Journal of Artificial Intelligence and Technology, (Ahead of Print)
https://doi.org/10.37965/jait.2026.1482

ISTP
RESEARCH ARTICLE

Traffic-Aware Imbalance Learning Network for Lightweight IoT Intrusion Detection

Sowmya Somanath,[1,3] Usha Banavikal Ajay,[1] and Sangeetha Kanevi Nanjunda Swamy[2]
[1]Department of Computer Science and Engineering, BMS Institute of Technology and Management, Visvesvaraya Technological University, Belagavi-590018, India
[2]Department of Electronics and Communication Engineering, JSSATE, Visvesvaraya Technological University, Belagavi-590018, India
[3]School of Computer Science and Engineering, REVA University, Yelahanka Bangalore-560064, India

(Received 05 June 2026; Revised 31 July 2026; Accepted 10 August 2026; Published online 07 September 2026)

Abstract: The rapid growth of internet of things (IoT) networks has significantly increased cybersecurity risks due to heterogeneous traffic characteristics, large-scale connectivity, and highly imbalanced attack distributions. Existing intrusion detection approaches primarily focus on improving overall detection performance through computationally expensive deep learning architectures or synthetic oversampling techniques. However, such methods often overlook semantic relationships among traffic features, increase computational complexity, and exhibit a limited capability to learn minority attack patterns. This paper presents a Traffic-Aware Imbalance Learning Network (TAIL-Net) for lightweight and imbalance-aware IoT intrusion detection. The proposed framework introduces a traffic-aware semantic feature mapping mechanism that reorganizes network traffic attributes according to their semantic relationships to improve feature representation learning. The mapped features are processed through a lightweight convolutional neural network (CNN)-GRU architecture integrated with a minority-aware attention and an Adaptive Class-Balanced Focal Loss (ACB-FL) function to enhance minority attack discrimination without relying on synthetic oversampling. Experimental evaluation on the Botnet-internet of things (BoT-IoT) dataset demonstrates that the proposed framework achieved 99% detection accuracy and 99% weighted F1-score. Furthermore, TAIL-Net requires only 81,866 trainable parameters with a model size of 0.3122 MB and achieves an average inference latency of 0.0041 ms/sample.

Keywords: Class imbalance; deep learning; Intrusion detection system; internet of things (IoT); severity analysis

I. INTRODUCTION

In recent years, due to the emergence of low-cost devices, wireless communication tools, and edge computing technologies, internet of things (IoT) networks have been widely used in many real-time application fields such as healthcare systems, industrial automation, smart transportation, agriculture, and smart cities [1]. However, the limited computing resources, memory, and battery capacity of IoT devices make them vulnerable to various security attacks. Furthermore, the constantly evolving nature of attacks and the generation of high-dimensional traffic data further exacerbate the complexity of the IoT ecosystem [2], due to which device compromise becomes common. Therefore, depending only on traditional security approaches like encryption [3] and authentication [4] becomes unworkable. For example, a smart camera is authenticated, and its communication is encrypted, but malware infects the camera. When the device starts, the infected device begins scanning ports, generating Distributed Denial of Service (DDoS); Bidirectional Long Short-Term Memory (BiLSTM) traffic, and communicating with botnet servers. In this case, the encryption and authentication still work, but the behavior is malicious, which can only be identified by intrusion detection systems (IDSs) during runtime operation.

IDSs have been extensively investigated in wireless and network security research, where traditional approaches mainly depended on signature-based detection and predefined attack rules [5]. However, these approaches have become ineffective against dynamic, unknown, and zero-day attacks. With the advancements in artificial intelligence (AI), machine learning (ML) and deep learning (DL) techniques have been widely adopted for intelligent intrusion detection [6]. ML algorithms such as support vector machine (SVM), random forest (RF), and gradient boosting (GB) have been widely used for IDS development because of their low computational complexity and suitability for small datasets. Although these methods perform well for known attacks, but they are not much scalable. Also, their performance is often affected by concept drift, where changes in traffic behavior over time reduce the effectiveness of previously learned patterns. On the other hand, DL models [7] such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks have shown promising performance for intelligent intrusion detection because they automatically extract complex traffic characteristics from network flow data. Furthermore, hybrid learning architectures such as CNN-recurrent models [8] have also been developed to jointly learn spatial and temporal patterns for improving detection performance and adaptability toward evolving attack behavior. However, several issues still remain unresolved, and one of the significant challenges is the class imbalance problem during model training. In real-world IoT environments, network traffic usually follows a long-tailed distribution

Corresponding author: Sowmya Somanath (e-mail: sowmyasadish@gmail.com).

where the normal traffic samples are extremely higher than the attack samples. Among attack scenarios, DDoS attack samples also dominate the training data, which further increases the imbalance among traffic categories. To address this issue, many existing studies have adopted synthetic oversampling techniques, but it often distorts the original network characteristics and increases the possibility of overfitting. Another challenge is computational complexity, as many IDS frameworks are based on deep architectures with high training and inference costs, which limit their deployment in resource-constrained IoT environments. Another limitation is that traffic features are commonly processed as unordered vectors, where meaningful relationships among protocol, packet, flow, and temporal characteristics are often ignored during feature learning.

Therefore, it becomes important to develop an intrusion detection framework that can preserve semantic relationships in input traffic data, improve minority attack learning, and reduce computational complexity.

This manuscript presents a Traffic-Aware Imbalance Learning Network (TAIL-Net) for lightweight IoT intrusion detection. The proposed framework formulates intrusion detection as a supervised multi-class traffic classification task and incorporates a severity assessment module as a post-prediction alert-prioritization module. Unlike conventional approaches that depend on arbitrary feature ordering, TAIL-Net incorporates imbalance awareness directly into representation learning to improve its minority attack detection capability while maintaining efficient runtime inference. The main contributions are highlighted as follows:

- A semantic feature mapping mechanism is proposed to organize traffic attributes according to their semantic relationships before they are fed to the learning model.
- A lightweight hybrid DL architecture is designed by integrating semantic-aware convolutional learning with a compact recurrent projection module for efficient traffic pattern learning.
- A minority-aware attention mechanism is introduced within the recurrent projection module to improve feature discrimination for underrepresented attack classes.
- Adaptive Class-Balanced Focal Loss (ACB-FL) is proposed to dynamically adjust optimization behavior according to class imbalance severity, for improving minority attack detection without generating synthetic traffic samples.

The remaining sections of the manuscript are organized as follows. Section II presents a brief literature review and identifies gaps; Section III discusses the proposed system design and methodology implementation in detail. Section IV discusses the experimental outcomes, and finally, Section V presents the conclusion and future scope of the research work.

II. RELATED WORK

In the literature, based IDSs have been widely reported for IoT security due to their computational efficiency and suitability for limited-resource devices, but their effectiveness often depends on the quality of features used. A similar analysis was conducted by Altulaihan *et al.* [9], where an anomaly-based IDS is presented using decision tree (DT), RF, K-nearest neighbor (KNN), and SVM classifiers together with feature selection strategies. In the same direction, Albulayhi *et al.* [10] combined entropy-based feature selection and multiple ML classifiers to improve IDS efficiency, whereas Douiba *et al.* [11] utilized GB and CatBoost-based learning algorithms for anomaly detection over multiple IoT datasets. These studies demonstrated that ML algorithms can achieve satisfactory intrusion detection performance with relatively low computational overhead. In addition, feature selection approaches can reduce redundant traffic attributes and improve learning efficiency. However, most of these methods depended on handcrafted features and manually selected traffic attributes; therefore, their capability to learn complex traffic behavior and adapt to evolving attack patterns remained limited.

Several researchers adopted DL models for automatic feature extraction and robust intrusion detection. The work presented by Khan *et al.* [12] suggested a DL framework based on deep neural network (DNN) and BiLSTM models, whereas Jain *et al.* [13] demonstrated the effectiveness of CNN-LSTM integration for learning spatial and sequential traffic characteristics. There are also many research efforts toward developing an advanced hybrid model, as seen in the work of Logeswari *et al.* [14], Chen and Mir [15], and Kilichev *et al.* [16], who developed hybrid CNN-recurrent frameworks with attention mechanisms, optimization strategies, and ensemble learning techniques. However, all these approaches reported higher detection accuracy by automatically learning complex traffic representations, but at the same time, they ignored the requirement for lightweight deployment. To address this limitation, the work of Wakili *et al.* [17] conducted a comparative investigation of classical, deep, and hybrid learning models and reported that hybrid architectures can provide a favorable balance between detection accuracy and computational efficiency. Similarly, Alayash *et al.* [18] evaluated recurrent learning models for IoT attack detection and showed that model performance remains highly dependent on traffic characteristics and attack distributions. Although these studies improved deployment feasibility, many frameworks still depended on complex architectures with considerable parameter overhead, memory consumption, and higher inference costs. Another significant problem considered by researchers is the class imbalance issues in the training dataset. Many works addressed this problem through data-level balancing techniques; for example, the work reported by Duraibi and Alashjaee [19], Hussien *et al.* [20], Widodo *et al.* [21], and Abdelkhalek and Mashaly [22] adopted SMOTE, ADASYN, and similar oversampling techniques to improve minority attack detection performance. Similarly, Rao and Babu [23] introduced an imbalanced generative adversarial network (IGAN) to generate synthetic minority samples before classification through a LeNet5-LSTM ensemble model. Gan Zhu *et al.* [24] further incorporated diffusion models to synthesize minority attack samples and improve CNN-Transformer-based intrusion detection performance. Although these approaches improved class distribution, the generated traffic samples may alter the original network characteristics and introduce additional noise into the learning process and even mimic the deepfake process. In the same research direction, the researchers in the studies of Mirsadeghi *et al.* [25] and Paray and Aydos [26] reported that balancing strategies often remain dependent on traffic distributions and deployment conditions. Therefore, consistent minority attack learning remains a challenging research problem.

Based on the analysis of the existing literature, it has been identified that the majority of the works have concentrated on feature selection, hybrid DL architectures, and data-level balancing techniques. It has also been observed that many works lack novelty in their implementation and report higher accuracy by changing the architecture and performing hyperparameter tuning. The first issue identified is that most CNN-based IDSs process traffic features according to their original dataset ordering, where semantic relationships among different traffic attributes are not explicitly preserved during feature extraction.

Another problem observed is class imbalance, which is commonly handled through data sampling methods that synthetically generate samples from existing samples, rather than through representation learning and optimization mechanisms. A third issue noticed is that there are few researches who have focused on reducing model complexity, which, though it reduces training cost to some extent, still exhibits considerable inference cost. On the other hand, the proposed TAIL-Net model addresses these research issues by integrating semantic traffic feature mapping, lightweight hybrid learning, minority-aware attention, and ACB-FL-based optimization within an integrated intrusion detection framework.

III. PROPOSED METHODOLOGY

This section presents the system design, modeling, and implementation details of the proposed TAIL-Net methodology, followed by the dataset description, preprocessing operations, and model training strategy.

A. PROPOSED SYSTEM

The primary aim of this paper is to develop a robust predictive model, named TAIL-Net, for supervised multi-class IoT intrusion detection. In this study, the model focuses on the detection and classification of five traffic categories, consisting of one Normal class and four attack classes, namely DDoS, DoS, Reconnaissance, and Theft, which are among the most common and high-impact threats in IoT environments. In addition to traffic classification, the proposed framework introduces a rule-based post-prediction threat severity assessment approach as a decision-support tool to prioritize detected events and support appropriate mitigation actions. Figure 1 illustrates the proposed security framework, which takes raw network traffic data from the BoT-IoT dataset [27], which is a benchmark dataset containing both normal and malicious network traffic generated from realistic IoT environments.

In this study, 74 BoT-IoT CSV files are used, and a stratified sampling strategy is applied at the subcategory level to preserve the original attack diversity. The processed dataset includes 35 attributes and 3,668,522 samples, out of which only 499 are normal, while the remaining samples belong to attack categories such as DDoS, DoS, Reconnaissance, and Theft. As shown in Table I, the traffic distribution is highly imbalanced, where DDoS and DoS attacks together constitute more than 97% of the total samples. On the other hand, the Normal and Theft classes contain very few traffic samples, and such long-tailed traffic distributions can bias the learning process toward the majority classes.

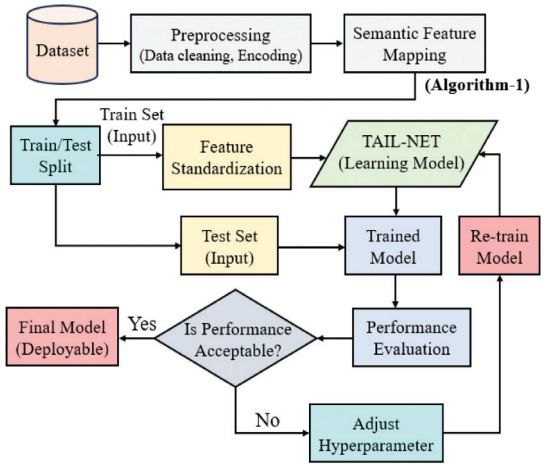

Fig. 1. Proposed IDS framework architecture.

Table 1. Dataset statistics

Class	Count	Percentage
DDoS	~1,926,655	52.5%
DoS	~1,650,269	45.0%
Recon	~91,010	2.5%
Normal	~499	0.01%
Theft	~89	0.001%

The next stage performs data preprocessing, which includes invalid-value handling, removal of irrelevant features, and conversion of categorical attributes into numerical representations. First, non-predictive identifiers and raw address-related fields, namely pkSeqID, stime, saddr, daddr, ltime, smac, dmac, attack, category, and subcategory, are removed from the input feature space. After this removal, the 25 traffic-related features, as listed in Table II, are retained for learning. Invalid and missing values are handled before feature scaling. All infinite values are replaced with missing values, and missing entries are removed or cleaned so that no Not a Number (NaN) values remain in the final feature matrix. Categorical traffic attributes are converted into numerical form using the label encoding method. The final input feature matrix therefore has a dimension of $3{,}668{,}522 \times 25$, and the corresponding label vector contains 3,668,522 class labels. After preprocessing and semantic feature ordering, Z-score standardization is applied using the training data statistics, and the same scaler is then used to transform the test data to prevent information leakage. Finally, the standardized feature matrix is reshaped into a three-dimensional input representation for the proposed one-dimensional convolutional learning model. Table II highlights the preprocessing operations carried out in this study.

Afterward, a traffic-aware semantic feature mapping mechanism is applied to reorganize traffic features according to their semantic relationships. The implementation procedure of the proposed semantic feature mapping is described in Algorithm 1,

Table II. Dataset preprocessing

Preprocessing step	Description
Total samples	3,668,522
Original attributes	35 columns
Classification labels	DDoS, DoS, Normal, Reconnaissance, Theft
Removed attributes	pkSeqID, stime, saddr, daddr, ltime, smac, dmac, attack, category, subcategory
Retained features	flgs, proto, sport, dport, pkts, bytes, state, seq, dur, mean, stddev, sum, min, max, soui, doui, sco, dco, spkts, dpkts, sbytes, dbytes, rate, srate, drate
Categorical encoding	Label encoding
Invalid-value handling	Infinite and missing values cleaned; final NaN count = 0
Normalization	Z-score standardization using training-set statistics
Final feature dimension	25
Deep-learning input shape	samples × 25 × 1

Algorithm 1. Semantic Feature Mapping

Input: Feature set F
Output: Semantically ordered feature set Fs
Start
1: Initialize $F_s \leftarrow \varnothing$
2: Define semantic groups:
G1 ← Protocol and Connection Features
G2 ← Packet Behaviour Features
G3 ← Byte Behaviour Features
G4 ← Rate Behaviour Features
G5 ← Temporal Flow Features
3: for each group G_i do
4: for each feature $f_j \in G_i$ do
5: $F_s \leftarrow F_s \cup F_j$
6: end for
7: end for
8: for each feature $f_k \in F$ do
9: if $f_k \notin F_s$ then
10: $F_s \leftarrow F_s \cup f_k$
11: end if
12: end for
13: return F_s
End

which preserves locality among related protocol, packet, byte, rate, and temporal traffic characteristics.

Algorithm 1 presents the semantic feature mapping procedure adopted in this study. The algorithm considers the preprocessed traffic feature set F as its input and returns a semantically organized feature set F_s. Initially, an empty feature container F_s is created (Step 1). Afterward, the traffic attributes are categorized into five predefined semantic groups, namely protocol and connection features, packet behavior features, byte behavior features, rate behavior features, and temporal flow features (Step 2). The algorithm then traverses each semantic group and sequentially appends the corresponding features to F_s according to the predefined traffic-aware ordering (Steps 3–7). Subsequently, any remaining feature that does not belong to the predefined groups is appended to the end of the feature sequence to preserve information completeness (Steps 8–12). Finally, the semantically ordered feature set F_s is returned (Step 13). This procedure preserves the locality among related traffic attributes before they are provided to the learning model, thereby enabling more meaningful local feature extraction during convolutional learning.

Figure 2 illustrates the semantic feature mapping process adopted in the proposed TAIL-Net framework. Figure 2(a) shows the original feature arrangement, where traffic attributes appear in an unstructured manner according to the original dataset representation. The issue with this arrangement is that semantically related traffic attributes remain separated from each other; therefore, local feature extraction during the convolutional learning phase becomes less effective. Algorithm 1 shows the proposed idea of reorganizing traffic attributes according to their semantic relationships. As shown in Fig. 2(b), related features are arranged into locality-preserving groups, including protocol and connection features, namely proto, state, flgs, soui, doui, sco, and dco; packet behavior features, namely pkts, spkts, and dpkts; byte behavior features, namely bytes, sbytes, and dbytes; rate behavior features, namely rate, srate, and drate; temporal flow features, namely dur, mean, stddev, sum, min, and max; and other retained attributes, namely sport, dport, and seq. This ordering is traffic-aware because it follows the functional meaning of network-flow attributes and places related traffic characteristics in adjacent positions before one-dimensional convolutional learning. As a result, semantically related traffic characteristics become adjacent, and when provided as input to the TAIL-Net model, they improve local feature learning and extraction.

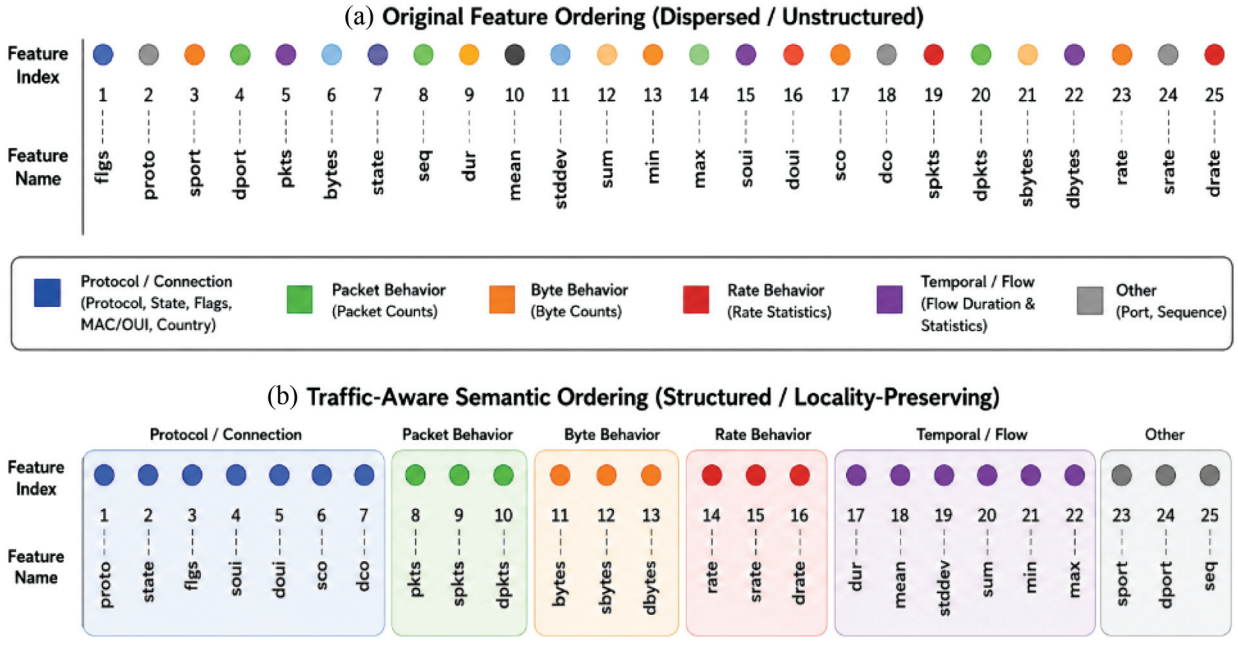

Fig. 2. Semantic traffic feature mapping adopted in TAIL-Net: (a) original dataset feature arrangement, and (b) semantically organized feature arrangement based on traffic behavior relationships.

After semantic feature mapping, the dataset is divided into training and testing subsets, where 80% of the samples are used for model training and the remaining 20% are kept for testing. Once the training process is completed, the trained model is evaluated using the test dataset to compute performance metrics such as accuracy, precision, recall, and F1-score. The obtained results are then compared against predefined performance requirements. If the achieved performance satisfies the desired criteria, the trained model is accepted as the final deployable IDS. Otherwise, the model hyperparameters are adjusted and the training process is repeated. This iterative optimization procedure continues until satisfactory intrusion detection performance is achieved. Table III presents the statistics of the class-wise dataset split operation.

B. PROPOSED TAIL-Net MODEL

The proposed TAIL-Net is designed with the objective of offering a lightweight and robust learning model for detecting and classifying intrusions in highly dynamic and complex IoT ecosystem. Figure 3 shows the architecture of the proposed TAIL-Net model. The proposed model takes semantically organized and standardized traffic features as its input. The first module of TAIL-Net is the semantic projection layer, where a one-dimensional (1D) convolution (Conv) operation with 32 filters and a kernel size of 1 transforms the input traffic sequence into a richer feature representation.

Afterward, layer normalization (LayerNorm) and Gaussian Error Linear Unit (GELU) activation are applied to stabilize the learning process and introduce non-linearity into the network. The semantic projection layer produces an output representation of 25×32. The projected feature representation is then passed to the convolutional learning module, which consists of two CNN blocks. CNN block 1 contains a one-dimensional convolution with 64 filters and a kernel size of 3, followed by layer normalization and GELU activation. It learns local traffic patterns from semantically grouped features and produces an output representation of 25×64. CNN block 2 follows a similar structure with 128 filters and kernel size of 3, which further improves local feature abstraction and generates an output representation of 25×128. In this way, the convolutional module learns local traffic behavior among related protocol, packet, byte, rate, and temporal attributes with a simple and computationally efficient design.

Table III. Class-wise training and testing split

Class	Training samples	Testing samples
DDoS	1,541,324	385,331
DoS	1,320,215	330,054
Reconnaissance	72,808	18,202
Normal	399	100
Theft	71	18
Total	2,934,817	733,705

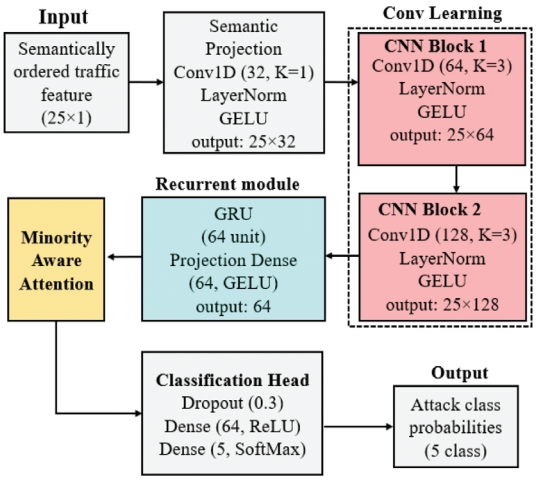

Fig. 3. Proposed TAIL-Net architecture.

The output from CNN Block 2 is then provided to the recurrent learning module. In this study, a Gated Recurrent Unit (GRU) [28] is adopted because it requires fewer parameters and lower memory usage, thereby supporting faster inference and reduced computational overhead. The GRU has 64 neurons which learns sequential dependency relationships among traffic features. The model learns both local feature interactions from the CNN module and longer-range dependency patterns through recurrent learning. Afterward, a projection dense layer with 64 neurons and GELU activation is incorporated to transform the recurrent output into a compact feature vector. This projection further refines the learned representation and preserves useful intrusion-related features before they are passed to the minority-aware attention module developed to address the class-imbalance problem in its learning process. Table IV presents the input, intermediate, and output tensor shapes of each major TAIL-Net module.

1). MINORITY-AWARE ATTENTION. The proposed minority-aware attention is not a traditional self-attention mechanism like transformer attention. It estimates the imbalance severity of each class from the training data, learns a preliminary class distribution from the recurrent feature vector, computes a minority-risk score using the class-imbalance prior, and uses this minority-risk information to generate an attention gate. It reweights the recurrent features so that features associated with minority classes receive greater emphasis. The implementation process is described in Algorithm 2. In the initial step of the algorithm, the class frequency of each attack category is computed from the training **dataset (Step1).**

Afterward, a minority prior is assigned to each class according to its relative occurrence frequency (Steps3–5). Let n_c denote the number of samples belonging to class c. The minority prior m_c is computed as in Eq. (1):

$$m_c = 1 - \frac{n_c}{n_{max}} \tag{1}$$

where n_c is the number of samples in class, n_{max} represents the number of samples in the majority class, and m_c denotes the minority prior associated with class. Table V shows the computed minority prior values.

As shown in Table IV, the majority attack class receives a prior value close to zero, and minority attack categories receive larger prior values, that is, higher severity of imbalance and greater importance is needed for minority classes during the attention generation. Based on the input H_r, an initial class probability distribution P is generated (Step 6) using a fully connected transformation followed by the SoftMax activation function as in Eq. (2), where W_p and b_p represent trainable parameters. Here, H_r denotes the recurrent feature representation obtained from the GRU projection layer, and P denotes the preliminary class-probability vector over the C traffic classes. Then, the expected minority risk is estimated using Eq. (3), where R_m denotes the probability-weighted

Table IV. Layer wise configuration details of the proposed TAIL-Net architecture

Module	Operation	Input shape	Output shape
Input layer	Semantically ordered traffic feature vector	25×1	25×1
Semantic projection layer	Conv1D with 32 filters, kernel size 1, LayerNorm, GELU	25×1	25×32
CNN Block 1	Conv1D with 64 filters, kernel size 3, LayerNorm, GELU	25×32	25×64
CNN Block 2	Conv1D with 128 filters, kernel size 3, LayerNorm, GELU	25×64	25×128
Recurrent module	GRU with 64 units	25×128	64
Recurrent projection	Dense layer with 64 neurons and GELU activation	64	64
Preliminary class distribution	Dense layer with SoftMax activation	64	5
Minority-risk estimation	Probability-weighted minority-risk computation	5	1
Attention concatenation	Concatenation of recurrent representation and minority-risk score	$64 + 1$	65
Attention gate	Dense layer with sigmoid activation	65	64
Minority-aware representation	Element-wise multiplication between attention gate and recurrent representation	64	64
Classification head	Dropout 0.3 and Dense layer with ReLU activation	64	64
Output layer	Dense layer with SoftMax activation	64	5

Algorithm 2. Minority-Aware Attention

Input: Recurrent feature representation H_r, class labels Y and number of classes C

Output: Minority-aware representation H_m

Start

1: Compute class frequency n_c for each c

2: Compute maximum class frequency n_{max}

3: for each class $c = 1$ to C do

4: Compute minority prior m_c from Eq. (1)

5: end for

6: Form minority-prior vector M

7: Generate initial class probability P from Eq. (2)

8: Compute expected minority risk R_m using Eq. (3)

9: Concatenate H_r and R_m: $Z = [H_r; R_m]$

10: Generate attention gate A using Eq. (5)

11: Refine recurrent representation as in Eq. (6)

12: Return H_m

End

Table V. Minority prior values

Class	Count	Minority prior
DDoS	1,541,324	0.0000
DoS	1,320,215	0.1435
Normal	399	0.9997
Recon	72,808	0.9528
Theft	71	0.9999

Table VI. ACB-FL-driven parameters

Class	n_c	α_c	γ_c
DDoS	1,541,324	0.0294	1.5000
DoS	1,320,215	0.0294	1.7152
Recon	72,808	0.0294	2.9291
Normal	399	0.7523	2.9996
Theft	71	4.1594	2.9999

minority risk associated with the current traffic sample (Step 7). In Eq. (3), $M = [m_1, m_2, \ldots, m_C]^T$ denotes the minority-prior vector computed from the training class distribution, where eachm_c represents the minority prior of class c. Then recurrent feature representation and the minority risk score are then concatenated (Step 8) to form a joint representation using Eq. (4):

$$P = Softmax(W_p H_r + b_p) \tag{2}$$

$$R_m = P \cdot M \tag{3}$$

$$Z = [H_r; R_m] \tag{4}$$

Afterward, an attention gate is generated as mentioned in Eq. (5), where $\sigma(\cdot)$ denotes the sigmoid activation function, and W_a and b_a represent trainable parameters (Step 9). The attention gate A has the same feature dimension as H_r and assigns adaptive weights to the recurrent feature components according to the estimated minority-risk information. The attention gate returns an output of precise feature representation via element-wise multiplication as in Eq (6), where $\odot$ denotes the Hadamard product (Step 10–11):

$$A = \sigma(W_a Z + b_a) \tag{5}$$

$$H_m = A \odot H_r \tag{6}$$

Unlike existing attention method that estimate feature importance only from hidden representations, the proposed algorithm also considers class-frequency-derived minority prior information. Therefore, the attention gate becomes sensitive to rare attack classes and refines the recurrent representation for classification layer, consisting of a dropout layer, a dense layer with ReLU activation, and a SoftMax output layer that generates probability scores for traffic categories.

2). ACB-FL LOSS FUNCTION. The effectiveness of any DL model does not depend only on the network architecture but also on the adopted training strategy and optimization mechanism.

Unlike existing cross-entropy loss that treats all classes equally and standard focal loss that employs a fixed focusing parameter, the proposed ACB-FL dynamically adjusts both class weighting and focusing strength according to class imbalance severity. Thus, minority attack classes receive stronger optimization emphasis during training, thereby improving learning under highly imbalanced IoT traffic distributions. The ACB-FL is designed based on two mechanisms viz. i) class-balanced weighting α_c in Eq (7) and ii) adaptive focal learning γ_c defined in Eq (8), where $c$denotes a traffic class, $n_c$denotes the number of training samples belonging to class c, and n_{max} denotes the number of samples in the majority class:

$$\alpha_c = \frac{1-\beta}{1-\beta^{n_c}} \tag{7}$$

where $\beta \in [0,1)$ represents the class-balancing parameter and α_c assigns larger weights to minority classes and smaller weights to majority classes. Further, the computed α_c values are normalized for all the classes before loss computation to maintain stable optimization:

$$\gamma_c = \gamma_0 + \lambda\left(1 - \frac{n_c}{n_{max}}\right) \tag{8}$$

where γ_0 is the base focusing parameter, λ is the imbalance adjustment coefficient, n_c represents the frequency of traffic classes, and n_{max} corresponds to the frequency of the majority class. So, minority classes receive stronger focusing behavior, whereas majority classes receive comparatively lower focusing strength according to imbalance severity. The final ACB-FL objective function is defined as in Eq (9), where p_y denotes the predicted probability associated with the target class, α_y represents the class-balanced weight, and γ_y denotes the adaptive focusing parameter:

$$L_{ACB} = -\alpha_y(1-p_y)^{\gamma_y}\ \log(p_y) \tag{9}$$

Here, y denotes the ground-truth class label of the current training sample, while α_y and γ_y are selected from the class-specific α_c and γ_c values according to y. Therefore, through the combined effect of adaptive weighting and adaptive focusing, the proposed loss function increases the optimization emphasis on underrepresented attack categories. For mini-batch training, the final ACB-FL value is computed as the average of L_{ACB} over all samples in the mini-batch. Table VI presents the weighting factors and focusing parameters generated by the proposed ACB-FL. The class-specific weighting factors and focusing parameters were automatically computed according to the class frequency distribution of the training dataset and are normalized before loss computation to maintain optimization stability.

In this study, the class-balancing parameter β is set to 0.9999 because the BoT-IoT dataset has a large number of traffic samples; choosing a large β value will allow for a stable estimation of class-balanced weights. The base focusing parameter γ_0 and imbalance adjustment coefficient λ were empirically set to 1.5 to achieve a balance between optimization stability and minority attack emphasis. An Adam optimizer is used with a learning rate of 0.0005 for enhanced gradient descent optimization during training. The model is trained using a batch size of 512 over 30 epochs.

C. COMPUTATIONAL COMPLEXITY ANALYSIS

The computational efficiency is an important requirement for IoT IDS because detection models are often deployed on resource-constrained devices. Therefore, TAIL-Net is designed as a compact CNN-GRU architecture that balances detection performance and computational overhead. The convolutional module extracts local traffic patterns from semantically organized features with complexity $O(LKC_{in}C_{out})$, where L is the feature sequence length, K is the kernel size, and C_{in} and C_{out} denote the input and output channels, respectively. For recurrent learning, a GRU is adopted instead of LSTM or their bidirectional variants often used in IDS development. The complexity of the GRU module is $O(TH^2)$, where T denotes the sequence length and H represents the hidden dimension. Since GRU employs fewer gating operations than LSTM, it requires fewer parameters and lower memory consumption. The proposed minority-aware attention operates on the compact recurrent representation rather than the complete traffic sequence, resulting in a small additional cost of $O(HC)$, where C denotes the number of classes. Therefore, the overall complexity of TAIL-Net can be expressed as:

$$O(LKC_{in}C_{out}) + O(TH^2) + O(HC) \tag{10}$$

The computational cost is primarily dominated by the CNN and GRU modules, while semantic feature mapping, minority-aware attention, and ACB-FL introduce only marginal overhead. Therefore, TAIL-Net is suitable for lightweight real-time IoT intrusion detection.

IV. RESULTS AND DISCUSSION

The design and development of the proposed system were carried out using Python programming executed in the Anaconda environment on a Windows machine equipped with an Intel Core i7 processor, 16 GB RAM, and an NVIDIA GeForce RTX 3050 GPU. The TensorFlow framework was utilized for model training and validation. The experimental outcomes and performance were analyzed considering standard metrics such as accuracy, precision, recall and F1-score. Additional analysis was also conducted to justify the selection of the proposed modules and its efficiency.

Figure 4 presents the confusion matrix comparison between the proposed TAIL-Net and a baseline CNN-GRU model trained using SMOTE-balanced traffic data. In order to carry out fair performance evaluation, the CNN-GRU model was configured using the same training settings that were used for TAIL-Net, including a learning rate (0.0005), batch size (512), and 30 epochs. The baseline model did not incorporate semantic feature mapping, minority-aware attention, or the proposed ACB-FL loss function; instead, it uses categorical cross-entropy (CCE).

As shown in Fig. 4(a), the CNN-GRU-SMOTE model achieved strong performance for DDoS and DoS traffic classes. However, some misclassifications happened for the minority classes, that is, Normal and Theft. On the other hand, Fig. 4(b) demonstrated that the proposed TAIL-Net maintained comparable performance for majority classes, whereas it improved the discrimination ability for underrepresented traffic classes. The number of correctly classified Normal samples increased from 94 to 97, Theft samples increased from 17 to 18, and Reconnaissance samples increased from 18,075 to 18,101. The quantitative comparison presented in Table VII further supports these observations. It can be seen that both models have achieved precision, recall, and F1-score values close to 0.99 for DDoS and DoS attacks. For the minority classes, the precision increased from 0.71 to 0.85, and the F1-score improved from 0.82 to 0.91 for the Normal class. Similarly, for the highly underrepresented Theft class, precision increased from 0.30 to 0.47, while the F1-score improved from 0.46 to 0.64. Both models maintained a recall value of 1.00 for Theft attacks. Although SMOTE improved class balance during training, the

(a) CNN-GRU + SMOTE (b) Proposed TAIL-Net

Fig. 4. Confusion matrix analysis (a) CNN-GRU + SMOTE. (b) Proposed TAIL-Net.

Table VII. Class-wise performance comparison

	CNN-GRU + SMOTE			TAIL-Net precision		
Class	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
DDoS	0.99	0.99	0.99	0.99	0.99	0.99
DoS	0.99	0.98	0.99	0.99	0.99	0.99
Normal	0.71	0.96	0.82	0.85	0.97	0.91
Reconnaissance	0.99	1.00	1.00	1.00	0.99	1.00
Theft	0.30	1.00	0.46	0.47	1.00	0.64

generated samples might not always have accurately represented evolving network traffic behavior in real-world IoT environments. The proposed TAIL-Net preserved the original traffic distribution and addressed class imbalance through semantic feature organization, minority-aware attention, and ACB-FL. Thus, the proposed framework achieved improved minority attack discrimination without depending on synthetic traffic generation.

To evaluate the generalization capability of the proposed TAIL-Net, a five-fold stratified cross-validation experiment was conducted. The model achieved an average accuracy of 99.18% ± 0.24%, a weighted F1-score of 99.19% ± 0.23%, a macro recall of 98.21% ± 0.61%, a macro F1-score of 85.06% ± 1.92%, and a macro-AUC of 0.9996 ± 0.0002. The low standard deviation for all the folds indicated that the proposed model maintained stable performance under different training-validation partitions. In addition, the high macro recall showed that TAIL-Net preserves strong detection sensitivity for minority attack categories, which was important under highly imbalanced IoT traffic distributions. Figure 5 presents the multi-class Receiver Operating Characteristic (ROC) curve analysis of the proposed TAIL-Net.

It can be seen that the curves are located close to the upper-left corner, which indicates effective class separability. The model achieved AUC values of 0.9996 for DDoS and DoS, and 1.0000 for Normal, Reconnaissance, and Theft. Table VIII presents the per-class precision–recall AUC of the proposed TAIL-Net. The model achieved very high PR-AUC values for DDoS, DoS, Reconnaissance, and Theft, indicating strong detection capability even for minority attack classes. The Normal class achieved a slightly lower PR-AUC of 0.962, suggesting some overlap between normal and attack traffic patterns. The macro-average PR-AUC of 0.9920 indicated that TAIL-Net maintained robust precision–recall performance under imbalanced traffic conditions.

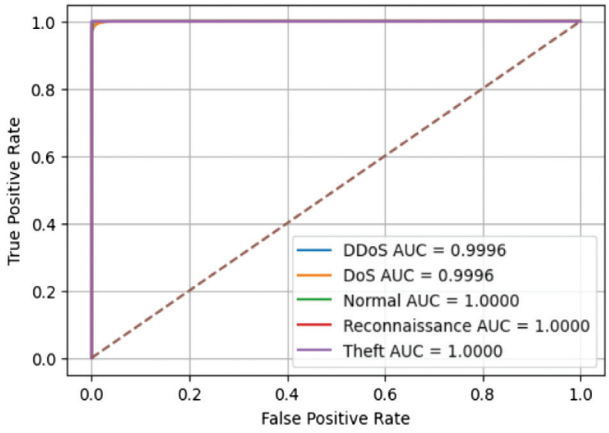

Fig. 5. ROC curve analysis.

Table VIII. Per-class precision–recall AUC

Class	**Proposed PR-AUC**
DDoS	0.9997
DoS	0.9996
Normal	0.962
Reconnaissance	0.9999
Theft	0.9968
Macro Average	0.9920

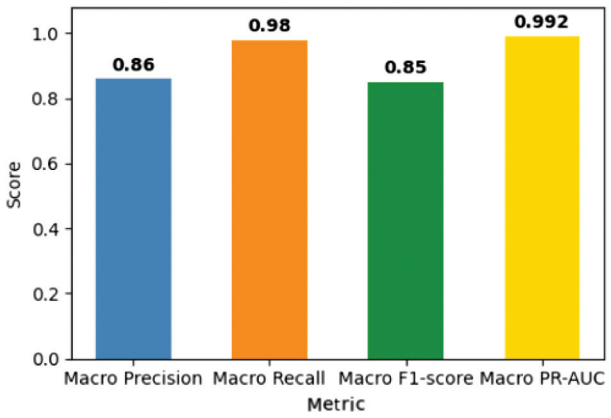

Fig. 6. Analysis of imbalance-sensitive evaluation.

Figure 6 presents the imbalance-sensitive evaluation of the proposed TAIL-Net, which demonstrated strong detection performance under imbalanced traffic conditions. The high macro recall indicated that minority attack samples were detected effectively, whereas the macro-PR-AUC confirmed stable precision–recall performance across classes.

Also, the model exhibited a Macro One-vs-Rest FAR of 0.36%, which represents the average false alarm rate when each class is evaluated separately against all other classes. On the other hand, the Operational FAR was 3.00%, which indicates the practical false alarm rate under deployment-style normal-versus-attack decision conditions. Thus, the macro FAR showed low class-wise false-positive tendency, while the operational FAR indicated the expected false alarm level in real usage.

A. COMPUTATIONAL EFFICIENCY ANALYSIS

Table IX presents the quantitative analysis of computational efficiency comparison of the proposed TAIL-Net against various existing DL models. The analysis considers accuracy, Macro F1-score, and Macro AUC for evaluating classification performance and class discrimination capability, whereas the number of trainable parameters, model size, and throughput are considered for complexity and runtime efficiency analysis. It can be observed that standalone recurrent models such as LSTM and GRU achieved lower detection performance, whereas CNN improved computational efficiency but exhibited limited capability in learning complex temporal traffic behavior. The hybrid architectures such as CNN-LSTM, CNN-GRU, and CNN-BiLSTM achieved better detection performance by jointly learning spatial and temporal traffic characteristics, although this improvement is associated with increased model complexity.

Among different DL models, CNN-BiLSTM achieved good detection performance; however, it contains the highest number of trainable parameters and largest model size, thereby increasing computational overhead. The proposed TAIL-Net achieved the highest accuracy (99%) and Macro F1-score (0.85) with only 81,866 trainable parameters and a model size of 0.3122 MB. When compared to CNN-BiLSTM, the proposed model has less parameter count and model size by approximately 41% while improving classification performance. Furthermore, TAIL-Net achieved a throughput of 245,623 samples/sec, which is higher than CNN-LSTM and CNN-BiLSTM, which indicated that the proposed model has improved runtime efficiency. The obtained results demonstrate that the proposed architecture achieved a better trade-off between detection performance and computational efficiency.

Figure 7 presents the training time comparison of the evaluated DL models. As expected, standalone models such as CNN, GRU, and LSTM took very training time due to their simpler architectures. The hybrid models are associated with higher training costs because they combined convolutional and recurrent learning mechanisms. Among the hybrid approaches, CNN-BiLSTM took longer training time (1,198.66 s), whereas the proposed TAIL-Net requires 1,111.34 s. Although TAIL-Net introduced semantic feature mapping, minority-aware attention, and ACB-FL, its training time is still lower than CNN-BiLSTM while delivering effective performance. Figure 8 compares the average inference latency of the different DL models.

It can be seen that CNN achieved the lowest inference time (0.0027 ms/sample), followed by GRU and LSTM. The proposed TAIL-Net showed an average inference time of 0.0040 ms/sample, which was identical to CNN-LSTM but lower than that of

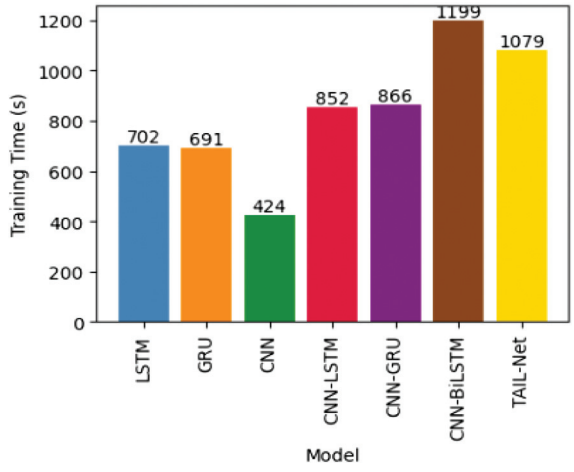

Fig. 7. Training time analysis.

Table IX. Comparative analysis for detection performance and computational efficiency

Model	Accuracy	Macro F1	Macro AUC	Parameters	Model size (MB)	Throughput (samples/s)
LSTM	0.92	0.72	0.99	21,381	0.0816	311,982.48
GRU	0.93	0.68	0.99	17,349	0.0662	330,857.16
CNN	0.93	0.74	0.98	40,005	0.1562	367,351.86
CNN-LSTM	0.97	0.76	0.99	85,317	0.3255	224,315.88
CNN-GRU	0.97	0.75	0.99	73,157	0.2791	262,483.80
CNN-BiLSTM	0.98	0.79	0.99	138,821	0.5296	166,228.23
TAIL-Net	0.99	0.85	0.99	81,866	0.3122	245,623.88

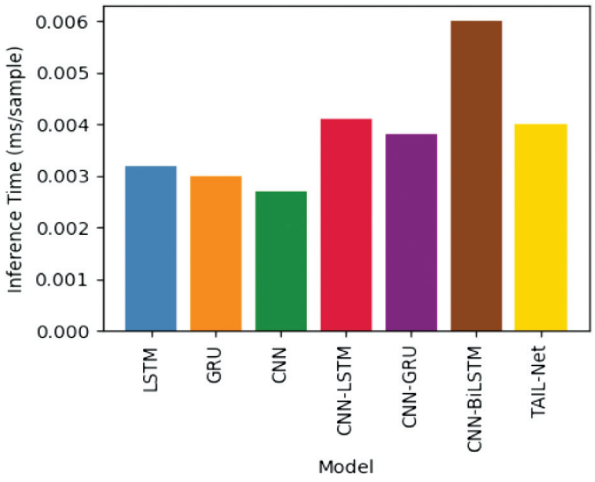

Fig. 8. Inference time (runtime) analysis.

CNN-BiLSTM (0.0060 ms/sample). To further assess deployment feasibility, CPU-only inference was also measured, where TAIL-Net achieved an inference latency of 0.195 ms/sample and a throughput of 5129.73 samples/s. Therefore, despite incorporating additional learning components, TAIL-Net maintained low prediction latency on both GPU and CPU, thereby supporting its suitability for real-time or near-real-time IoT intrusion detection.

Table X presents the bootstrap-based 95% confidence intervals for the proposed TAIL-Net. The quantified outcome demonstrated that the model maintained stable performance under resampled test conditions. The macro-PR-AUC is consistently high, with a confidence interval of 0.986–0.997, which confirmed its reliable performance for imbalanced traffic classification. The operational FAR had a mean of 0.030, with a wider interval of 0.010–0.060, thereby indicating that false alarms may vary depending on the sampled traffic distribution but remain within an acceptable range.

Table X. Analysis of confidence intervals

Metric	Mean	95% confidence interval
Macro precision	0.86	0.84–0.89
Macro recall	0.98	0.97–0.99
Macro F1-score	0.85	0.83–0.88
Macro PR-AUC	0.992	0.986–0.997
Operational FAR	0.030	0.010–0.060

B. ABLATION STUDY

Table XI presents the ablation study to analyze the effectiveness of different components used in the TAIL-Net. The inclusion of semantic feature mapping improved the Macro F1-score from 0.68 to 0.74, which indicated that semantic grouping of related traffic attributes enhanced local feature extraction. The CNN-GRU configuration has further improved performance by capturing dependency relationships among learned traffic representations. When semantic mapping was combined with GRU, the accuracy increases to 0.984 and the Macro F1-score improves to 0.81 that demonstrated the complementary effect of semantic locality and recurrent learning.

The addition of minority-aware attention further improved the Macro F1-score to 0.83, which indicated that minority-sensitive feature refinement underrepresented attack detection. The full TAIL-Net achieved the best performance with an accuracy of 0.992, a Macro F1-score of 0.85, and a Macro AUC of 0.999. The component removal analysis also confirmed the contribution of each module, as removing semantic mapping, minority-aware attention, or ACB-FL reduced the overall performance. Therefore, the ablation results demonstrated that the proposed components jointly improved traffic representation learning and imbalance-aware intrusion detection. Further analysis was carried out in Table XII to compare different imbalance-handling strategies. The basic CNN-GRU with cross-entropy achieved an accuracy of 0.972 and a macro F1-score of 0.760, while SMOTE improved the macro F1-score to 0.790. Loss-based methods provided further gains, with class-weighted CE, focal loss, and class-balanced focal loss increasing the macro F1-score to 0.800, 0.810, and 0.830, respectively. The proposed TAIL-Net achieved the best results with an accuracy of 0.992, a macro F1-score of 0.850, and a macro-AUC of 0.999 using the original data, which demonstrated that ACB-FL with minority-aware attention improved imbalance handling.

C. COMPARATIVE ANALYSIS

Table XIII presents the comparative analysis of the proposed TAIL-Net with existing similar works on BoT-IoT-dataset [29–33]. It can be observed that several existing approaches reported very high detection accuracy and F1-score values through the use of SMOTE-based balancing, ensemble learning, feature selection, and complex hybrid architectures. Although such approaches

Table XI. Ablation study

Configuration	Semantic mapping	GRU	Minority attention	ACB-FL	Accuracy	Macro F1	Macro AUC
Baseline CNN	✗	✗	✗	✗	0.952	0.68	0.984
CNN + Semantic Mapping	✓	✗	✗	✗	0.970	0.74	0.989
CNN-GRU	✗	✓	✗	✗	0.972	0.76	0.991
CNN-GRU + Semantic Mapping	✓	✓	✗	✗	0.984	0.81	0.996
CNN-GRU + Semantic Mapping + Minority Attention	✓	✓	✓	✗	0.989	0.83	0.998
Full TAIL-Net	✓	✓	✓	✓	0.992	0.85	0.999
Full TAIL-Net without Semantic Mapping	✗	✓	✓	✓	0.982	0.80	0.996
Full TAIL-Net without Minority Attention	✓	✓	✗	✓	0.987	0.82	0.997
Full TAIL-Net without ACB-FL	✓	✓	✓	✗	0.989	0.83	0.998

Table XII. Analysis of additional imbalance-handling comparison

Configuration	Balancing	Loss function	Accuracy	Macro F1	Macro AUC
CNN-GRU	None	Cross-Entropy	0.972	0.760	0.991
CNN-GRU	SMOTE	Cross-Entropy	0.978	0.790	0.994
CNN-GRU	None	Class-Weighted CE	0.980	0.800	0.995
CNN-GRU	None	Focal Loss	0.984	0.810	0.996
CNN-GRU	None	Class-Balanced Focal Loss	0.988	0.830	0.998
Proposed TAIL-Net	Original Data	ACB-FL + Minority Attention	0.992	0.850	0.999

Table XIII. Comparative analysis with existing work on BoT-IoT dataset

Performance factor	Existing studies [29–33]	Proposed TAIL-Net
Detection accuracy	99%–99.9%	99%
F1-score	99.0%–99.9%	99%
Dataset distribution	Mostly SMOTE-balanced traffic	Original highly imbalanced traffic
Class-imbalance handling	SMOTE, weighted loss functions	Minority-aware attention + ACB-FL
Synthetic data dependency	Required in most studies	Not required
Feature learning	Conventional feature selection or raw features	Traffic-aware semantic feature mapping
Model architecture	CNN-LSTM, BiLSTM, ensemble and federated models	Lightweight CNN-GRU architecture
Model complexity	Often unreported or relatively high	81,866 parameters
Model size	Generally, not reported	0.3122 MB
Inference efficiency	Limited reporting	0.0041 ms/sample
Deployment suitability	Moderate to high computational requirements	Suitable for real-time IoT deployment

improved classification performance, they often increased model complexity and altered the original traffic distribution through synthetic sample generation. On the other hand, the proposed TAIL-Net achieved comparable detection performance on the original highly imbalanced traffic distribution. Instead of depending on the synthetic oversampling, the proposed framework addressed class imbalance through minority-aware attention and ACB-FL. Furthermore, traffic-aware semantic feature mapping was incorporated to preserve semantic locality during feature learning. From a deployment perspective, TAIL-Net maintained a lightweight architecture with only 81,866 trainable parameters and a model size of 0.3122 MB while achieving an inference latency of 0.0041 ms/sample. Therefore, the obtained results indicated that the proposed framework provided an effective balance between detection performance, computational efficiency, and practical deployment suitability for real-time IoT intrusion detection environment.

D. DISCUSSION

The experimental results have demonstrated that the proposed TAIL-Net addressed three core challenges in IoT intrusion detection, namely feature representation learning, class imbalance, and lightweight deployment.

The outcome analysis validates the effectiveness of the proposed model and as observed from a computational perspective in Table IX, the TAIL-Net maintained a lightweight architecture with only 81,866 trainable parameters, and it achieved a favorable balance between detection capability and runtime cost, thereby making it suitable for resource-constrained IoT deployment scenarios. Apart from attack classification, the proposed framework also provided a practical mechanism for risk prioritization. Since the BoT-IoT dataset does not provide explicit threat severity annotations, severity estimation was formulated as a rule-based post-classification decision-support process rather than a supervised learning task. The severity score was computed using attack

Table XIV. Rule-based threat severity assessment for risk prioritization

Predicted class	Base risk R(C)	Confidence condition Pmax	Traffic intensity condition I	Severity score rule	Assigned severity
Normal	0.00	Any	Any	$S = 0$	No threat
Reconnaissance	0.40	Low to moderate	Low to moderate	$S = 0.50R(C) + 0.30Pmax + 0.20I$	Medium
Reconnaissance	0.40	High	High	$S = 0.50R(C) + 0.30Pmax + 0.20I$	High
DoS	0.75	Moderate to high	Moderate	$S = 0.50R(C) + 0.30Pmax + 0.20I$	High
DoS	0.75	High	High	$S = 0.50R(C) + 0.30Pmax + 0.20I$	Critical
DDoS	0.90	Moderate to high	Moderate to high	$S = 0.50R(C) + 0.30Pmax + 0.20I$	Critical
Theft	0.85	Moderate to high	Any	$S = 0.50R(C) + 0.30Pmax + 0.20I$	Critical

category risk, model confidence, and traffic intensity indicators. Based on the computed score, detected events are categorized into qualitative severity levels including No Threat, Medium, High, and Critical.

Table XIV presents the adopted threat severity assessment strategy. As shown in Table XIV, DDoS and Theft attacks were assigned higher severity levels due to their potentially significant impact on service availability and data security, whereas Reconnaissance activities were assigned comparatively lower risk levels. Consequently, the proposed framework can assist security administrators not only in identifying attack categories but also in prioritizing security alerts according to their potential operational impact.

Despite the encouraging results, the proposed study was evaluated using offline traffic data from the BoT-IoT dataset. Moreover, the threat severity module was rule-based due to the absence of ground-truth severity labels; therefore, the generated severity levels should be interpreted as risk-prioritization indicators rather than independently validated severity predictions. Future research will focus on integrating concept drift monitoring, online model adaptation, supervised severity annotation, and automated response generation.

V. CONCLUSION

This paper presented TAIL-Net, a lightweight and imbalance-aware intrusion detection framework for IoT environments. The proposed framework integrated semantic traffic feature mapping, CNN-GRU-based feature learning, minority-aware attention, and ACB-FL to enhance intrusion detection under highly imbalanced traffic distributions without relying on synthetic oversampling techniques. The obtained results demonstrated that the proposed architecture effectively improved minority attack learning while maintaining computational efficiency suitable for practical IoT deployment. Furthermore, the proposed framework incorporated a rule-based threat severity assessment mechanism to support alert prioritization and security decision-making. The future work will focus on concept drift adaptation, online learning mechanisms, severity-aware supervised modelling, and real-world IoT deployment for continuous cyber threat monitoring and response.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] H. M. Ibrahim, "Artificial Intelligence (AI) and Internet of Things (IoT) applications in smart cities: Literature review," *Iraqi J. for Comput. Inform.*, vol. 52, no. 1, pp. 35–53, 2026.

[2] S. S. Mangalampalli, P. Choppara, and M. Ijaz Khan, "Introduction to IoT and IIoT security," in *Edge Intelligence: Advanced Deep Transfer Learning for IoT Security*, J. Ahmad, S. Latif, W. Boulila, A. Koubaa, M. Ur Rehman, and I. Ullah Khan, Eds. Cambridge, MA, USA: Syngress (Elsevier), 2026, pp. 1–16, doi: 10.1016/B978-0-44-338297-0.00007-6.

[3] S. Shakya, R. Abbas, and S. Maric, "A Novel Zero-Touch, Zero-Trust, AI/ML enablement framework for IoT network security," *arXiv preprint arXiv:2502.03614*, 2025.

[4] S. Lakshminarayana and P. S. Thilagam, "Next-generation DDoS attacks on IoT deployments: Targeting the advanced features of MQTT v5.0 protocol," *IEEE Internet Things J.*, vol. 12, no. 12, pp. 22090–22109, 2025, doi: 10.1109/JIOT.2025.3549784.

[5] U. Ahmed et al., "Signature-Based Intrusion Detection Using Machine Learning and Deep Learning Approaches Empowered with Fuzzy Clustering," *Sci. Rep.*, vol. 15, no. 1, p. 1726, 2025.

[6] B. Madhu et al., "Intrusion detection models for IoT networks via deep learning approaches," *Meas.: Sensors*, vol. 25, Art. no. 100641, 2023.

[7] M. A. Hossain, "Deep Learning-Based Intrusion Detection for IoT Networks: A Scalable and Efficient Approach," *EURASIP J. Inf. Secur.*, vol. 2025, no. 1, Art. no. 28, 2025.

[8] S. Senthil and N. Muthukumaran, "Joined Bi-Model RNN with Spatial Attention and GAN-Based IoT Botnet Attacks Detection," *Sādhanā*, vol. 48, no. 3, Art. no. 141, 2023.

[9] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly detection IDS for detecting DoS attacks in IoT networks based on machine learning algorithms," *Sensors*, vol. 24, no. 2, Art. no. 713, 2024, doi: 10.3390/s24020713.

[10] K. Albulayhi et al., "IoT intrusion detection using machine learning with a novel high performing feature selection method," *Appl. Sci.*, vol. 12, no. 10, Art. no. 5015, 2022, doi: 10.3390/app12105015.

[11] M. Douiba et al., "An improved anomaly detection model for IoT security using decision tree and gradient boosting," *J. Supercomput.*, vol. 79, no. 3, pp. 3392–3411, 2023, doi: 10.1007/s11227-022-04783-y.

[12] A. Khan, M. Asdaque Hussain, and F. Anwer, "A hybrid lightweight deep learning-based intrusion detection approach in IoT utilizing feature selection & explainable artificial intelligence," *IEEE Access*, vol. 13, pp. 192451–192466, 2025, doi: 10.1109/ACCESS.2025.3630753.

[13] V. Jain et al., "Optimizing intrusion detection using a hybrid CNN-LSTM deep learning framework on the NSL-KDD dataset," *Discov. Comput.*, vol. 29, Art. no. 194, 2026, doi: 10.1007/s10791-026-10056-6.

[14] R. Logeswari et al., "IA-IDS: An intelligent adaptive intrusion detection system for IoT security using CNN, BiLSTM, and attention mechanism," *Peer-to-Peer Netw. Appl.*, vol. 19, no. 1, Art. no. 32, 2026, doi: 10.1007/s12083-025-02177-4.

[15] Y. Chen and M. Mir, "An intrusion detection system for industrial IoT using CNN-LSTM and reptile search algorithm," *Cluster Comput.*, vol. 28, no. 7, Art. no. 452, 2025, doi: 10.1007/s10586-024-05086-y.

[16] D. Kilichev, D. Turimov, and W. Kim, "Next-generation intrusion detection for IoT EVCS: Integrating CNN, LSTM, and GRU models," Mathematics, vol. 12, no. 4, Art. no. 571, 2024, doi: 10.3390/math12040571.

[17] A. A. Wakili et al., "Advancing machine learning strategies for power consumption-based IoT botnet detection," *Sensors*, vol. 25, no. 24, Art. no. 7553, 2025, doi: 10.3390/s25247553.

[18] W. Alayash et al., "Assessing LSTM and GRU for multi-dataset intrusion detection in IoT environments," *Stat. Optim. Inf. Comput.*, vol. 15, no. 4, pp. 3155–3173, 2026, doi: 10.19139/soic-2310-5070-3226.

[19] S. Duraibi and A. M. Alashjaee, "Detection of Internet of Things network attacks by hybrid deep learning (CNN-LSTM) algorithm to enhance security," *Sci. Rep.*, vol. 16, Art. no. 20653, 2026, doi: 10.1038/s41598-026-51043-7.

[20] S. A. S. Hussien, M. S. I. Alsumaidaie, and N. H. M. Ali, "Enhanced IOT cyber-attack detection using grey wolf optimized feature selection and adaptive SMOTE," *Mesopotam. J. Comput. Sci.*, vol. 2025, pp. 355–370, 2025, doi: 10.58496/mjcsc/2025/023.

[21] A. O. Widodo, B. Setiawan, and R. Indraswari, "Machine learning-based intrusion detection on multi-class imbalanced dataset using

SMOTE," *Procedia Comput. Sci.*, vol. 234, pp. 578–583, 2024, doi: 10.1016/j.procs.2024.03.042.

[22] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *J. Supercomput.*, vol. 79, no. 10, pp. 10611–10644, 2023, doi: 10.1007/s11227-023-05073-x.

[23] Y. N. Rao and K. Suresh Babu, "An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset," *Sensors*, vol. 23, no. 1, Art. no. 550, 2023, doi: 10.3390/s23010550.

[24] G. Zhu, Y. Yu, Z. Li, and Y. Dai, "GMDDPM-CT: Advancing network intrusion detection through diffusion-based sample synthesis and deep learning," *ICT Express*, vol. 12, in press, 2026, doi: 10.1016/j.icte.2026.01.014.

[25] S. M. H. Mirsadeghi, H. Bahsi, R. Vaarandi, and W. Inoubli, "Learning from few cyber-attacks: Addressing the class imbalance problem in machine learning-based intrusion detection in software-defined networking," *IEEE Access*, vol. 11, pp. 140428–140442, 2023, doi: 10.1109/ACCESS.2023.3341755.

[26] A. Demirbaş Paray and M. Aydos, "Federated learning for intrusion detection under class imbalance: A multi-domain ablation study with per-client SMOTE," *Appl. Sci.*, vol. 16, no. 2, Art. no. 801, 2026, doi: 10.3390/app16020801.

[27] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[28] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1724–1734, doi: 10.3115/v1/D14-1179.

[29] M. Fatima, O. Rehman, I. M. H. Rahman, A. Ajmal, and S. J. Park, "Towards ensemble feature selection for lightweight intrusion detection in resource-constrained IoT devices," *Future Internet*, vol. 16, no. 10, Art. no. 368, 2024, doi: 10.3390/fi16100368.

[30] H. Zhang, "Development of an intelligent intrusion detection system for IoT networks using deep learning," *Discov. Internet Things*, vol. 5, Art. no. 74, 2025, doi: 10.1007/s43926-025-00177-7.

[31] S. Alosaimi and S. M. Almutairi, "An intrusion detection system using BoT-IoT," *Appl. Sci.*, vol. 13, no. 9, Art. no. 5427, 2023, doi: 10.3390/app13095427.

[32] A. M. Rasool, N. S. Safa, and C. Mbarushimana, "Towards privacy-preserving deep learning for intelligent IoT botnet detection," *Appl. Sci.*, vol. 16, no. 3, Art. no. 1665, 2026, doi: 10.3390/app16031665.

[33] S. Ullah, J. Wu, Z. Lin, M. M. Kamal, H. Mostafa, M. Sheraz, and T. C. Chuah, "Comparative analysis of deep learning and traditional methods for IoT botnet detection using a multi-model framework across diverse datasets," *Sci. Rep.*, vol. 15, Art. no. 31072, 2025, doi: 10.1038/s41598-025-16553-w.