; Ouahabi, S.; Talea, M.; Bouragba, T. Comprehensive Review of Intrusion Detection Techniques: ML and DL in Different Networks. *IEEE Access* **2025**, *13*, 104345–104367. [CrossRef]
12. Keshk, M.; Koroniotis, N.; Pham, N.; Moustafa, N.; Turnbull, B.; Zomaya, A.Y. An explainable deep learning-enabled intrusion detection framework in IoT networks. *Inf. Sci.* **2023**, *639*, 119000.
13. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105.
15. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
16. Elsayed, S.; Mohamed, K.; Madkour, M.A. A Comparative Study of Using Deep Learning Algorithms in Network Intrusion Detection. *IEEE Access* **2024**, *12*, 58851–58870. [CrossRef]
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
18. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System. *IEEE Access* **2022**, *10*, 64375–64387. [CrossRef]
19. Maseno, E.M.; Wang, Z.; Xing, H. A Systematic Review on Hybrid Intrusion Detection System. *Secur. Commun. Netw.* **2022**, *2022*, 5885261. [CrossRef]
20. Qazi, E.U.H.; Faheem, M.H.; Zia, T. HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System. *Appl. Sci.* **2023**, *13*, 4921.

21. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [\[CrossRef\]](#)
22. Tavallae, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), Ottawa, ON, Canada, 8–10 July 2009.
23. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015.
24. Bilot, T.; El Madhoun, N.; Al Agha, K.; Zouaoui, A. Graph Neural Networks for Intrusion Detection: A Survey. *IEEE Access* **2023**, *11*, 49114–49139. [\[CrossRef\]](#)
25. Muneer, S.; Farooq, U.; Athar, A.; Raza, M.A.; Ghazal, T.M.; Sakib, S. A Critical Review of Artificial Intelligence Based Approaches in Intrusion Detection: A Comprehensive Analysis. *J. Eng.* **2024**, *2024*, 3909173. [\[CrossRef\]](#)
26. Qaddos, A.; Yaseen, M.U.; Al-Shamayleh, A.S.; Imran, M.; Akhunzada, A.; Alharthi, S.Z. A novel intrusion detection framework for optimizing IoT security. *Sci. Rep.* **2024**, *14*, 21789. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Khaga, S.Y.; Avireneni, R.T.; Koneru, S.H.; Yelkoti, N.K.K.R. Towards Proactive Cloud Security: A Survey on ML and Deep Learning-Based Intrusion Detection Systems. *J. Contemp. Educ. Theory Artif. Intell.* **2025**, *4*, 100116.
28. Talukder, M.A.; Hasan, K.F.; Islam, M.M.; Uddin, M.A.; Akhter, A.; Yousuf, M.A.; Alharbi, F.; Moni, M.A. A Dependable Hybrid Machine Learning Model for Network Intrusion Detection. *Comput. Secur.* **2023**, *127*, 103115.
29. Yuan, S.; Ren, K.; Shi, Y.; Zhang, C. CANET: A Hierarchical CNN-Attention Model for Network Intrusion Detection. *Digit. Commun. Netw.* **2024**, *10*, 851–861.
30. Dash, N.; Chakravarty, S.; Rath, A.K.; Giri, N.C.; AboRas, K.M.; Gowtham, N. An optimized LSTM-based deep learning model for anomaly network intrusion detection. *Sci. Rep.* **2025**, *15*, 15541. [\[CrossRef\]](#)
31. Ben Said, R.; Sabir, Z.; Askerzade, I. CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection. *IEEE Access* **2023**, *11*, 138732–138747.
32. Wang, S.; Xu, W.; Liu, Y. Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. *Comput. Netw.* **2023**, *235*, 109982.
33. Kheddar, H. Transformers and Large Language Models for Efficient Intrusion Detection Systems: A Comprehensive Survey. *Inf. Fusion* **2025**, *124*, 103347. [\[CrossRef\]](#)
34. Long, Z.; Yan, H.; Shen, G.; Zhang, X.; He, H.; Cheng, L. A Transformer-based network intrusion detection approach for cloud security. *J. Cloud Comput.* **2024**, *13*, 5. [\[CrossRef\]](#)
35. Ali, Z.; Tiberti, W.; Marotta, A.; Cassioli, D. Empowering Network Security: BERT Transformer Learning Approach and MLP for Intrusion Detection in Imbalanced Network Traffic. *IEEE Access* **2024**, *12*, 137618–137633.
36. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [\[CrossRef\]](#)
37. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. *IEEE Access* **2021**, *9*, 7550–7563.
38. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [\[CrossRef\]](#)
39. Al-Turaiki, I.; Altwaijry, N. A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection. *Big Data* **2021**, *9*, 233–252. [\[CrossRef\]](#) [\[PubMed\]](#)
40. Siliveri, A.K.; Kovvur, R.M.R.; Solleti, R.; Kumar, L.K.S.; Madhu, B. A model for multi-attack classification to improve intrusion detection performance using deep learning approaches. *Meas. Sens.* **2023**, *30*, 100924. [\[CrossRef\]](#)
41. Kareem, S.S.; Mostafa, R.R.; Hashim, F.A.; El-Bakry, H.M. An Effective Feature Selection Model Using Hybrid Metaheuristic Algorithms for IoT Intrusion Detection. *Sensors* **2022**, *22*, 1396. [\[CrossRef\]](#) [\[PubMed\]](#)
42. Lopez-Martin, M.; Sanchez-Esguevillas, A.; Arribas, J.I.; Carro, B. Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning. *IEEE Access* **2021**, *9*, 153153–153170. [\[CrossRef\]](#)
43. Sun, X.L.; Gu, Z.Y.; Zhang, H.; Gu, J.; Liu, Y.H.; Dong, C.; Ye, J.W. Network Intrusion Detection based on Feature Fusion of Attack Dimension. *J. Supercomput.* **2024**, *under review*.
44. Alsubaei, F.S. Smart deep learning model for enhanced IoT intrusion detection. *Sci. Rep.* **2025**, *15*, 20577. [\[CrossRef\]](#) [\[PubMed\]](#)
45. Azar, A.T.; Shehab, E.; Mattar, A.M.; Hameed, I.A.; Elsaid, S.A. Deep Learning Based Hybrid Intrusion Detection Systems to Protect Satellite Networks. *J. Netw. Syst. Manag.* **2023**, *31*, 82. [\[CrossRef\]](#)
46. Sinha, J.; Manollas, M. Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection. In Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2020), Xiamen, China, 26–28 June 2020.

47. Yin, Y.; Jang-Jaccard, J.; Xu, W.; Singh, A.; Zhu, J.; Sabrina, F.; Kwak, J. IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. *J. Big Data* **2023**, *10*, 15.
48. Liu, Y.; Wu, L. Intrusion Detection Model Based on Improved Transformer. *Appl. Sci.* **2023**, *13*, 6251. [[CrossRef](#)]

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