Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084490

ARTICLE

A Compact Hybrid TCN-BiGRU-TinyTransformer Framework for Fine-Grained Multiclass Intrusion Detection

Ye Lu[1], Haoyang Hu[1,*], Wenyi Chang[1], Wanbin Liu[1] and Wenyuan Zhang[2]

[1]School of Computer Science and Artificial Intelligence, Lanzhou University of Technology, Lanzhou, China
[2]School of Mechanical and Electrical Engineering, Lanzhou University of Technology, Lanzhou, China

*Corresponding Author: Haoyang Hu. Email: 242085412030@lut.edu.cn

Received: 23 April 2026; Accepted: 24 June 2026; Published: 13 August 2026

ABSTRACT: Fine-grained multiclass intrusion detection over flow-level traffic remains difficult, largely because class boundaries are often entangled, temporal dependence is non-negligible, and the label distribution is heavily long-tailed. In this study, a compact temporal convolutional network (TCN)-bidirectional gated recurrent unit (BiGRU)-TinyTransformer framework is developed to bring these issues into a single modeling pipeline: the TCN branch focuses on short-range anomalous patterns, the BiGRU branch captures bidirectional temporal structure, and the TinyTransformer branch complements them with broader contextual interaction learning. To reduce the bias induced by extreme imbalance, training is not driven by a single correction mechanism, but by a coordinated strategy that combines class-balanced (CB) sampling and weighting, label-distribution-aware margin (LDAM)-Focal loss, and staged warmup plus deferred re-weighting (DRW) optimization. On the NetFlow-based UNSW-NB15-v2 dataset (NF-UNSW-NB15-v2), the resulting model reaches 98.06% Accuracy, 98.05% Weighted-F1, and 80.19% Macro-F1. These results point to a practical improvement in fine-grained attack discrimination, especially for minority classes, without weakening overall detection performance.

KEYWORDS: Intrusion detection; fine-grained multiclass classification; long-tailed learning; compact hybrid architecture; temporal convolutional network

1 Introduction

As industrial network environments become increasingly interconnected and dependent on network services, their attack surfaces continue to expand. As a crucial component of cybersecurity defense, Intrusion Detection Systems (IDS) need not only to identify malicious traffic, but also to provide attack-category information that supports security operations [1,2]. Fine-grained multiclass intrusion detection is therefore better aligned with the protection needs of security-critical network environments than binary detection, because different attack types imply different threat characteristics and response procedures. A mere judgment of "normal" or "abnormal" is often insufficient for practical security analysis. However, fine-grained multiclass intrusion detection in flow-level network traffic is significantly more difficult than coarse-grained detection, as it requires simultaneous handling of complex class boundaries, temporal dependencies in traffic evolution, and severe class imbalance issues.

In this work, the term "fine-grained multiclass intrusion detection" refers to the ability to distinguish specific attack categories, such as denial-of-service (DoS) vs. Exploits rather than labeling both generically as attacks, while maintaining practically useful recognition of minority attack classes under severe class

imbalance. This objective goes beyond binary normal/abnormal detection and differs from simply achieving high aggregate accuracy while allowing tail classes to be ignored. These two aspects are related but not identical: fine-grained classification defines the prediction granularity, whereas class imbalance describes the skewed sample distribution that makes rare attack categories difficult to learn.

This setting introduces several challenges in NetFlow-based IDS. First, NetFlow records are compact flow-level summaries, so different attack categories may share overlapping statistics such as total bytes, packet counts, flow duration, and protocol flags. Such overlap makes multiclass decision boundaries ambiguous, especially for minority classes with limited training examples. Second, temporal information is observed as chronological changes in traffic summaries rather than as raw packet sequences or explicit host/session trajectories. A single flow may be weakly discriminative in isolation, whereas its surrounding traffic context can provide useful evidence about short-term changes in traffic composition. Third, long-tailed class distributions bias optimization toward high-frequency categories, so high Accuracy or Weighted-F1 may still conceal weak recognition of rare attacks. These issues motivate both stronger traffic representation and more balanced optimization for minority categories.

Deep learning has advanced traffic modeling, but no single paradigm fully covers the coupled requirements of this task. Convolutional neural networks (CNN) effectively capture local patterns [3], recurrent architectures are suitable for stepwise temporal dependencies [4,5], and Transformer architectures readily extract long-range interactions between different inputs [6]. The central challenge is how to integrate local, temporal, and global semantics within a single compact framework while reducing majority-class bias under severe imbalance. This problem is further constrained by practical deployment requirements, where computational efficiency and compactness are important considerations [7,8].

To address this setting, a compact serial hybrid framework is constructed by integrating TCN, BiGRU, and a TinyTransformer. The contribution is not the individual use of these well-known modules, but the stage-wise role assignment among them for fine-grained NetFlow intrusion detection. The TCN first serves as a local sharpening stage, extracting short-range burst patterns and abrupt variations from neighboring NetFlow records. The BiGRU then receives the locally enhanced sequence and models bidirectional temporal evolution. Finally, the TinyTransformer operates on temporally enriched hidden states to capture broader cross-position interactions. This order follows the structure of the traffic modeling problem: local traffic irregularities are extracted first, temporal evolution is then modeled on the enhanced sequence, and global attention is applied after the sequence already contains temporal context.

This serial integration also differs from a parallel hybrid design in which recurrent and attention branches process features at the same abstraction level and are fused only at the end. A parallel scheme may mix local, temporal, and global cues before their roles are clearly separated. In contrast, the proposed TCN→BiGRU→TinyTransformer path gives each component a clearer modeling responsibility: local feature sharpening, temporal dependency modeling, and global contextual interaction. The design therefore provides a more interpretable integration strategy while keeping the model compact.

The learning stage faces an additional difficulty. Under severe class imbalance, performance degradation usually appears earliest in minority attack categories. A single correction is often insufficient, whether it is applied through resampling alone or through the loss function alone. The training procedure therefore adopts a coordinated long-tail optimization strategy that combines class-balanced sampling, Class-Balanced (CB) weighting, LDAM-Focal loss, and an optimization schedule based on warmup and DRW. These mechanisms act on different imbalance-related bottlenecks, so their joint use helps restrain majority-class dominance and supports more stable optimization for minority categories. The present analysis is limited to intrusion detection itself. Downstream extensions beyond detection, such as risk scoring and the prioritization of response actions, are not evaluated here and are left for future study.

Accordingly, the substantial novelty of this work should be understood at the task-oriented framework level rather than at the level of inventing a new standalone neural operator. The framework links three aspects that are often treated separately in multiclass IDS studies: a compact serial local-temporal-global representation path, a coordinated long-tail optimization procedure, and an evaluation focus on macro-level and minority-class discrimination under chronological NetFlow splits. This positioning is important because high overall Accuracy alone can hide weak tail-class recognition; therefore, the design is centered on improving fine-grained attack discrimination while maintaining a compact inference profile.

The principal contributions of this paper are summarized as follows:

1. A compact serial hybrid TCN-BiGRU-TinyTransformer framework is developed for fine-grained multiclass intrusion detection under a flow-level long-tailed setting. Its contribution lies in the stage-wise role assignment among local feature extraction, temporal modeling, and global context interaction, rather than in the isolated use of the individual modules.
2. For the pronounced class-imbalance problem in multiclass traffic classification, a coordinated long-tail optimization strategy is introduced. More specifically, class-balanced sampling, CB weighting, LDAM-Focal loss, and a staged schedule that couples warmup with DRW are jointly employed to curb bias toward majority classes and to strengthen the practical discriminability of minority classes.
3. Extensive experiments are conducted on NF-UNSW-NB15-v2, with additional validation on NF-ToN-IoT-v2 under the same blocked evaluation protocol. The evaluation emphasizes macro-level performance, minority-class recognition, and computational efficiency, showing that the proposed framework maintains competitive detection performance while keeping a compact model footprint.

The remainder of this paper is organized as follows. Section 2 reviews related work on benchmark datasets, deep traffic modeling, and long-tailed learning for multiclass intrusion detection. Section 3 introduces the dataset characteristics, preprocessing pipeline, and sequence construction strategy. Section 4 presents the proposed TCN-BiGRU-TinyTransformer framework and the coordinated long-tail optimization strategy. Section 5 reports the experimental setup and evaluation results, including baseline comparisons, ablation studies, fine-grained diagnostic analysis, Cross-Dataset Validation on NF-ToN-IoT-v2, and efficiency assessment. Finally, Section 6 concludes the paper and outlines future work.

2 Related Work

2.1 Benchmark Datasets and Traffic Representation for Multiclass IDS

For multiclass intrusion detection systems, benchmark selection directly affects whether results are meaningfully comparable. Dataset construction determines feature definitions, traffic scenarios, and attack taxonomies; therefore, results obtained from datasets with inconsistent feature spaces should not be interpreted as direct head-to-head comparisons. KDD Cup 99, for example, contains substantial data redundancy [9] and no longer fully reflects modern network environments. UNSW-NB15 is more suitable for contemporary attack scenarios [2,10], and NF-UNSW-NB15-v2 further reformats the original traffic into a standardized NetFlow-oriented feature space [11,12], making it better aligned with the flow-level detection focus of this study.

Other studies use ToN-IoT or Edge-IIoTset [13,14], which provide useful IoT-oriented scenarios. However, evaluating models across datasets with different protocols and feature spaces can make direct performance comparison ambiguous. Consistent feature spaces are therefore necessary for fair comparison, rather than diverse scenarios alone.

2.2 Deep Models for Fine-Grained Traffic Modeling under Efficiency Constraints

Deep learning has become one of the primary methods for traffic feature representation [15]. Convolutional networks extract local spatial patterns [3], recurrent networks process stepwise temporal dependencies [4,5], and Transformer mechanisms capture global sequence context [6,7]. Fine-grained anomaly detection often benefits from combining these three perspectives. Therefore, hybrid model architectures have become a reasonable evolution direction in this field [16,17].

Recent studies have also emphasized that flow records should not be treated only as isolated tabular samples. Chen et al. [18] proposed HC-NIDS, which incorporates historical traffic context and feature-correlation modeling for IoT intrusion detection. This study is relevant because it highlights the value of contextual traffic information; however, different from identifier-based historical aggregation, the present work removes identifier-like fields such as source IP, destination IP, and FlowID, and models monitoring-point chronological NetFlow windows through a compact serial TCN-BiGRU-TinyTransformer pipeline.

Different from many existing hybrid IDS architectures that mainly combine convolutional, recurrent, and attention modules through parallel branches or late-stage feature concatenation [7,16,17], the proposed framework adopts a serial representation path tailored to chronological NetFlow windows. In this design, local feature extraction, temporal modeling, and broader contextual interaction are organized stage by stage, which reduces premature mixing of heterogeneous representations and clarifies the role of each module.

However, simply increasing the scale of the model to improve performance will introduce significant computational overhead. A real-world IDS operates under memory constraints and must process local, temporal, and global data within a compact computational budget [8]. Many studies report improved detection accuracy, yet maintaining high accuracy with a compact parameter footprint remains an ongoing challenge for practical deployment [7,8,17,19].

2.3 Long-Tailed Learning in Multiclass Intrusion Detection

Within multiclass intrusion detection, class imbalance remains a persistent obstacle. Under imbalanced learning conditions, model optimization is typically pulled toward prevalent classes because they exert disproportionate influence during parameter updates. Minority classes, by contrast, provide markedly weaker learning signals. This often produces models that attain apparently acceptable overall Accuracy while still delivering inadequate recognition of minority classes and poor macro-level performance [20,21]. To alleviate this issue, prior work has explored data-level resampling [22], class-balanced reweighting based on the effective number of samples [23], Focal Loss for hard-example emphasis [24], and margin-based adjustment such as LDAM [25]. However, the effectiveness of these methods may vary across datasets, evaluation metrics, and task settings, and stable gains on minority-class performance remain difficult to guarantee in complex multiclass traffic classification [22,26–28].

Generative augmentation has recently become another important direction for imbalanced IDS. Zhang et al. [29] proposed DID-IDS, which uses diffusion-based augmentation to generate minority-class intrusion samples before classification. In contrast, this study does not generate synthetic traffic records; instead, it preserves the original NetFlow feature space and combines balanced sampling, class-balanced weighting, LDAM-Focal loss, and staged optimization to reduce majority-class dominance.

Overall, existing studies suggest that this long-tailed multiclass IDS setting is constrained by two coupled challenges: achieving sufficiently rich local-temporal-global traffic modeling under efficiency requirements, and improving minority-class utility under long-tailed optimization.

3 Data Preprocessing

3.1 Dataset Distribution and Long-Tailed Characteristics

This study uses NF-UNSW-NB15-v2 as the primary benchmark for fine-grained multiclass intrusion detection. NF-UNSW-NB15-v2 is a NetFlow-based variant of UNSW-NB15 designed for flow-level intrusion detection. It is selected as the main benchmark because it provides standardized NetFlow-style records, supports multiclass attack-category evaluation, and exhibits a severe long-tailed distribution that directly matches the focus of this work. The dataset contains benign traffic and nine attack categories: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. The class distribution is shown in Fig. 1.

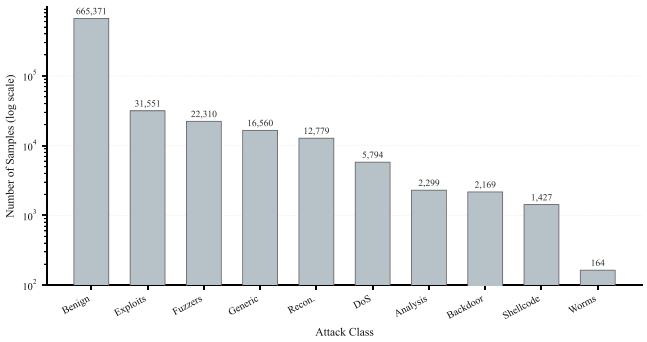


Figure 1: Class distribution of the processed NF-UNSW-NB15-v2 dataset.

NF-UNSW-NB15-v2 exhibits a pronounced long-tailed distribution. The benign class contains 665,371 samples, whereas the tail class Worms contains only 164, yielding an imbalance ratio of more than 4000:1. Other minority classes, including Analysis, Backdoor, and Shellcode, are also far smaller than head classes such as Exploits, Fuzzers, and Generic. This extreme imbalance biases model training toward majority classes, making NF-UNSW-NB15-v2 an appropriate testbed for fine-grained multiclass intrusion detection under severe class imbalance.

3.2 Feature Engineering: Field Pruning, Hybrid Encoding, and Standardization

According to the data characteristics in Section 3.1, directly using raw NetFlow records may hinder fine-grained classification because of heterogeneous feature scales, high-cardinality categorical attributes, and identifier-like fields that can induce shortcut learning. Therefore, a structured preprocessing pipeline is applied before sequence construction, as summarized in Fig. 2. Unless otherwise stated, all preprocessing statistics are estimated on the training subset of each fold and then applied to the corresponding validation and test subsets.

(1) **Field pruning and type assignment.** Identifier-like fields, including `Source IP`, `Destination IP`, `FlowID`, and `DNS_QUERY_ID`, are removed to reduce overfitting to specific hosts or sessions. Protocol-related fields, including `PROTOCOL`, `L7_PROTO`, `ICMP_TYPE`, and `TCP_FLAGS`, are treated as categorical variables to preserve protocol semantics rather than impose artificial ordinal relationships.

(2) **Numerical transformation.** Missing values in numerical features are imputed with the median, and extreme values are clipped to the 0.01–0.99 quantile range. To regularize highly skewed traffic statistics, logarithmic transformation is applied to numerical columns whose names contain keywords such as `BYTE`, `BYTES`, `PKT`, `PKTS`, `DURATION`, `THROUGHPUT`, `TCP_WIN`, `LONGEST`, and `SHORTEST`.

Negative values are clipped to zero before transformation, and the transformed value is computed as $\log(1+x)$. All numerical features are then standardized using Z-score normalization.

(3) **Hybrid encoding of categorical features.** Categorical variables are encoded using a hybrid strategy with a cardinality threshold of 64. Fields with no more than 64 unique values are one-hot encoded. In addition, the columns specified in `force_onehot_cols`, such as `DNS_QUERY_TYPE` and `FTP_COMMAND_RET_CODE`, are also one-hot encoded even when their cardinality exceeds this threshold. For remaining high-cardinality fields, logarithmic frequency encoding is applied, where each category value x is mapped to $\ln(\text{count}(x)+1)$ and $\text{count}(x)$ denotes its occurrence frequency in the fitted training data. Missing or unseen values are assigned a default frequency of 1 before transformation.

(4) **Encoded feature dimension.** After field pruning, numerical transformation, and categorical encoding, each flow record is represented as an F-dimensional feature vector. Here, F is determined by the realized input distribution after preprocessing. It is not treated as a manually fixed constant.

$$F = N_{\text{num}} + \sum_{i \in \text{OneHot}} |C_i| + N_{\text{freq}}, \tag{1}$$

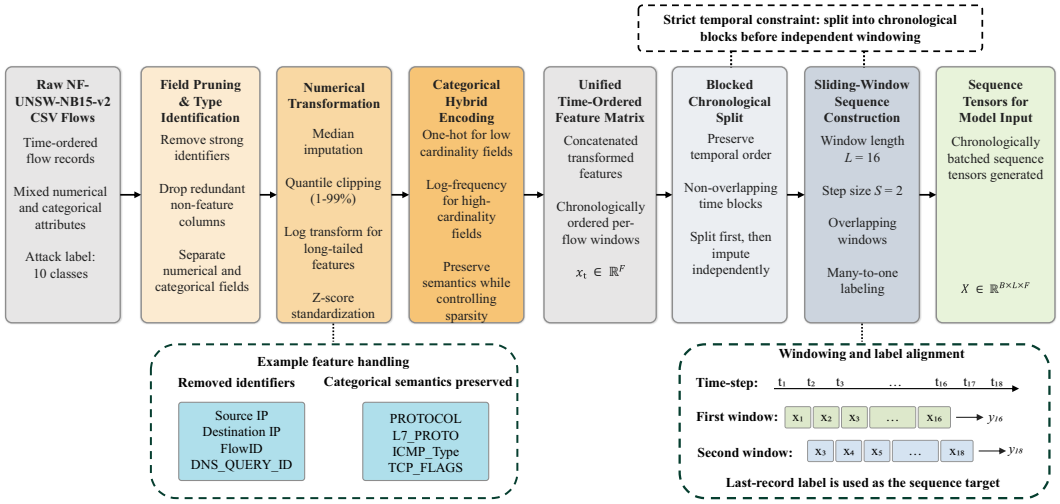


Figure 2: Preprocessing and sequence construction pipeline for NF-UNSW-NB15-v2.

Here, N_{num} is the number of retained numerical features, $|C_i|$ is the number of unique values of the i-th one-hot encoded categorical field, and N_{freq} is the number of categorical fields encoded by logarithmic frequency mapping.

3.3 Sequence Generation: Sliding Window Sampling and Temporal Constraints

After preprocessing, individual flow records are converted into fixed-length sequential samples using a sliding-window mechanism so that short-term temporal dependencies can be modeled explicitly. The sequence construction process is defined as:

$$M = \left\lfloor \frac{N-L}{S} \right\rfloor + 1,$$
$$\mathbf{X}_m = \left[\mathbf{x}_{1+(m-1)S}, \mathbf{x}_{2+(m-1)S}, \ldots, \mathbf{x}_{L+(m-1)S}\right] \in \mathbb{R}^{L \times F}, \quad m = 1, 2, \ldots, M. \tag{2}$$

where N denotes the number of flow records, L is the window length, S is the sliding step, and F is the encoded feature dimension. In this work, $L = 16$ and $S = 2$ are used to construct overlapping sequential samples. The choice of $L = 16$ follows a compact temporal-window design: it provides a short chronological receptive field for local traffic evolution while keeping the TinyTransformer input sequence small. This is important because the self-attention cost grows quadratically with the sequence length, i.e., $\mathcal{O}(L^2)$. The step size $S = 2$ gives an overlap ratio of $1 - S/L = 87.5\%$, which reduces boundary information loss for short attack segments and preserves sample continuity between adjacent windows. Therefore, $L = 16$ and $S = 2$ are used as a practical balance among temporal context, overlapping coverage, and computational efficiency, rather than as the result of an exhaustive window-size grid search.

It should be emphasized that the constructed sequence is a monitoring-point chronological context window rather than a reconstructed host session or bidirectional connection. Since identifier-like fields such as `Source IP`, `Destination IP`, and `FlowID` are removed to reduce host-identity leakage and deployment-specific memorization, adjacent records in a window may indeed originate from different hosts, protocols, or sessions. The purpose of the sliding window is therefore not to assume strict session-level semantics for all records in the window, but to model short-term changes in the observed traffic mixture, such as bursts, scanning periods, shifts in protocol composition, and attack-stage prevalence at the sensor level.

A many-to-one labeling strategy is adopted, in which the label of the last record in each window is used as the sequence target. This design treats the preceding $L - 1$ records as contextual evidence for predicting the state of the current record, rather than assuming that every record in the window necessarily shares the same label. To maintain temporal order, the dataset is first divided into non-overlapping chronological blocks. Sliding-window construction is then performed independently within the training and testing subsets. This split-then-window strategy preserves chronological continuity and avoids constructing windows across evaluation boundaries. Nevertheless, this chronological-window design is weaker than host- or session-level trajectory modeling; entity-aware sequence construction is therefore left as a useful direction for future work when reliable identifiers can be used without inducing shortcut learning.

4 Methodology

Building on the preprocessing and sequence construction in Section 3, the resulting fine-grained multiclass task involves two coupled challenges: joint local-temporal-global modeling of heterogeneous traffic sequences and minority-class learning under extreme long-tailed imbalance. To address them, this study proposes a compact TCN-BiGRU-TinyTransformer framework composed of a hybrid representation-learning architecture and the coordinated optimization strategy summarized in Fig. 3.

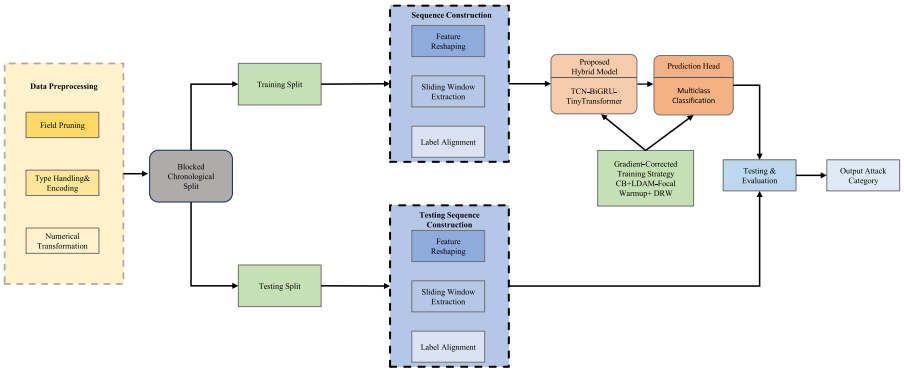


Figure 3: Overall framework of the proposed method.

The resulting chronological sequence samples are first encoded by the compact hybrid network and then optimized under the proposed strategy to perform fine-grained attack-category prediction. The detailed architecture of the TCN-BiGRU-TinyTransformer model is presented in Fig. 4.

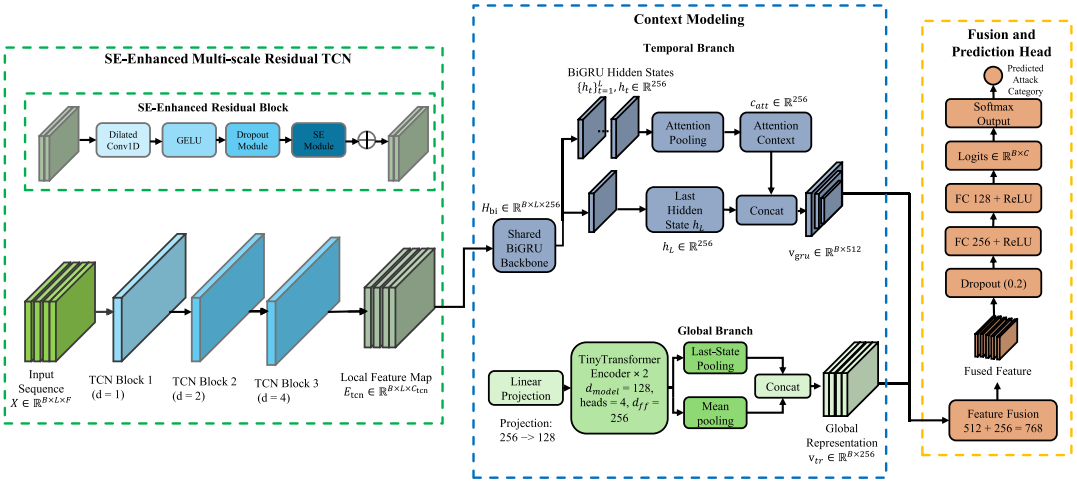


Figure 4: Detailed architecture of the proposed TCN-BiGRU-TinyTransformer model.

4.1 Compact Hybrid Network Architecture

As shown in Fig. 4, the proposed TCN-BiGRU-TinyTransformer model follows a hierarchical hybrid encoding architecture for fine-grained multiclass traffic analysis. Given the preprocessed input tensor $\mathbf{X} \in \mathbb{R}^{B\times L\times F}$, where B, L, and F denote the batch size, sequence length, and feature dimension, respectively, the model performs representation learning through three successive stages: local pattern extraction by a multi-scale residual TCN, temporal dependency modeling by a BiGRU with attention pooling, and global contextual interaction modeling by a compact TinyTransformer encoder. The resulting representations are then fused for final classification.

4.1.1 Local Feature Extraction via Multi-Scale Residual TCN with Squeeze-and-Excitation (SE)

To capture short-range anomalous patterns in flow sequences, a TCN front end with three residual blocks is adopted [3]. The dilation rates are set to $d \in \{1, 2, 4\}$ so that the receptive field can be enlarged progressively across multiple temporal scales while preserving local sensitivity. Each block uses one-dimensional convolutions with kernel size 3 and 128 channels, followed by Gaussian error linear unit (GELU) activation [30] and Dropout. An SE module is further embedded to recalibrate channel responses and suppress less informative features [31].

For the i-th residual block, the transformation is formulated as:

$$\mathbf{H}_i = F(\mathbf{H}_{i-1}; \mathbf{W}_i, d_i) + \mathbf{H}_{i-1}, \tag{3}$$

where $F(\cdot)$ denotes the residual branch parameterized by $\mathbf{W}_i$ under dilation rate d_i. Through this design, the TCN front end extracts multi-scale local representations and provides a stronger input basis for subsequent temporal and global modeling.

4.1.2 Temporal Dependency Modeling via Bidirectional GRU and Attention Pooling

After local multi-scale features are extracted by the TCN front end, a BiGRU is used to model bidirectional temporal dependencies over the sequence [4,5]. Let $\mathbf{h}_t$ denote the hidden representation at time step t. To emphasize informative positions, attention pooling is introduced to aggregate the BiGRU outputs:

$$\mathbf{g}_{\text{attn}} = \sum_{t=1}^{L} \alpha_t \mathbf{h}_t, \tag{4}$$

where α_t denotes the normalized attention weight of the t-th time step. Since the implemented BiGRU uses 128 hidden units in each direction, its output at each time step has 256 dimensions. The temporal branch concatenates the last hidden state and the attention-pooled representation, yielding $\mathbf{g}_{\text{gru}} \in \mathbb{R}^{B\times 512}$. Through this design, the BiGRU branch captures bidirectional temporal evolution while assigning higher importance to discriminative sequence positions.

4.1.3 Global Interaction Modeling via TinyTransformer Encoder

To further model longer-range contextual interactions, a compact TinyTransformer encoder is introduced after the BiGRU stage [6]. It consists of two encoder layers with $d_{\text{model}} = 128$, four attention heads, and $d_{\text{ff}} = 256$, as summarized in Table 1. Its core self-attention operation is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}. \tag{5}$$

Table 1: Key hyperparameters and training settings.

Parameter/Component	Configuration (Proposed Method)
Sequence Setup	Length $L = 16$, Sliding Step $S = 2$
TCN Block	Filters = 128, Kernel Size = 3, Dilation Rates = $\{1, 2, 4\}$, SE Block, Dropout = 0.2
BiGRU Block	Hidden Size = 128, Bidirectional = True, Attention Pooling
TinyTransformer	$d_{\text{model}} = 128$, Heads = 4, Layers = 2, $d_{\text{ff}} = 256$
Classifier	Input dimension = 768, MLP $768 \to 256 \to 128 \to C$, Dropout = 0.2
Optimization	AdamW, $lr = 1 \times 10^{-3}$, Weight Decay = 1×10^{-4}, ReduceLROnPlateau (Factor = 0.5, Patience = 2)
Training Schedule	Max Epochs = 50, Early Stopping (Patience = 8), Batch Size = 512
Long-Tail Strategy	Balanced batch sampler; CB Weight: $\beta = 0.9999$; LDAM-Focal: $\gamma = 2$, $m_{\max} = 0.5$; warmup: 3 epochs; DRW: 0.6 × epochs

Before Transformer encoding, the BiGRU sequence output is linearly projected from 256 to 128 dimensions at each time step, so the TinyTransformer input and output are both in $\mathbb{R}^{B\times L\times 128}$. Following Transformer encoding, sequence-level global information is aggregated through a hybrid final-state and mean-pooling scheme, yielding $\mathbf{g}_{\text{trans}} \in \mathbb{R}^{B\times 256}$. By doing so, the Transformer stage serves as a complement to the BiGRU branch, capturing broader contextual dependencies that extend beyond local structure and short-horizon temporal patterns.

4.1.4 Feature Fusion and Classification Decision

The temporal representation $\mathbf{g}_{\text{gru}}$, when taken together with the global contextual representation $\mathbf{g}_{\text{trans}}$, is concatenated to yield the final hybrid feature:

$$\mathbf{h}_{\text{fuse}} = [\mathbf{g}_{\text{gru}}; \mathbf{g}_{\text{trans}}] \in \mathbb{R}^{B\times 768}, \tag{6}$$

where 768 is obtained from the concatenation of the 512-dimensional temporal representation and the 256-dimensional global representation. The fused vector is then regularized by Dropout and fed into a two-layer multilayer perceptron (MLP) classification head with dimensions $768 \to 256 \to 128 \to C$, where $C = 10$ in this multiclass setting. Therefore, the fully connected layers with 256 and 128 units (FC 256 and FC 128) denote the hidden layers of the prediction head rather than the fused feature dimension. Through this fusion design, the classifier jointly exploits temporal-dependency information and global contextual interactions learned from the flow sequence.

4.2 Coordinated Long-Tail Optimization Strategy for Long-Tailed Distributions

Although the hybrid architecture improves sequence representation capability, the learning process is still strongly affected by the extreme class imbalance of NF-UNSW-NB15-v2. In particular, the benign and Worms classes differ by more than 4000:1 in the processed dataset, which makes conventional optimization prone to head-class dominance and reduced optimization signals for minority classes. To address this issue, a coordinated long-tail optimization strategy is designed from three complementary aspects: data sampling, class-weight correction, and dynamic optimization.

4.2.1 Data-Level: Class-Balanced Mini-Batch Sampling

Under conventional random sampling, minority classes appear only rarely in mini-batches, which reduces their contribution to parameter updates. Therefore, a balanced batch sampler is adopted to construct approximately class-balanced mini-batches. Specifically, minority classes are sampled with replacement, whereas majority classes are sampled without replacement, so that the expected per-class exposure within a batch becomes more balanced during training.

4.2.2 Weight-Level: Class Reweighting Based on the Effective Number of Samples

At the loss level, class weights are computed according to the effective number of samples [23]:

$$w_c^{(\text{cb})} = \frac{1-\beta}{1-\beta^{n_c}}, \tag{7}$$

where n_c denotes the number of training samples in class c, and $\beta = 0.9999$ in this work. These class-balanced weights are used to reduce optimization bias toward head classes under extreme imbalance. In the following optimization objective, the class weight of the ground-truth category is written as $w_y = w_y^{(\text{cb})}$.

4.2.3 Optimization-Level: LDAM-Focal Loss with Warmup

To further improve the decision boundary under long-tailed distributions, the final training objective combines LDAM-based margin adjustment with Focal Loss:

$$\Delta_y = \frac{m_{\max}\, n_y^{-1/4}}{\max_{k\in\{1,\ldots,C\}} n_k^{-1/4}}, \tag{8}$$

$$\mathcal{L} = -w_y(1-p_y)^{\gamma}\log\left(\frac{\exp(s(z_y-\Delta_y))}{\exp(s(z_y-\Delta_y))+\sum_{\substack{j=1\\ j\neq y}}^{C}\exp(sz_j)}\right), \tag{9}$$

where C is the number of classes, $y \in \{1,\ldots,C\}$ is the ground-truth class index, and j and k are class indices. The class-frequency term n_k denotes the number of training samples in class k within the current training fold. The logit of class j is denoted by z_j, and Δ_y denotes the class-dependent LDAM margin of the ground-truth class. The weight w_y denotes the corresponding class-balanced weight defined in Eq. (7). The probability p_y is the margin-adjusted softmax probability of the ground-truth class, corresponding to the fraction inside the logarithm in Eq. (9). The parameters s and γ are the scaling and focusing parameters, respectively. In this work, $\gamma = 2$ and $m_{\max} = 0.5$.

The choices of β, γ, and $m_{\max}$ can be interpreted from their mathematical roles in the objective. In Eq. (7), β determines how the effective number of samples grows with the class frequency n_c. A smaller β makes the weights closer to uniform class weighting, whereas a value closer to 1 makes the effective sample number saturate more evidently for head classes and therefore strengthens the smoothing of the long-tailed distribution. Thus, β mainly controls the degree of distribution-level smoothing and class rebalancing. By contrast, γ acts on the instance-level focal factor $(1-p_y)^{\gamma}$ in Eq. (9); increasing γ suppresses easy samples with high p_y more strongly and preserves relatively larger gradients for difficult samples. The parameter $m_{\max}$ controls the upper bound of the LDAM margin in Eq. (8); since Δ_y is proportional to $m_{\max} n_y^{-1/4}$ after normalization, minority classes receive a larger corrective margin, while excessively large margins may destabilize the decision boundary. The values $\beta = 0.9999$, $\gamma = 2$, and $m_{\max} = 0.5$ were selected as a stable balance point in preliminary small-scale tuning, so the main experiments report this final setting rather than an exhaustive hyperparameter grid.

To stabilize optimization, a three-stage schedule is adopted. During epochs 1–3, standard cross-entropy loss is used for warmup. During epochs 4–30, LDAM-Focal is applied without class-balanced reweighting to support basic representation learning. During epochs 31–50, full reweighting is enabled so that the decision boundary can be further refined for minority classes. By structuring the procedure in stages, training stability is improved, even as the corrective contribution of long-tail optimization is retained.

5 Performance Evaluation

This section assesses the proposed framework across five evaluative dimensions: baseline performance, comparison with prior studies, ablation outcomes, fine-grained diagnostic results, and computational efficiency.

5.1 Experimental Setup

5.1.1 Dataset Split and Evaluation Metrics

A five-fold rolling-origin blocked time-series protocol is adopted to preserve temporal order [32]. After chronological ordering and preprocessing, the processed flow records are divided into six contiguous non-overlapping temporal blocks. For fold k ($k = 1,\ldots,5$), block B_k is used as the test block, all earlier blocks $\{B_0,\ldots,B_{k-1}\}$ form the historical training portion, and the last 10% of this historical portion is held out as the validation set for early stopping and model selection. Therefore, the test block moves forward with each fold, while no future flow record is used to fit preprocessing statistics, construct class weights, tune early stopping, or train model parameters. Sliding windows are then generated independently within the

training, validation, and test subsets to avoid windows crossing evaluation boundaries. Table 2 summarizes the resulting split sizes and window-label distributions.

Table 2: Rolling-origin blocked time-series fold construction.

Fold	Split	Flows	Windows	Window-Label Distribution
1	Train	114064	57025	1, 11, 55368, 92, 519, 711, 82, 208, 29, 4
1	Val.	12673	6329	0, 0, 6329, 0, 0, 0, 0, 0, 0, 0
1	Test	126737	63361	0, 0, 63361, 0, 0, 0, 0, 0, 0, 0
2	Train	228127	114056	1, 11, 112399, 92, 519, 711, 82, 208, 29, 4
2	Val.	25347	12666	0, 0, 12666, 0, 0, 0, 0, 0, 0, 0
2	Test	126737	63361	173, 177, 56700, 380, 2006, 1534, 1549, 745, 88, 9
3	Train	342190	171088	35, 44, 166870, 197, 1336, 1277, 717, 538, 67, 7
3	Val.	38021	19003	139, 144, 14904, 275, 1189, 968, 913, 415, 50, 6
3	Test	126737	63361	295, 318, 50380, 785, 4324, 2823, 2451, 1745, 210, 30
4	Train	456254	228120	356, 357, 212265, 902, 5042, 4035, 2937, 1974, 232, 20
4	Val.	50694	25340	129, 170, 19989, 334, 1760, 1131, 1010, 739, 72, 6
4	Test	126737	63361	346, 336, 50539, 810, 4460, 2998, 1775, 1851, 219, 27
5	Train	570317	285151	646, 713, 257266, 1628, 8986, 6751, 5096, 3623, 402, 40
5	Val.	63368	31677	146, 184, 25488, 418, 2330, 1451, 601, 953, 99, 7
5	Test	126739	63362	351, 224, 49976, 809, 4439, 3048, 2483, 1803, 201, 28

Given the severe long-tailed distribution of NF-UNSW-NB15-v2, this study reports Accuracy, macro-averaged Precision/Recall/F1, and weighted-averaged Precision/Recall/F1. Macro-F1 is treated as the primary metric because it better reflects minority-class performance under imbalanced multiclass classification [20,21]. Validation Macro-F1 is also used as the criterion for early stopping and model selection. Weighted metrics and Accuracy are reported to assess whether macro-level gains are achieved without substantial loss of overall performance.

5.1.2 Implementation Details

All models are implemented in PyTorch 2.8.0 with Python 3.12 and CUDA 12.8. Experiments are conducted on a workstation with a single NVIDIA RTX 5090 GPU (32 GB VRAM), an Intel Xeon CPU, and 90 GB RAM. Random seeds are fixed throughout. The model is optimized with AdamW [33], and staged warmup is applied before full reweighting [34]. Unless otherwise stated, the same blocked time-series protocol, optimization settings, and evaluation procedure are used across all experiments. Table 1 summarizes the core hyperparameters, optimization settings, and long-tail training schedule of the proposed method.

5.2 Results and Discussion

Based on the experimental protocol and evaluation metrics described above, this subsection reports the quantitative results and provides empirical analysis from multiple perspectives. Table 3 compares this method with six baseline methods on the NF-UNSW-NB15-v2 dataset, including XGBoost [35], long short-term memory (LSTM), one-dimensional convolutional neural network with bidirectional long short-term memory (1D-CNN+BiLSTM), TCN+GRU, and two recent competitive models, bidirectional encoder

representations from transformers with conditional generative adversarial network (BERT-CGAN) [36] and FlowTransformer [7]. All methods were trained and evaluated under the same five-fold blocked time-series protocol and the same preprocessing pipeline. Under this protocol, BERT-CGAN obtains 98.01% Accuracy and 75.80% Macro-F1, while FlowTransformer obtains 98.08% Accuracy and 76.01% Macro-F1. While these recent architectures perform well in terms of overall Accuracy, they do not explicitly account for the extreme long-tailed class imbalance of NF-UNSW-NB15-v2. In contrast, the proposed model achieves a superior Macro-F1 of 80.19% and Macro Precision of 82.28%, demonstrating that the coordinated long-tail optimization strategy effectively alleviates majority-class bias and provides more robust minority-class discrimination without sacrificing overall detection performance.

Table 3: Protocol-matched baseline comparison on NF-UNSW-NB15-v2.

Method	Acc/%	Macro-Averaged/%			Weighted-Averaged/%		
		Precision	Recall	F1	Precision	Recall	F1
XGBoost	97.23 ± 0.21	67.30 ± 1.85	74.30 ± 1.52	68.29 ± 1.68	97.93 ± 0.18	97.23 ± 0.19	97.49 ± 0.18
LSTM	97.82 ± 0.16	76.79 ± 1.22	75.85 ± 1.15	76.01 ± 1.18	97.86 ± 0.14	97.82 ± 0.15	97.82 ± 0.14
1D-CNN+BiLSTM	97.90 ± 0.14	77.73 ± 1.18	77.63 ± 1.05	77.48 ± 1.10	97.98 ± 0.13	97.90 ± 0.14	97.93 ± 0.12
TCN+GRU	97.79 ± 0.18	78.22 ± 0.05	74.64 ± 0.12	75.64 ± 0.08	97.80 ± 0.15	97.79 ± 0.14	97.78 ± 0.16
BERT-CGAN	98.01 ± 0.09	75.12 ± 1.95	76.50 ± 1.62	75.80 ± 1.78	98.01 ± 0.06	98.01 ± 0.09	97.99 ± 0.07
FlowTransformer	98.08 ± 0.06	76.85 ± 1.45	75.20 ± 1.80	76.01 ± 1.62	98.09 ± 0.06	98.08 ± 0.06	98.06 ± 0.06
Ours	98.06 ± 0.12	82.28 ± 0.91	79.03 ± 0.78	80.19 ± 0.82	98.06 ± 0.11	98.06 ± 0.10	98.05 ± 0.11

5.2.1 Comparison with Existing Studies

Table 4 compares the proposed method with several representative studies on multiclass classification of the NF-UNSW-NB15-v2 dataset (references [19,36,37]). As noted above, these literature-reported results are retained only as an informative reference rather than as a strict head-to-head comparison, because different studies adopt different preprocessing pipelines, feature-engineering choices, sampling rates, data segmentation strategies, and evaluation protocols. Therefore, Table 4 is used only for contextual comparison, whereas Table 3 remains the primary protocol-matched comparison.

Table 4: Comparison with existing works on NF-UNSW-NB15-v2.

Study	Model	Acc/%	Macro-Averaged/%			Weighted-Averaged/%		
			Precision	Recall	F1	Precision	Recall	F1
Panel I: Macro-averaged results								
Ours	TCN–BiGRU–TinyTransformer	98.06	82.28	79.03	80.19	98.06	98.06	98.05
Alzaher & AlJarullah [37]	XGBoost	99.00	73.00	80.00	71.00	–	–	–
Sayed et al. [19]	IoTCNN	–	36.50	71.70	40.20	–	–	–
Panel II: Weighted-averaged results								
Ours	TCN–BiGRU–TinyTransformer	98.06	–	–	–	98.06	98.06	98.05
Li et al. [36]	BERT-CGAN	87.40	–	–	–	91.69	87.40	89.01

5.2.2 Ablation Study

(1) **Component contribution analysis.** Table 5 gives the results of architectural ablation analysis. Of all the evaluated variants, the full TCN-BiGRU-TinyTransformer configuration produces the strongest performance. After removing TCN, the decline of Macro-F1 is the most pronounced, which indicates that local multi-scale feature extraction provides the most important front-end contribution. When BiGRU or TinyTransformer is excluded, the performance also deteriorates. Both variants support the same conclusion: time series modeling and global context interaction have made beneficial contributions.

(2) **Effectiveness analysis of the long-tail training strategy.** Table 6 reports the ablation results of the long-tail training strategy. The full strategy achieves the best Macro-F1 (80.19%), while all incomplete variants underperform it, indicating that exposure balancing, class reweighting, and margin-aware optimization contribute complementarily to minority-class recognition under extreme imbalance.

(3) **Fine-grained component ablation.** To further examine the components that are not isolated in Tables 5 and 6, Table 7 reports additional ablation results for the SE module, attention pooling, and staged warmup schedule. Removing the SE module reduces Macro-F1 from 80.19% to 78.42%, indicating that channel recalibration contributes to discriminative local feature extraction. Removing attention pooling decreases Macro-F1 to 79.66%, which suggests that explicit aggregation of informative BiGRU states improves sequence-level temporal representation. The largest decrease appears when staged warmup is removed, where Macro-F1 drops to 75.33%. This result supports the role of staged optimization in stabilizing long-tail training before full reweighting is applied. Together with the class-wise diagnostic results in Table 8, these ablations provide a more complete view of both component-level contribution and remaining tail-class behavior.

Table 5: Ablation study of architectural components.

Variant	Acc/%	Macro-Averaged/%			Weighted-Averaged/%		
		Precision	Recall	F1	Precision	Recall	F1
TCN+BiGRU	97.88	79.42	77.25	77.30	97.90	97.88	97.87
TCN+ TinyTransformer	97.82	76.77	77.30	76.58	97.87	97.82	97.82
BiGRU+ TinyTransformer	97.84	75.57	76.53	75.54	97.87	97.84	97.84
Ours	98.06	82.28	79.03	80.19	98.06	98.06	98.05

Table 6: Ablation study of long-tailed training strategies.

Variant	Acc/%	Macro-Averaged/%			Weighted-Averaged/%		
		Precision	Recall	F1	Precision	Recall	F1
LDAM+CB	97.92	80.00	76.53	77.82	97.93	97.92	97.91
LDAM+sampler	97.80	78.18	77.49	76.78	97.85	97.80	97.80
CB+sampler	97.79	78.64	76.47	76.75	97.83	97.79	97.78
Ours	98.06	82.28	79.03	80.19	98.06	98.06	98.05

Table 7: Component ablation study on NF-UNSW-NB15-v2 (mean ± std across five folds).

Variant	Acc/%	Macro-P/%	Macro-R/%	Macro-F1/%	W-F1/%
Ours	98.06 ± 0.12	82.28 ± 0.91	79.03 ± 0.78	80.19 ± 0.82	98.05 ± 0.11
w/o SE module	97.94 ± 0.15	80.15 ± 1.42	77.64 ± 1.12	78.42 ± 0.95	97.93 ± 0.14
w/o attention pooling	98.01 ± 0.08	81.54 ± 1.10	78.42 ± 0.85	79.66 ± 0.74	98.01 ± 0.08
w/o staged warmup	97.82 ± 0.21	77.12 ± 1.95	74.58 ± 1.54	75.33 ± 1.68	97.81 ± 0.19

Table 8: Class-wise performance of the proposed model on NF-UNSW-NB15-v2.

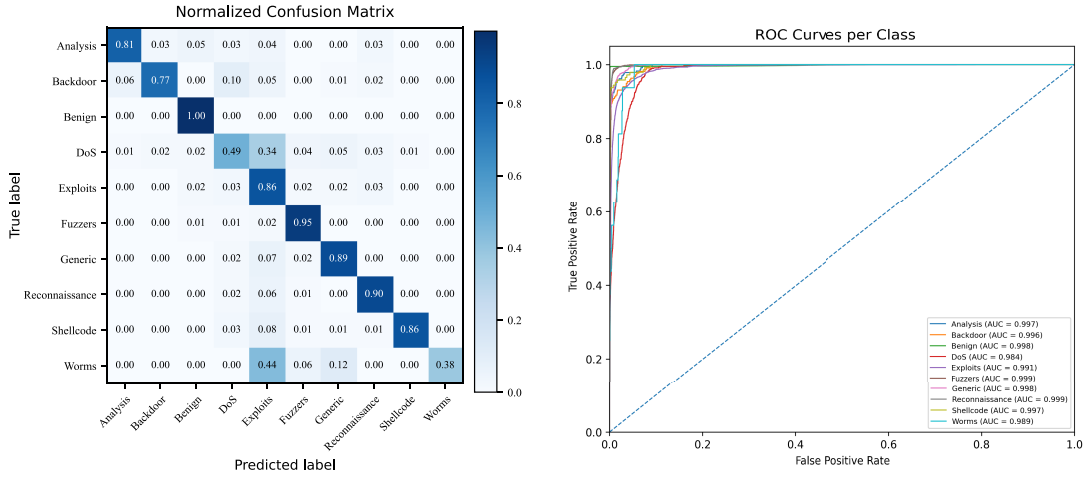
Class	Precision/%	Recall/%	F1/%	F1 std
Analysis	79.50	80.85	80.17	±1.92
Backdoor	81.07	76.61	78.77	±2.15
Benign	99.84	99.70	99.77	±0.05
DoS	57.11	48.97	52.73	±6.42
Exploits	83.83	85.96	84.89	±1.23
Fuzzers	90.07	95.50	92.70	±0.87
Generic	91.63	88.50	90.04	±0.95
Reconnaissance	88.60	90.50	89.54	±1.10
Shellcode	84.46	86.21	85.32	±1.68
Worms	66.67	37.50	48.00	±13.69

5.2.3 Fine-Grained Analysis

For fine-grained diagnostic analysis, Table 8 reports the class-wise Precision, Recall, and F1-score obtained under five-fold blocked time-series cross-validation, and Fig. 5 presents the corresponding confusion matrix and multiclass receiver operating characteristic (ROC) curves. Across the five folds, the primary metric Macro-F1 exhibits a standard deviation of ±0.82%, confirming stable temporal generalization under the blocked time-series protocol.

(1) **Class-wise performance and remaining tail-class gaps.** The proposed method performs strongly on most categories, with an F1-score of 99.77% for Benign and 84.89%–92.70% for major attack classes including Exploits, Fuzzers, Generic, Reconnaissance, and Shellcode. However, the results also show that the coordinated long-tail strategy mitigates but does not fully solve the most difficult tail-class cases. In particular, DoS obtains 57.11% Precision, 48.97% Recall, and 52.73% F1, while Worms obtains 66.67% Precision, 37.50% Recall, and 48.00% F1. These results are explicitly treated as remaining limitations rather than as complete resolution of the long-tail problem.
For Worms, the main bottleneck is the extremely small number of available samples, which limits the diversity of temporal patterns that can be learned even when reweighting and balanced sampling are applied. Its relatively higher Precision but much lower Recall indicates a conservative decision pattern: when the model predicts Worms, the prediction is often reliable, but many true Worms instances are still assigned to other classes. For DoS, the challenge is different. DoS traffic shares aggregate NetFlow characteristics with more frequent attack categories, especially volume- or burst-related patterns that overlap with Exploits and Generic traffic. This overlap makes the final multiclass boundary more ambiguous, so a portion of DoS samples is still absorbed by neighboring high-frequency classes.

(2) **Area under the curve (AUC)–F1 disconnect and practical implications.** Fig. 5 shows that DoS and Worms still achieve high one-vs.-rest AUC values of 0.984 and 0.989, respectively, despite their low F1-scores. This indicates that the learned representations retain useful ranking information, but the final hard decision under severe imbalance remains conservative for rare or overlapping categories. In practical IDS deployment, this behavior has two implications. First, low Recall for Worms and DoS means that these classes should not be treated as fully solved by the current model; missed detections may still occur and should be considered in high-sensitivity environments. Second, the relatively conservative predictions can reduce false alarms, but they may require downstream threshold adjustment, analyst review, or risk-aware post-processing when detecting rare high-impact attacks is more important than minimizing alert volume. Therefore, the proposed strategy should be understood as improving the overall long-tailed multiclass balance, while further work is still needed to strengthen recall for the most difficult minority classes.



a Normalized confusion matrix **b** Multiclass ROC curves

Figure 5: Classification performance of the proposed model on NF-UNSW-NB15-v2.

5.2.4 Cross-Dataset Validation on NF-ToN-IoT-v2

The proposed framework is further evaluated on NF-ToN-IoT-v2, an IoT-oriented NetFlow dataset with a ten-class label space. The evaluation uses 761,360 chronologically ordered NetFlow records, including Benign, Backdoor, distributed denial-of-service (DDoS), denial-of-service (DoS), Injection, man-in-the-middle (MITM), Password, Ransomware, Scanning, and cross-site scripting (XSS). The same model configuration in Table 1 and the blocked time-series protocol described in Section 5 are used, with sequential samples generated using the standard sequence configuration ($L = 16$, $S = 2$).

As summarized in Table 9, the proposed model achieves 96.18 ± 0.11% Accuracy, 93.09 ± 0.18% Macro-Precision, 91.78 ± 0.14% Macro-Recall, 92.30 ± 0.23% Macro-F1, and 96.14 ± 0.12% Weighted-F1 on NF-ToN-IoT-v2. These results suggest that the proposed local–temporal–contextual representation and coordinated long-tail optimization can maintain stable multiclass detection performance on an IoT-oriented NetFlow benchmark. Since NF-UNSW-NB15-v2 serves as the primary benchmark for detailed class-wise diagnosis, the NF-ToN-IoT-v2 results are reported at the aggregate level in Table 9 to provide compact cross-dataset validation.

Table 9: Cross-dataset validation results on NF-ToN-IoT-v2.

Dataset	Acc/%	Macro-P/%	Macro-R/%	Macro-F1/%	Weighted-F1/%
NF-ToN-IoT-v2	96.18 ± 0.11	93.09 ± 0.18	91.78 ± 0.14	92.30 ± 0.23	96.14 ± 0.12

5.2.5 Model Complexity and Resource Efficiency

The efficiency assessment compares real evaluated model structures under a unified deployment-time profiling protocol: FlowTransformer, BERT-CGAN, 1D-CNN+BiLSTM, TCN+GRU, and the proposed TCN-BiGRU-TinyTransformer. The purpose of this analysis is to report parameter count, peak GPU memory, and inference latency under the same input format and measurement procedure.

Efficiency evaluation protocol. All models are evaluated in FP32 inference mode with `model.eval()` and `torch.inference_mode()`. A fixed single-sample input tensor with shape $(1, 16, 40)$ is extracted from the same NF-UNSW-NB15-v2 preprocessing pipeline. Latency is measured using CUDA events with 50 warm-up iterations and 200 timed iterations, and is reported as milliseconds per sample. Peak GPU memory is obtained via `torch.cuda.max_memory_allocated()` after resetting memory statistics, using 25 warm-up and 50 measured forward passes. For BERT-CGAN, only the deployment-time detector/classifier is profiled; the training-time conditional generative adversarial network (CGAN) generator is not included in online inference latency, memory, or parameter count.

As shown in Fig. 6, the proposed model contains 0.824M trainable parameters, requires 18.52 MB peak GPU memory during FP32 inference, and achieves 0.954 ms/sample single-sample latency. These values place the model within a compact sub-million-parameter and millisecond-level inference regime while supporting the full local-temporal-global representation pipeline needed for fine-grained multiclass intrusion detection.

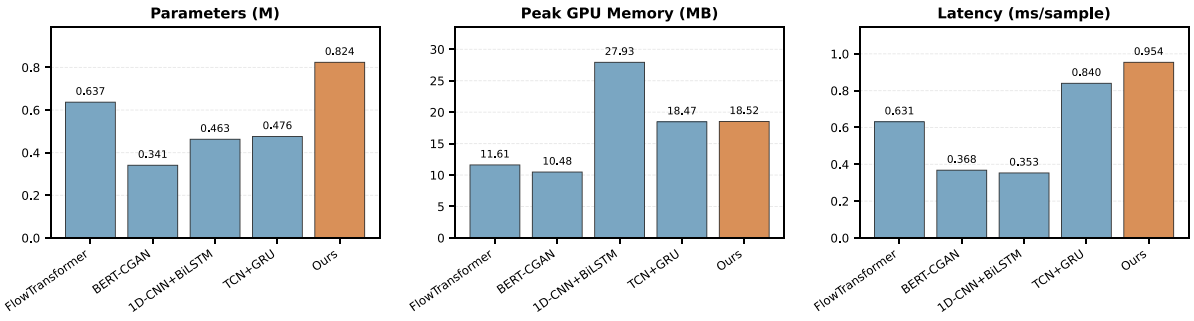


Figure 6: Computational efficiency comparison with real baseline architectures.

6 Conclusion

A compact TCN-BiGRU-TinyTransformer framework and a coordinated long-tail optimization strategy were proposed for fine-grained multiclass intrusion detection in flow-level traffic. Unlike traditional models that sacrifice minority class recognition to maximize overall accuracy in severe imbalance, the proposed architecture provides a comprehensive representation pipeline by jointly capturing local anomalies, temporal dependencies, and global context. The experiments on NF-UNSW-NB15-v2 and NF-ToN-IoT-v2 show that the framework maintains strong macro-level detection performance while retaining practical deployment efficiency under blocked time-series evaluation.

Future work. Guided by the identified performance bottlenecks and practical requirements in this study, future research will focus on three directions to further enhance the utility of fine-grained intrusion detection. First, to address the conservative prediction pattern caused by the scarcity of extreme tail-class samples such as Worms, this work plans to investigate few-shot learning and synthetic feature generation techniques that can strengthen optimization signals for rare attack types. Second, to alleviate the observed feature overlap between classes such as DoS and Exploits, decomposed representation learning will be explored to better distinguish latent traffic patterns. Third, to bridge the gap between detection and response, it is planned to integrate these fine-grained detection outputs into dynamic risk assessment and adaptive access control frameworks. By quantifying the threat level of specific attack categories, future systems may better support risk identification, assessment, and response in changing network environments.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by the National Natural Science Foundation of China, grant number 62462044.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Ye Lu and Haoyang Hu; methodology, Haoyang Hu and Ye Lu; software, Haoyang Hu; validation, Haoyang Hu and Wenyi Chang; formal analysis, Haoyang Hu; investigation, Haoyang Hu; data curation, Haoyang Hu; visualization, Haoyang Hu; writing—original draft preparation, Haoyang Hu; writing—review and editing, Ye Lu, Haoyang Hu, Wenyi Chang, Wanbin Liu and Wenyuan Zhang; supervision, Ye Lu and Wenyuan Zhang; project administration, Ye Lu; funding acquisition, Ye Lu. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in the University of Queensland eSpace at https://doi.org/10.48610/ffbb0c1. The source code is available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

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

References

1. Chou TS, Jiang S. A survey on data-driven network intrusion detection. ACM Comput Surv. 2021;54(9):1–36. doi:10.1145/3472753.
2. Kilincer IF, Ertam F, Sengur A. Machine learning methods for cyber security intrusion detection: datasets and comparative study. Comput Netw. 2021;188:107840. doi:10.1016/j.comnet.2021.107840.
3. Bai S, Kolter JZ, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271. 2018. doi:10.48550/arXiv.1803.01271.
4. Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE Trans Signal Process. 1997;45(11):2673–81. doi:10.1109/78.650093.
5. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014 Oct 25–29; Doha, Qatar. p. 1724–34. doi:10.3115/v1/D14-1179.
6. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017); 2017 Dec 4–9; Long Beach, CA, USA. p. 5998–6008.
7. Manocchio LD, Layeghy S, Lo WW, Kulatilleke GK, Sarhan M, Portmann M. FlowTransformer: a transformer framework for flow-based network intrusion detection systems. Expert Syst Appl. 2024;241(10):122564. doi:10.1016/j.eswa.2023.122564.

8. Tiwari RS, Lakshmi D, Das TK, Tripathy AK, Li KC. A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security. Telecommun Syst. 2024;87(3):605–24. doi:10.1007/s11235-024-01200-y.
9. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. 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); 2009 Jul 8–10; Ottawa, ON, Canada. p. 1–6. doi:10.1109/CISDA.2009.5356528.
10. 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); 2015 Nov 10–12; Canberra, Australia. p. 1–6. doi:10.1109/MilCIS.2015.7348942.
11. Sarhan M, Layeghy S, Portmann M. Towards a standard feature set for network intrusion detection system datasets. Mob Netw Appl. 2022;27(1):357–70. doi:10.1007/s11036-021-01843-0.
12. Sarhan M, Layeghy S, Portmann M. Machine learning-based network intrusion detection system datasets in NetFlow and CICFlowMeter representations. St. Lucia, QLD, Australia: The University of Queensland; 2023. doi:10.48610/ffbb0c1.
13. Alsaedi A, Moustafa N, Tari Z, Mahmood A, Anwar A. TON_IoT telemetry dataset: a new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access. 2020;8:165130–50. doi:10.1109/ACCESS.2020.3022862.
14. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H. Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access. 2022;10:40281–306. doi:10.1109/ACCESS.2022.3165809.
15. Gamage S, Samarabandu J. Deep learning methods in network intrusion detection: a survey and an objective comparison. J Netw Comput Appl. 2020;169(2):102767. doi:10.1016/j.jnca.2020.102767.
16. Henry A, Gautam S, Khanna S, Rabie K, Shongwe T, Bhattacharya P, et al. Composition of hybrid deep learning model and feature optimization for intrusion detection system. Sensors. 2023;23(2):890. doi:10.3390/s23020890.
17. Yaras S, Dener M. IoT-based intrusion detection system using new hybrid deep learning algorithm. Electronics. 2024;13(6):1053. doi:10.3390/electronics13061053.
18. Chen Z, Zou H, Hu T, Yuan X, Fang X, Pan Y, et al. HC-NIDS: historical contextual information based network intrusion detection system in Internet of Things. Comput Secur. 2025;152(6):104367. doi:10.1016/j.cose.2025.104367.
19. Sayed N, Shoaib M, Ahmed W, Qasem SN, Albarrak AM, Saeed F. Augmenting IoT intrusion detection system performance using deep neural network. Comput Mater Contin. 2023;74(1):1351–74. doi:10.32604/cmc.2023.030831.
20. He H, Garcia EA. Learning from imbalanced data. IEEE Trans Knowl Data Eng. 2009;21(9):1263–84. doi:10.1109/TKDE.2008.239.
21. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inf Process Manag. 2009;45(4):427–37. doi:10.1016/j.ipm.2009.03.002.
22. Abdelkhalek A, Mashaly M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. J Supercomput. 2023;79(10):10611–44. doi:10.1007/s11227-023-05073-x.
23. Cui Y, Jia M, Lin TY, Song Y, Belongie S. Class-balanced loss based on effective number of samples. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019 Jun 15–20; Long Beach, CA, USA. p. 9268–77. doi:10.1109/CVPR.2019.00949.
24. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2017 Oct 22–29; Venice, Italy. p. 2980–8. doi:10.1109/ICCV.2017.324.
25. Cao K, Wei C, Gaidon A, Aréchiga N, Ma T. Learning imbalanced datasets with label-distribution-aware margin loss. In: Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019); 2019 Dec 8–14; Vancouver, BC, Canada.
26. Jamoos M, Mora AM, AlKhanafseh M, Surakhi O. A new data-balancing approach based on generative adversarial network for network intrusion detection system. Electronics. 2023;12(13):2851. doi:10.3390/electronics12132851.

27. Rao YN, Suresh Babu K. An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset. Sensors. 2023;23(1):550. doi:10.3390/s23010550.
28. Yang H, Xu J, Xiao Y, Hu L. SPE-ACGAN: a resampling approach for class imbalance problem in network intrusion detection systems. Electronics. 2023;12(15):3323. doi:10.3390/electronics12153323.
29. Zhang W, Chen Z, Chen D, Li J, Pan Y. DID-IDS: a novel diffusion-based imbalanced data intrusion detection system. In: Proceedings of the 2023 IEEE 11th International Conference on Information, Communication and Networks (ICICN); 2023 Aug 17–20; Xi'an, China. p. 364–9. doi:10.1109/ICICN59530.2023.10392308.
30. Hendrycks D, Gimpel K. Gaussian error linear units (GELUs). arXiv:1606.08415. 2016. doi:10.48550/arXiv.1606.08415.
31. Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2018 Jun 18–23; Salt Lake City, UT, USA. p. 7132–41. doi:10.1109/CVPR.2018.00745.
32. Landauer M, Skopik F, Stojanović B, Flatscher A, Ullrich T. A review of time-series analysis for cyber security analytics: from intrusion detection to attack prediction. Int J Inf Secur. 2025;24(1):3. doi:10.1007/s10207-024-00921-0.
33. Loshchilov I, Hutter F. Decoupled weight decay regularization. In: Proceedings of the International Conference on Learning Representations (ICLR); 2019 May 6–9; New Orleans, LA, USA. doi:10.48550/arXiv.1711.05101.
34. Ma J, Yarats D. On the adequacy of untuned warmup for adaptive optimization. Proc AAAI Conf Artif Intell. 2021;35(10):8828–36. doi:10.1609/aaai.v35i10.17069.
35. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD); 2016 Aug 13–17; San Francisco, CA, USA. p. 785–94. doi:10.1145/2939672.2939785.
36. Li F, Shen H, Mai J, Wang T, Dai Y, Miao X. Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection. Peer-to-Peer Netw Appl. 2024;17(1):227–45. doi:10.1007/s12083-023-01595-6.
37. Alzaher FJ, AlJarullah A. Intrusion detection using machine learning and deep learning. Int J Adv Comput Sci Appl. 2025;16(8):440–54. doi:10.14569/IJACSA.2025.0160844.