Split-Federated BERT with Adversarial Training for Edge Intrusion Detection

Hang Shen, *Member, IEEE,* Qi Liu, Fang Li, Tianjing Wang, *Member, IEEE,* Yuanfei Dai, and Guangwei Bai

***Abstract*—Pre-trained language models, represented by Bidirectional Encoder Representations from Transformers (BERT), show great potential for deep learning-based intrusion detection systems (IDS) due to their strong semantic modeling capability. However, the high cost of training and fine-tuning limits their applicability in large-scale and resource-constrained environments. To address this challenge, we propose a Split-Federated BERT framework with adversarial training for edge intrusion detection. The framework partitions BERT into an Embedding layer deployed at the edge and Transformer and Head layers hosted in the cloud, enabling collaborative training between edge devices and the cloud. At the edge, a conditional generative adversarial network (CGAN) integrated with BERT enhances traffic feature extraction. Guided by BERT, the generator adapts to local traffic distributions, improving sample coverage and feature representation. Edge devices perform local updates to the Embedding layer, while the cloud conducts high-dimensional semantic learning using BERT's Transformer and Head layers. During federated aggregation, a multi-head attention mechanism is employed in the cloud to differentially weight model updates, ensuring distributional alignment and stable convergence. This design decouples edge-side adversarial enhancement from federated aggregation, reducing both computational and communication overhead. Experimental results on multiple authoritative datasets demonstrate that the proposed method consistently outperforms local deep learning, BERT, federated learning, and split learning baselines in precision, recall, and F1-score, while improving edge computational efficiency and communication cost.**

***Index Terms*—BERT, Multi-Head Attention, Federated Learning, Split learning, Adversarial Training, Intrusion Detection.**

I. INTRODUCTION

WITH the increasing deployment of connected devices across diverse networked environments, the ecosystem of these devices has gradually become a major target for network attacks. According to the 2024 OneKey OT/IoT Cybersecurity Report[1], 52% of organizations experienced at least one attack caused by connected devices in the past year. The 2025 Bitdefender IoT Security Landscape Report[2] indicates that, among 58 million monitored devices, the frequency of attack events is on the rise. Intrusion Detection Systems (IDSs) [1] play a critical role in monitoring network traffic, identifying anomalous behavior, and blocking potential threats. However, the heterogeneity of connected devices, the variety of protocols, and the high-dimensional nature of network traffic pose significant challenges for traditional machine learning methods in capturing the deep semantics of traffic.

Deep learning (DL) enables end-to-end feature learning from raw network traffic and is effective in detecting complex attack behaviors through hierarchical representations [2]. However, network traffic exhibits strong sequential dependencies and protocol-level semantics, which conventional DL models often fail to capture, particularly in fine-grained interactions across complex protocols. Based on the Transformer architecture [3], large language models (LLMs) [4] leverage self-attention mechanisms to model long-range dependencies and contextual relationships in sequential data. Researchers have explored fine-tuning LLMs for direct traffic classification [5], [6], enabling them to perform end-to-end intrusion detection without handcrafted feature engineering.

Despite their impressive representational capabilities, LLMs introduce substantial computational and memory overhead. For example, TrafficLLM [6] achieves high detection accuracy, yet its substantial computational and memory overhead limits its practicality for real-time detection of high-density traffic in resource-constrained edge environments. In contrast, relatively lightweight pre-trained language model BERT [7], [8] offers a favorable balance between semantic modeling and computational efficiency. Through bidirectional encoding, BERT captures contextual dependencies in traffic flows. This capability enables effective intrusion detection by modeling traffic as serialized sequences of protocol fields, event patterns, and behavioral signals [9]–[12].

Split learning (SL) [13], [14] partitions models across edge and cloud, allowing lightweight layers to run locally while offloading computation-intensive components to the cloud. Federated learning (FL) [15], [16] further enables collaborative training across distributed edge nodes by exchanging model updates through centralized aggregation. The integration with SL and FL facilitates edge collaboration and edge-cloud decoupling, providing a scalable and resource-efficient framework for BERT-enhanced edge intrusion detection.

Manuscript received dd mm 2026; revised dd mm 2026; accepted dd mm 2026. This work was supported in part by the National Natural Science Foundation of China under Grants 61502230, 61501224 and 62202221, in part by the Natural Science Foundation of Jiangsu Province under Grants BK20201357 and BK20220331, and in part by the State Administration for Market Regulation Science and Technology Program under Grant 2025MK183, and in part by the Market Supervision Administration Science and Technology Fund of Jiangsu Province under Grant KJ2026027. (Corresponding author: Hang Shen)

Hang Shen, Qi Liu, Fang Li, Tianjing Wang, Yuanfei Dai, and Guangwei Bai are with the College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, Nanjing, 211816, China. E-Mail: {hshen, qiliu, wangtianjing, daiyuanfei, bai}@njtech.edu.cn, lns1802026@163.com

Digital Object Identifier: xxx

[1]https://www.onekey.com/resource/ot-iot-cybersecurity-report-2024

[2]https://blogapp.bitdefender.com/hotforsecurity/content/files/2025/10/2025_iot_security_report.pdf

A. Challenging Issues and Related Works

Despite the potential, collaborative BERT training for edge intrusion detection remains challenging.

1) Consistency learning across edge models. FL client's heterogeneity leads to divergence in attention patterns and intermediate semantic representations, which conventional parameter-averaging aggregation cannot resolve. To address heterogeneity, Lee et al. [17] introduce a meta-network guided by local statistical features to dynamically adjust model parameters, improving both local adaptability and global stability. DAFL [18] mitigates the impact of low-quality clients on the global model, enhancing performance while reducing communication overhead. XGBoost [19] further integrates anonymous aggregation and differential privacy to enable secure cross-organization anomaly detection. RingSFL [20] integrates FL with a model split mechanism and employs a ring topology to handle client heterogeneity, reduce straggler effects, and improve training efficiency in distributed settings. FedPepTAO [21] introduces a parameter-efficient prompt tuning approach for LLMs within FL, reducing parameters updated during training. Despite these advances, existing approaches primarily focus on parameter-level heterogeneity and convergence optimization, while neglecting aligning representations across edge models.

2) Efficient edge-cloud collaborative training. Under the SL paradigm, frequent exchange of gradients or intermediate activations may incur substantial communication cost and training latency. Kim et al. [22] partition DL models into sequentially executed segments, alleviating local computational pressure. In [23], recurrent neural networks (RNNs) are split into multiple sub-networks, and intermediate features are exchanged only when necessary, reducing communication rounds while preserving accuracy. MergeSF [24] further improves training and communication efficiency through feature merging and batch-level adjustment. Several studies have also explored split-federated designs. FedBERT [25] introduces parallel and sequential federated split training modes, while HSFL [26] models training latency and privacy protection by formulating edge–cloud model splitting as a contextual bandit optimization problem. CHEESE [27] coordinates distributed clusters, model splitting, and bandwidth scheduling. MobiFormer [14] is a split-federated transfer learning framework for drone network resource allocation. However, existing split-federated approaches are largely designed for conventional DL models and lack solutions tailored to pre-trained language models.

3) Decoupling edge-side distribution enhancement from aggregation. In distributed edge networks, traffic data is often long-tailed and non-IID [28], limiting the effectiveness of federated and centralized training in capturing semantically rich local representations. Existing studies show that generative adversarial network (GAN)-based augmentation can partially mitigate semantic sparsity induced by data imbalance in intrusion detection [2], [29]. TMG-GAN [30] introduces a GAN-based imbalanced learning framework that targets minority attack classes by synthesizing diverse and semantically meaningful samples, improving detection under highly skewed traffic distributions. Constantin et al. [31] present multi-GAN architectures for augmenting streaming and tabular network traffic. However, the heterogeneity in locally augmented distributions can amplify noise under indiscriminate aggregation, and the inherent instability of adversarial training further exacerbates convergence issues. He et al. [32] propose a conditional GAN (CGAN [33])-based FL approach for intrusion detection, which integrates long short-term memory (LSTM) for classification and incorporates generated samples into the original data for enhanced detection. Similarly, FGA-IDS [34] explores FL-GAN integration to improve data sufficiency and robustness in distributed intrusion detection. FedGAN-ID [35] leverages GAN-generated attack samples in FL. Despite these advancements, existing approaches primarily focus on performance and data augmentation, neglecting to decouple edge-side distribution enhancement from federated aggregation.

B. Contributions and Organization

To address the aforementioned challenges, we propose a split-federated BERT framework with adversarial training for intrusion detection. This framework integrates edge-cloud decoupling, distribution enhancement, and a BERT-centric split-federation design, optimizing both the computational load at the edge and the global model aggregation in the cloud. The main contributions are as follows:

- **Edge-side distribution enhancement through adversarial training.** We introduce a CGAN-based distribution enhancement on the edge, where BERT is embedded into the CGAN discriminator to improve protocol semantics and anomalous pattern detection. Backpropagated adversarial gradients guide the generator to adapt to local traffic distributions, enhancing sample diversity and improving the quality of traffic feature representation.
- **Split-BERT for edge-cloud collaboration.** BERT is partitioned into a lightweight Embedding layer at the edge and a computationally intensive Transformer layer with a prediction head in the cloud. This design reduces edge computational load while preserving BERT's semantic modeling power, enabling efficient local updates and high-dimensional semantic learning on the cloud.
- **Cloud-side federated aggregation with multi-head attention.** Multi-head attention is used during federated aggregation to assign differential weights to edge model updates, improving semantic alignment and stability. Edge-side adversarial enhancement is decoupled from federated aggregation to prevent negative effects on global model.

Experimental results on the CSE-CIC-IDS2018[3], NF-ToN-IoT[4], and NF-UNSW-NB15[5] datasets demonstrate that the proposed method consistently outperforms local DL, BERT, FL, and SL baselines in terms of precision, recall, and F1-score, while simultaneously improving edge computational efficiency and reducing cloud–edge communication costs.

The remainder of this paper is organized as follows. Section II outlines split-federated BERT architecture with adversarial training. In Section III, we present the edge-cloud

[3]https://www.unb.ca/cic/datasets/ids-2018.html
[4]https://rdm.uq.edu.au/files/a4ad7080-ef9c-11ed-a964-b70596e96ad5
[5]https://rdm.uq.edu.au/files/8c6e2a00-ef9c-11ed-827d-e762de186848

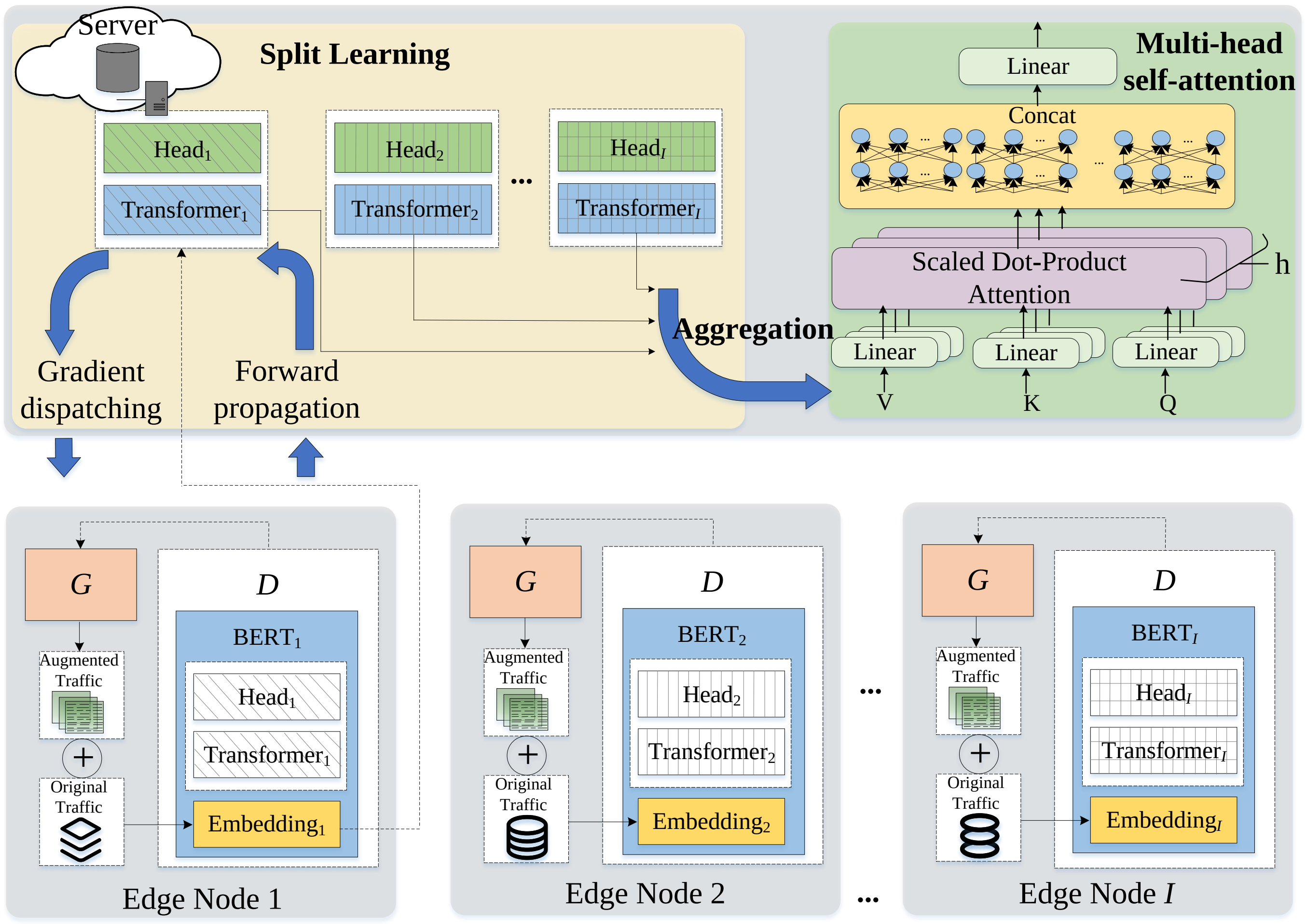


Fig. 1: Split-federated learning framework for cloud-edge collaboration. During training, edge nodes deploy a generator G, a classifier C, and BERT's Embedding layer, where adversarial training acts as an auxiliary enhancement for BERT. The cloud hosts BERT's Transformer and Head layers for high-dimensional modeling, while a cloud-side multi-head attention module performs federated aggregation by weighting edge model updates. During online detection, edge nodes rely on the BERT-enhanced traffic classifier.

collaborative training of BERT across edge devices. Section IV discusses experimental setup, ablation study, and baseline selection. Section V presents the evaluation of detection performance, along with the communication and computational costs. Finally, we summarize the findings and discuss potential directions for future work in Section VI.

II. Split-Federated Architecture

This section outlines the partitioning of BERT across edge and cloud. As shown in Fig. 1, the split-federated architecture reduces the computational load on edge devices by handling low-complexity tasks locally, while offloading resource-intensive computations to the cloud. This collaboration enhances both local processing and global model aggregation.

A. BERT Model Partitioning

To facilitate efficient collaborative training on resource-constrained edge nodes, we propose a cloud-edge federated split framework that integrates layered BERT deployment, edge-side adversarial enhancement, and cloud-side federated aggregation. As illustrated in Fig. 1, the system comprises a cloud server and I edge nodes.

The partitioning strategy optimizes resource utilization by allocating tasks based on the computational complexity of each layer. Accordingly, BERT is partitioned into Embedding (w_b^E), Transformer (w_b^T), and Head layers (w_b^H). The Embedding layer, which has low complexity and fewer parameters, is deployed on edge nodes to convert raw traffic features into vector representations. In contrast, the Transformer layer, which requires more computational resources, and the Head layer, which performs high-dimensional semantic classification, are migrated to the cloud to reduce training costs on the edge.

To enhance sample diversity, a CGAN is incorporated at the edge. Specifically, the Discriminator (D) is a composite architecture consisting of a BERT-enhanced semantic encoder and a classification network (w_b^C), while the Generator (G) synthesizes high-dimensional feature representations. Formally, the discriminator is defined as $w^D = [w_b^E, w_p^T, w_p^H, w_b^C]$, where w_b^E and w_b^C are deployed on the edge, while w_p^T and w_p^H are

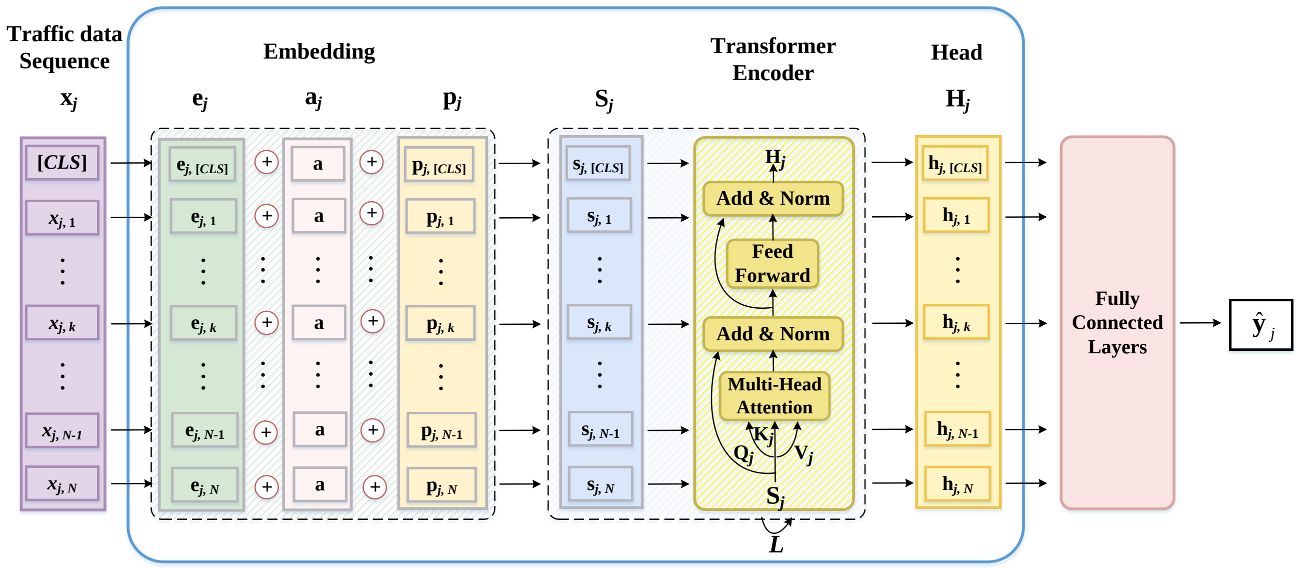


Fig. 2: BERT-enhanced traffic classifier.

hosted on the cloud. Accordingly, the edge-resident bottom-tier model is denoted as $w_b = \{w_b^G, w_b^C, w_b^E\}$, and the cloud-hosted top-tier model is represented as $w_p = \{w_p^T, w_p^H\}$. Although physically distributed, these components jointly form the discriminator through split learning, collaborating via forward propagation and gradient backpropagation across the split point.

This dual-dimensional optimization approach enhances the system performance by balancing computational resources between the edge and cloud. The edge side leverages a CGAN for local data enhancement and BERT for efficient feature extraction, ensuring that edge devices handle only low-complexity tasks. Meanwhile, the cloud optimizes the aggregation of updates using multi-head attention, ensuring that the global model benefits from improved parameter updates informed by data from edge nodes. The edge-side distribution enhancement improves local model updates, which are then aggregated at the cloud via multi-head attention to produce a refined global model. The updated global parameters are subsequently distributed back to the edge nodes, guiding the next round of local adversarial training and distribution enhancement, thereby forming an edge-cloud closed-loop optimization.

B. BERT-Augmented CGAN Model

On the edge side, a CGAN enhances sample diversity. The D incorporates BERT as a semantic encoder to capture global dependencies between traffic features, improving the recognition of protocol semantics and anomalous patterns. The G produces high-dimensional feature representations based on class labels, and D, enhanced by BERT, distinguishes real from synthetic ones. This process optimizes both components to enhance the model's ability to capture complex traffic patterns.

As shown in Fig. 2, BERT is integrated into the D, thereby forming an enhanced traffic classifier. In the Embedding layer, traffic data is transformed into a sequence format suitable for BERT input. Specifically, each sequence of traffic data is preceded by the special identifier $[CLS]$ to facilitate recognition by the edge node i in $w_{b,i}^E$. Let x be the attribute features of the traffic data. Then, each traffic sequence is represented as $\mathbf{x} = [CLS, x_1, \ldots, x_k, \ldots, x_N]$, where $\mathbf{x}_j$ represents the j-th traffic sequence, and $x_{j,k}$ represents the k-th attribute feature of the j-th sequence. Via a single linear layer, $x_{j,k}$ is first projected into a d-dimensional space

$$\mathbf{e}_{j,k} = W^E x_{j,k} + b^E \tag{1}$$

where $\mathbf{e}_{j,k}$ is the embedding of $x_{j,k}$, while W^E and b^E are the trainable weight matrix and bias term, respectively.

However, since the self-attention mechanism in Transformers is permutation-invariant, it cannot inherently capture the sequential order of traffic features. We introduce positional embeddings $\mathbf{p}_{j,k}$ and segment embeddings $\mathbf{a}$ to enrich the representation. The final composite input is expressed as

$$\mathbf{s}_{j,k} = \mathbf{e}_{j,k} + \mathbf{a} + \mathbf{p}_{j,k} \tag{2}$$

where $\mathbf{s}_{j,k}$ denotes the word embedding of the k-th attribute feature of the j-th traffic sequence. Accordingly, traffic sequence $\mathbf{x}_j$ is encoded as the embedding sequence, $\mathbf{S}_j = [\mathbf{s}_{j,[CLS]}, \mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,k}, \ldots, \mathbf{s}_{j,N}]$, as the input to the Transformer encoder.

Each encoder includes a multi-head self-attention mechanism, a feedforward network, and residual connections with layer normalization. The multi-head self-attention mechanism models global dependencies between traffic features by calculating attention weights between queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$). After self-attention, a feedforward neural network applies a nonlinear transformation to each position's input to capture more complex features. Finally, the features extracted by the Transformer encoder are mapped to the output space for the specific task. For traffic classification, BERT uses the output for special identifier $[CLS]$ as the feature representation of the entire sequence and connects it to a fully connected layer

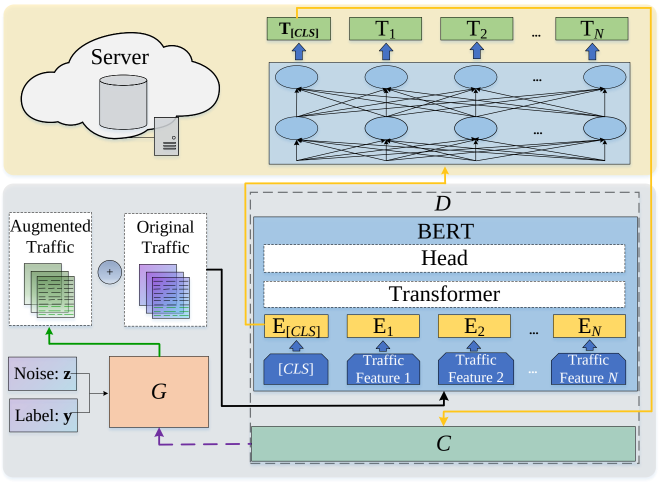


Fig. 3: Adversarial splitting iteration for edge models.

to predict the traffic sequence's category.

III. Split-Federated Training

Split-federated BERT training involves iterative cloud-edge collaboration with adversarial training on the edge. In each global round, edge node i performs local adversarial training, optimizing sample quality through dynamic games between G and D. The edge node uploads intermediate features to the cloud, which performs backpropagation and applies multi-head attention to aggregate BERT updates. After receiving the global parameters, edge nodes continue adversarial training and local optimization.

A. Split Iteration with Adversarial Training

In the cloud-edge architecture, edge node i trains the bottom-tier model $w_{b,i}$ (which includes G, the classification network, and BERT's Embedding layer) using local data $\mathbb{D}_i$, while the cloud server updates $w_{p,i}$. Let $\mathbf{W}_{b,i}$ and $\mathbf{W}_{p,i}$ represent the parameters of $w_{b,i}$ and $w_{p,i}$. The prediction of edge node i for input $\mathbf{x}$ is determined by $f(\mathbf{x};\mathbf{W}_{b,i},\mathbf{W}_{p,i})$. The goal of federated split training is to minimize the cross-entropy loss across all edge nodes

$$\min_{\mathbf{W}} L(\mathbf{W}) = -\frac{1}{I}\sum_{i=1}^{I}\sum_{(\mathbf{x},\mathbf{y})\in\mathbb{D}i} \mathbf{y}\log f(\mathbf{x};\mathbf{W}_{b,i},\mathbf{W}_{p,i}) \quad (3)$$

where $\mathbf{W} = \{\mathbf{W}_b,\mathbf{W}_p\}_i^I$ represent the global parameters, and $\mathbf{y}$ denotes the ground-truth label.

The basic training process of the bottom-tier model includes forward and backward propagation for both $w_{b,i}$ and $w_{p,i}$, as shown in Fig. 3. Edge node i performs forward propagation on a batch of local traffic data and sends the intermediate feature representations of the split layer to the cloud. The cloud server performs forward propagation on $w_{p,i}$ based on the received intermediate features and updates $w_{p,i}$ via backpropagation. Finally, the cloud sends the backpropagated gradients back to edge node i, where the node updates $w_{b,i}$ through backward propagation. This complete forward/backward propagation constitutes a local iteration.

Traffic data is encoded into $\mathbf{S}_j$ in the embedding layer and uploaded to the cloud. The L layers of the Transformer encoder extract deep semantic features. Each Transformer encoding layer consists of two sub-layers: a multi-head attention mechanism and a feedforward neural network, both of which are followed by residual connections and normalization. The outputs of the first, second, and final sub-layers in the l-th Transformer encoding layer are expressed as

$$\mathbf{E}_{j,l} = \mathrm{Norm}(\mathbf{S}_{j,l} + \mathrm{MultiHeadAtt}(\mathbf{S}_{j,l}))) \quad (4)$$

$$\mathbf{H}_{j,l} = \mathrm{Norm}(\mathbf{E}_{j,l} + \mathrm{FFN}(\mathbf{E}_{j,l}))) \quad (5)$$

$$\mathbf{S}_{j,l+1} = \mathbf{H}_{j,l}, \forall l < L \quad (6)$$

where MultiHeadAtt$(\cdot)$ represents the multi-head attention function, FFN$(\cdot)$ represents the position-wise feedforward function, and Norm$(\cdot)$ is the layer normalization function. The multi-head attention mechanism allows the model to simultaneously capture different aspects of the traffic sequence. The traffic sequence is projected into subspaces computed by different attention heads, and the results are then concatenated and projected back into the representation space via a weight matrix. Specifically, the attention for the t-th head in the l-th Transformer encoding layer is computed as

$$\mathbf{head}_{j,l,t} = \mathrm{Att}(\mathbf{Q}_{j,l,t},\mathbf{K}_{j,l,t},\mathbf{V}_{j,l,t}) \quad (7)$$

where $\mathbf{Q}_{j,l,t}$, $\mathbf{K}_{j,l,t}$, and $\mathbf{V}_{j,l,t}$ are the query, key, and value vectors for the t-th attention head in the l-th Transformer encoding layer for the j-th traffic sequence. By concatenating the attention outputs from all M heads and applying a weight matrix W^O, the final sequence representation is obtained as

$$\bar{\mathbf{S}}_{j,l} = [\mathbf{head}_{j,l,1},\ldots,\mathbf{head}_{j,l,t},\ldots,\mathbf{head}_{j,l,M}]W^O. \quad (8)$$

After passing through the Transformer encoders in w_p^T on the cloud, the traffic sequence yields a new vector representation.

The output vector at $[CLS]$ in $w_{p,i}^H$ is used as the semantic representation of the traffic sequence. Edge node i in $w_{b,i}^C$ processes $\mathbf{h}_j$ and $\mathbf{g}_j$, where $\mathbf{h}_j$ is the probability distribution of the j-th traffic data encoded in high-dimensional space by $w_{p,i}$ through the process, and $\mathbf{g}_j$ is the high-dimensional feature representation generated by $w_{b,i}^G$ based on class labels and random noise vectors. The output of $w_{b,i}^C$ provides the probability of traffic type. During iteration, $w_{b,i}^G$ generates high-dimensional feature representations of traffic to confuse D, while $w_{b,i}^C$ attempts to correctly classify the traffic data.

Let $\mathbf{W}_{b,i,r}$ denote the model parameters of $w_{b,i}$ at the r-th iteration, and B_i be a mini-batch of data samples. The random gradient under $\mathbf{x}$ and $\mathbf{W}_{b,i,r}$ is $\nabla\ell(\mathbf{x};\mathbf{W}_{b,i,r})$. Let the learning rate be η. After one iteration, $\mathbf{W}_{b,i,r}$ are updated as

$$\mathbf{W}_{b,i,r+1} = \mathbf{W}_{b,i,r} - \eta\frac{1}{|B_i|}\sum_{\mathbf{x}\in B_i}\nabla\ell(\mathbf{x};\mathbf{W}_{b,i,r}). \quad (9)$$

Similarly, for the r-th iteration, let $\mathbf{W}_{p,i,r}$ denote the model parameters of $w_{p,i}$ at edge node i, and the random gradient under $\mathbf{W}_{b,i,r}$ output and $\mathbf{W}_{p,i,r}$ is $\nabla\ell(\mathbf{W}_{b,i,r}(\mathbf{x});\mathbf{W}_{p,i,r})$.

After one iteration, $\mathbf{W}_{p,i,r}$ are updated as

$$\mathbf{W}_{p,i,r+1} = \mathbf{W}_{p,i,r} - \eta \frac{1}{|B_i|} \sum_{\mathbf{x} \in B_i} \nabla \ell(\mathbf{W}_{b,i,r}(\mathbf{x}); \mathbf{W}_{p,i,r}). \tag{10}$$

During iteration, $w_{b,i}^G$ takes random noise $\mathbf{z}$ and label $\mathbf{y}$ as input, and outputs high-dimensional feature representations $w_{b,i}^G(\mathbf{z}, \mathbf{y})$. The discriminator at edge node i is denoted as $w_i^D = [w_{b,i}^E, w_{p,i}^T, w_{p,i}^H, w_{b,i}^C]$, which classifies both real traffic and generated features conditioned on $\mathbf{y}$. Specifically, $w_i^D(\mathbf{x}, \mathbf{y})$ denotes the discrimination of real traffic $\mathbf{x}$, and $w_i^D(w_{b,i}^G(\mathbf{z}, \mathbf{y}), \mathbf{y})$ denotes the discrimination of generated features. The optimization objective for $w_{b,i}^G$ and w_i^D is formulated as

$$\begin{aligned}\min_{w_{b,i}^G} \max_{w_i^D} V(w_i^D, w_{b,i}^G) &= \mathbb{E}_{\mathbf{x} \sim p_x}[\log w_i^D(\mathbf{x}, \mathbf{y})] \\ &+ \mathbb{E}_{\mathbf{z} \sim p_z}[\log(1 - w_i^D(w_{b,i}^G(\mathbf{z}, \mathbf{y}), \mathbf{y}))].\end{aligned} \tag{11}$$

Through dynamic adversarial feedback, the BERT-enhanced w_i^D continuously improves its feature discrimination, guiding $w_{b,i}^G$ to adapt to local traffic distributions. In turn, the diverse feature representations generated by $w_{b,i}^G$ expand the coverage of the feature space, enabling BERT to learn more robust and discriminative representations.

B. Multi-Head Attention based Split-Federated Aggregation

Unlike the self-attention within BERT's Transformer encoder for modeling dependencies among traffic features, the multi-head attention serves to weight model updates across edge nodes during federated aggregation. In the split-federated aggregation phase, the cloud server maintains separate Transformer and Head layers, denoted as $w_{p,i}^T$ and $w_{p,i}^H$ for each edge node. $w_{p,i}^T$ from all edge nodes are aggregated using the multi-head attention mechanism and distributed back to the edge to update the global model.

The scaled dot-product attention mechanism is used by the cloud server to compute the similarity between $w_{p,i}^T$ from different edge nodes. First, $\mathbf{W}_{p,i}^T$ for all $w_{p,i}^T$ are stacked into a tensor, $\mathbf{W}_p^T = [\mathbf{W}_{p,1}^T, \ldots, \mathbf{W}_{p,i}^T, \ldots, \mathbf{W}_{p,I}^T]$. Next, $\mathbf{W}_p^T$ is projected to generate the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, and value matrix $\mathbf{V}$. The similarity between each query and key is computed as the dot product of $\mathbf{Q}$ and $\mathbf{K}$. If they are highly similar, the corresponding value is considered relevant, and the output is the weighted sum of the values. Using this mechanism, the cloud server applies multi-head attention to capture feature representations of $\mathbf{W}_p^T$ in different subspaces. The projection of $\mathbf{W}_p^T$ to subspaces computed by different attention heads is given by

$$\begin{cases} \mathbf{Q}_t = \mathbf{W}_p^T \mathbf{W}_t^Q \\ \mathbf{K}_t = \mathbf{W}_p^T \mathbf{W}_t^K \\ \mathbf{V}_t = \mathbf{W}_p^T \mathbf{W}_t^V. \end{cases} \tag{12}$$

The query, key, and value vectors for the t-th attention head are denoted as $\mathbf{Q}_t$, $\mathbf{K}_t$, and $\mathbf{V}_t$, respectively. These vectors are computed by projecting $\mathbf{W}_p^T$ using the corresponding learnable weight matrices $\mathbf{W}_t^Q$, $\mathbf{W}_t^K$, and $\mathbf{W}_t^V$. Each attention head independently computes an attention matrix as

$$\alpha_t = \text{softmax}\left(\frac{\mathbf{Q}_t \cdot {\mathbf{K}_t}^T}{\sqrt{d_k}}\right) \tag{13}$$

where d_k represents the dimension of $\mathbf{K}_t$. The attention weight for the corresponding $\mathbf{W}_{p,i}^T$ from the t-th attention head is calculated as

$$\gamma_{i,t} = \alpha_t \mathbf{V}_t. \tag{14}$$

Each attention head independently projects all $\mathbf{W}_{p,i}^T$, with $\mathbf{Q}_t = [\mathbf{Q}_{t,1}, \mathbf{Q}_{t,2}, \ldots, \mathbf{Q}_{t,I}]$, $\mathbf{K}_t = [\mathbf{K}_{t,1}, \mathbf{K}_{t,2}, \ldots, \mathbf{K}_{t,I}]$, and $\mathbf{V}_t = [\mathbf{V}_{t,1}, \mathbf{V}_{t,2}, \ldots, \mathbf{V}_{t,I}]$.

The cloud aggregates parameters $\mathbf{W}_{p,i}^T$ from the edge nodes with the multi-head attention mechanism. For each attention head, self-attention is performed to obtain a new vector representation. By aggregating the results from all attention heads, the cloud can prioritize edge nodes that contribute to the global update, thereby refining model aggregation. After r rounds of iterative updates, parameters $\mathbf{W}_p^T$ are updated as

$$\mathbf{W}_p^T = \sum_{i=1}^{I} \gamma_i \mathbf{W}_{p,i}^T \tag{15}$$

where γ_i is the average of $\gamma_{i,t}$ from all attention heads.

The multi-head attention-based aggregation mechanism enables the cloud to prioritize relevant edge nodes and efficiently aggregate model updates, improving the global model while reducing communication cost. The detailed split-federated training process with multi-head attention is presented in Algorithm 1, including both forward and backward propagation at the edge and cloud.

IV. Experimental Preparation

A. Edge Data Configuration and Distribution Analysis

We evaluated all methods on three datasets: CSE-CIC-IDS2018, NF-ToN-IoT, and NF-UNSW-NB15. Table I summarizes their data distributions, highlighting significant class imbalance. In both CSE-CIC-IDS2018 and NF-UNSW-NB15, normal traffic predominates, accounting for 66.01% in NF-UNSW-NB15, while attack traffic represents less than 35%. In NF-ToN-IoT, normal traffic accounts for 36.01%, whereas the least frequent attack class constitutes only 0.02%. To emulate data heterogeneity across distributed edge gateways, we randomly down-sampled three categories in each dataset to create edge nodes with distinct data distributions.

B. Ablation and Baseline Methods

For ablation evaluation, the proposed method is divided into four variants, as shown in Table II. Proposed-1 integrates BERT as a feature extractor into D, forming a split-federated BERT architecture. In this version, the edge deploys G, C, and BERT's embedding layer, while the cloud deploys BERT's Transformer and Head layers and aggregates the Transformer layer using a multi-head attention mechanism. Proposed-2, compared to Proposed-1, removes the CGAN and implements feature extraction and classification using BERT. The edge deploys BERT's embedding layer and classifier network, whereas the cloud deploys BERT's Transformer and Head layers and

Algorithm 1: Cloud-edge split-federated algorithm

```
Input: Number of edge nodes $I$; number of split iterations per global training round $R$; batch size for split iterations $B$; learning rate $\eta$;
Output: Global model parameters $\mathbf{W}_p^T$;
Cloud Server:
Initialize model parameters $\mathbf{W}_p^T, \mathbf{W}_p^H$ as $\mathbf{W}_{p,0}^T, \mathbf{W}_{p,0}^H$;
for global epoch $j = 0, 1, ...$ do
    $\{\mathbf{W}_{p,i,j}^T\}|_{i=1}^I \leftarrow$ Assign $\mathbf{W}_{p,j}^T$ to edge node $i$'s $\mathbf{W}_{p,i}^T$;
    for edge node $i = 1, \ldots, I$ in parallel do
        EdgeUpdate();
    $\mathbf{W}_p^T \leftarrow [\mathbf{W}_{p,1,j}^T, \ldots, \mathbf{W}_{p,i,j}^T, \ldots, \mathbf{W}_{p,I,j}^T]$;
    $\alpha_j \leftarrow$ Calculate attention weight matrix softmax($\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}$);
    $\gamma_{i,j} \leftarrow$ Calculate attention output $\alpha_j \mathbf{V}$;
    $\mathbf{W}_{p,j+1}^T \leftarrow$ Calculate global parameters $\sum_{i=1}^I \gamma_{i,j} \mathbf{W}_{p,i,j}^T$;
Function ServerTrmFordprop($\mathcal{P}_{ford}^E, i$):
    $\mathbf{h}_j \leftarrow$ Compute forward propagation of $\mathbf{W}_{p,i,j}^T$, $\mathbf{W}_{p,i,j}^H$;
    Output $\mathbf{h}_j$;
Function ServerTrmBackprop($\mathcal{P}_{back}^C, i$):
    $\mathbf{W}_{p,i,j+1}^H \leftarrow \mathbf{W}_{p,i,j}^H - \eta \nabla W_{p,i,j,k}^H$;
    $\mathcal{P}_{back}^T \leftarrow$ Compute backpropagation gradient $\nabla W_{p,i,j,k}^T$;
    $\mathbf{W}_{p,i,j+1}^T \leftarrow \mathbf{W}_{p,i,j}^T - \eta \nabla W_{p,i,j,k}^T$;
    Output $\mathcal{P}_{back}^T$;
Edge node $i$:
Function EdgeUpdate():
    Initialize model parameters $\mathbf{W}_{b,i}^G, \mathbf{W}_{b,i}^C, \mathbf{W}_{b,i}^E$ as $\mathbf{W}_{b,i,0}^G, \mathbf{W}_{b,i,0}^C, \mathbf{W}_{b,i,0}^E$;
    for each local epoch $r = 1, \ldots, R$ do
        for batch $k = 1, \ldots, B$ do
            $\mathcal{P}_{ford}^E \leftarrow w_{b,i}^E(\mathbf{x})$;
            $\mathbf{h}_j \leftarrow$ ServerTrmFordprop($\mathcal{P}_{ford}^E, i$);
            $\mathbf{g}_j \leftarrow w_{b,i}^G(\mathbf{z}, \mathbf{c})$;
            $\hat{\mathbf{y}}_j \leftarrow w_{b,i}^C(\mathbf{h}_j \cup \mathbf{g}_j)$;
            Calculate cross-entropy loss $L(\mathbf{y}_j, \hat{\mathbf{y}}_j)$;
            $\mathcal{P}_{back}^C \leftarrow$ Compute backpropagation gradient $\nabla W_{b,i,r,k}^C$;
            $\mathbf{W}_{b,i,r+1}^C \leftarrow \mathbf{W}_{b,i,r}^C - \eta \nabla W_{b,i,r,k}^C$;
            $\mathcal{P}_{back}^T \leftarrow$ ServerTrmBackprop ($\mathcal{P}_{back}^C, i$);
            $\mathbf{W}_{b,i,j+1}^E \leftarrow \mathbf{W}_{b,i,j}^E - \eta \nabla W_{b,i,j,k}^E$;
            $\mathbf{W}_{b,i,j+1}^G \leftarrow \mathbf{W}_{b,i,j}^G - \eta \nabla W_{b,i,j,k}^G$;
```

aggregates the Transformer layer using a multi-head attention mechanism. Proposed-3, compared to Proposed-1, does not introduce SL. The edge model employs a complete BERT-enhanced CGAN, whereas the cloud aggregates the full model parameters via the multi-head attention mechanism. Proposed-4 replaces the multi-head attention on the cloud with federated averaging [36].

TABLE I: Data distribution of three datasets

Dataset	Category	Train	Test
CSE-CIC-IDS2018	Benign	360162	183683
	DoS-Hulk	80391	40187
	DDoS-HOIC	61670	30828
	DDoS-LOIC-HTTP	43214	21607
	SSH-Bruteforce	40314	20159
	Infiltration	36275	18109
	Bot	35743	17868
	FTP-BruteForce	28009	14112
	DoS-GoldenEye	16598	8302
	DoS-SlowHTTPTest	13416	6731
	In total	**715792**	**361586**
NF-ToN-IoT	Benign	129636	32409
	Scanning	80352	20088
	XSS	52164	13041
	DDoS	43056	10764
	Password	24516	6129
	DoS	15156	3789
	Injection	14544	3636
	Backdoor	360	90
	MITM	144	36
	Ransomware	72	18
	In total	**360000**	**90000**
NF-UNSW-NB15	Benign	120000	37000
	Exploits	20509	11042
	Fuzzers	14505	7805
	Generic	10770	5790
	Reconnaissance	8337	4442
	DoS	3773	2021
	Analysis	1446	853
	Backdoor	1416	753
	Shellcode	937	490
	Worms	91	73
	In total	**181784**	**70269**

For a comprehensive comparison, three distributed learning methods are selected as baselines, as summarized in Table III. Baseline-1 is a BERT-based FL architecture in which the cloud aggregates full model parameters using a traditional attention mechanism. Baseline-2 is an LSTM-based FL architecture that also employs attention-based aggregation across the full model. Baseline-3 is a BERT-based split-federated architecture, where the Embedding and Head layers are deployed at the edge and the Transformer layer at the cloud, with parameters aggregated via federated averaging [36].

V. Results Analysis

This section compares intrusion detection performance across datasets, including precision, recall, and F1-score, as well as communication and computational costs.

A. Detection Performance Analysis

Table IV shows the weighted average accuracy, precision, recall, and F1-score of different methods on the CSE-CIC-IDS2018, NF-ToN-IoT, and NF-UNSW-NB15 test datasets.

TABLE II: Ablation classification of the proposed methods

Name	Edge-side enhancement		Cloud-side federated aggregation	
	Subsection II-B	Subsection II-A	Multi-head attention (Section III)	Weight aggregate [37]
Proposed-1	✓	✓	✓	
Proposed-2		✓	✓	
Proposed-3	✓		✓	
Proposed-4	✓	✓		✓

TABLE III: Classification of baseline methods

Name	Edge-side model			Cloud-side federated aggregation	
	BERT [12]	FedBERT [25]	LSTM [38]	Weight aggregate [37]	Federated averaging [36]
Baseline-1	✓			✓	
Baseline-2		✓	✓	✓	
Baseline-3	✓	✓			✓

Proposed-1 outperforms all other methods across all evaluation metrics, followed by Proposed-2 and Proposed-4. Compared to Proposed-2, Proposed-1 shows improvements in Accuracy, Precision, and F1-score in the three datasets, ranging from 0.7% to 1.5%, 0.4% to 0.5%, and 1.3% to 1.6%, respectively. These results indicate that integrating a CGAN into the edge model to implement a generative adversarial split-iteration training strategy effectively mitigates data imbalance. Compared to Proposed-4, Proposed-1 achieves improvements in Accuracy, Precision, and F1-score, ranging from 1.2% to 1.3%, 0.4% to 0.5%, and 0.6% to 1.1%, respectively. The traditional attention mechanism improves model performance by focusing on key regions in the input sequence, but it relies on context information provided by the encoder. In contrast, the self-attention mechanism directly models the inherent relationships among elements in the input sequence, enabling the capture of complex dependencies in traffic sequences. Furthermore, the multi-head attention mechanism adopts a parallelized design that computes self-attention matrices across multiple feature subspaces simultaneously, thereby capturing the multi-layered semantic features of traffic sequences. Proposed-1, by incorporating a federated aggregation strategy based on the multi-head attention mechanism, can train better global model parameters, further improving detection performance.

Compared to Baseline-1, Proposed-1 shows improvements in Accuracy, Precision, and F1-score in the three datasets, ranging from 1.4% to 2.3%, 0.5% to 0.7%, and 2% to 3.2%, respectively. In terms of model architecture, Proposed-1 not only leverages BERT's feature extraction but also integrates CGAN's adversarial training and introduces the multi-head attention mechanism for global parameter aggregation. These advantages significantly reduce misclassification and false negatives, thus improving the F1-score. Additionally, compared to Baseline-2, Proposed-1 achieves improvements in Accuracy, Precision, and F1-score in the three datasets, ranging from 12.1% to 15%, 12.4% to 16.4%, and 19% to 23%, respectively. BERT's self-attention mechanism enables the model to capture dependencies among distant statistical features in traffic sequences, thereby capturing complex internal information. Proposed-1 can extract more information about attack categories from BERT's high-dimensional traffic features, thereby improving classification accuracy. In contrast, LSTM has limited ability to capture long-range dependencies in traffic sequences and fails to accurately distinguish attack types when attack classes are small.

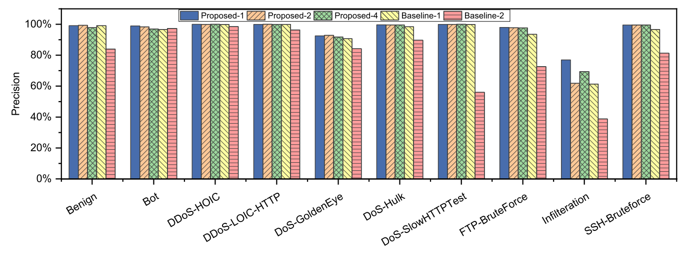
Figs. 4, 5, and 6 show the precision, recall, and F1-score for detecting normal traffic and various attack categories. The results indicate that Proposed-1 performs best across nearly all traffic categories for all three evaluation metrics. As shown in Fig. 4, in CSE-CIC-IDS2018, for the covert Infiltration attack category, Proposed-1 improves Precision by approximately 15.6% and 38.2% compared to Baseline-1 and Baseline-2. This result demonstrates that BERT can effectively capture long-range dependencies among statistical features in traffic sequences, thereby improving the detection of difficult-to-recognize attack categories. By incorporating BERT as a feature extractor, Proposed-1 effectively combines the generative adversarial process and trains global model via multi-head attention on the cloud. Compared to Proposed-2 and Proposed-4, Proposed-1 improves Precision for detecting the Infilteration attack by approximately 15.2% and 7.6%.

Similarly, in NF-ToN-IoT, for Backdoor, MITM, and Ransomware attack categories, which have only 90, 36, and 18 samples, Proposed-1 improves Precision compared to Baseline-1 and Baseline-2 by 15.2% to 78.5%, and compared to Proposed-2 and Proposed-4 by 6.9% to 30%. In NF-UNSW-NB15, for the Worms and Shellcode attack categories, with only 73 and 490 samples, Proposed-1 achieves the highest Precision among all methods, with improvements ranging from 7% to 42.7%. These results suggest that Proposed-1 can effectively mitigate misclassification in less frequent attack categories with hidden features.

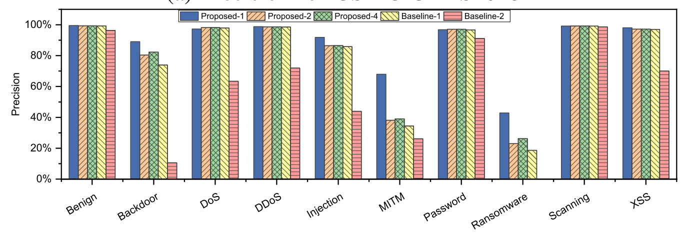
As shown in Fig. 5, compared to Baseline-2, Proposed-1 shows significant improvements in recall across all traffic types. In detecting the Infilteration category in CSE-CIC-IDS2018, Proposed-1 improves Recall compared to Proposed-4, but experiences a decrease of approximately 2.1% and 2.3% compared to Proposed-2 and Baseline-1. For the Injection attack category in the NF-ToN-IoT dataset, recall for Proposed-2, Proposed-4, and Baseline-1 is over 93%, while Proposed-1's Recall is 90.814%, which is lower than the three comparison methods. In CSE-CIC-IDS2018 and NF-ToN-IoT, for these small sample and covert attack categories, Proposed-1 exhibits a decrease in Recall of approximately 2%, but an

TABLE IV: Weighted average performance on three datasets

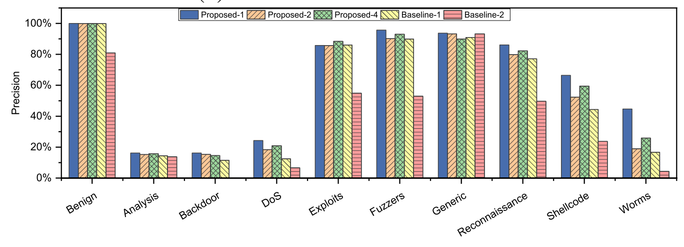
Dataset	Method	Accuracy	Precision	Recall	F1-Score
CSE-CIC-IDS2018	Proposed-1	97.806%	98.037%	97.806%	97.875%
	Proposed-2	96.340%	97.336%	96.340%	96.621%
	Proposed-4	96.511%	96.851%	96.511%	96.627%
	Baseline-1	95.532%	96.595%	95.532%	95.810%
	Baseline-2	85.743%	83.831%	85.743%	82.925%
NF-ToN-IoT	Proposed-1	98.536%	98.556%	98.535%	98.541%
	Proposed-2	98.027%	98.141%	98.027%	98.066%
	Proposed-4	98.054%	98.164%	98.054%	98.091%
	Baseline-1	97.864%	98.008%	97.864%	97.915%
	Baseline-2	83.630%	86.153%	83.630%	82.141%
NF-UNSW-NB15	Proposed-1	87.098%	91.499%	87.098%	88.748%
	Proposed-2	85.807%	90.047%	85.807%	87.169%
	Proposed-4	85.968%	90.850%	85.968%	87.642%
	Baseline-1	83.907%	89.466%	83.907%	85.774%
	Baseline-2	68.141%	68.476%	68.141%	65.377%



(a) Precision for CSE-CIC-IDS2018

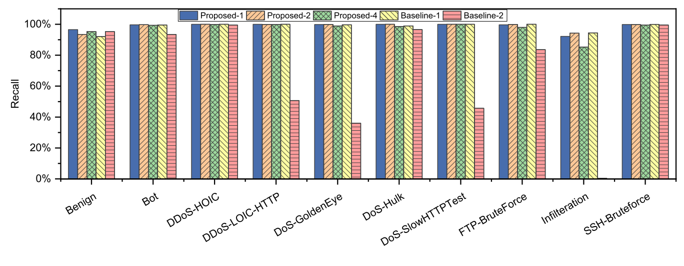


(b) Precision for NF-ToN-IoT

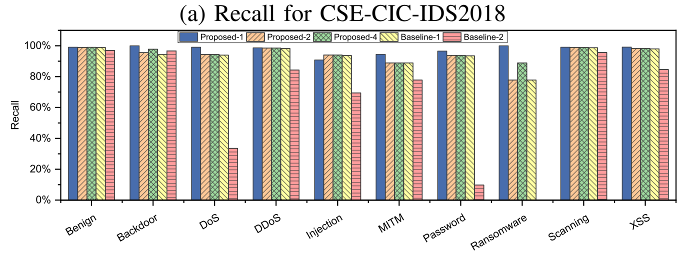


(c) Precision for NF-UNSW-NB15

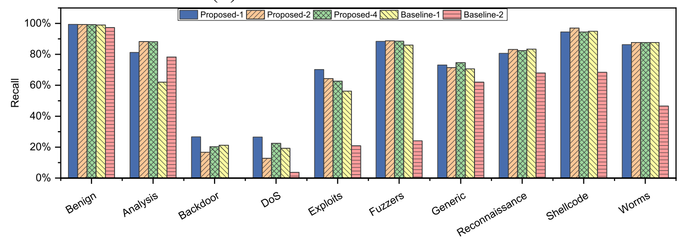
Fig. 4: Precision for benign and individual attack classes.



(a) Recall for CSE-CIC-IDS2018



(b) Recall for NF-ToN-IoT

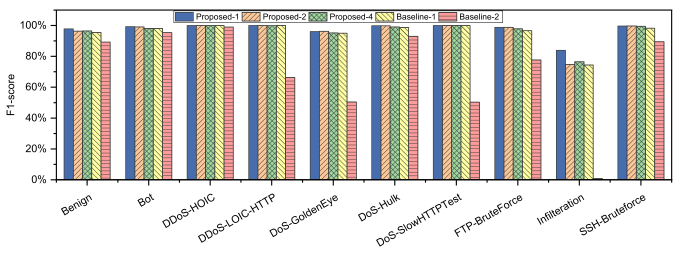


(c) Recall for NF-UNSW-NB15

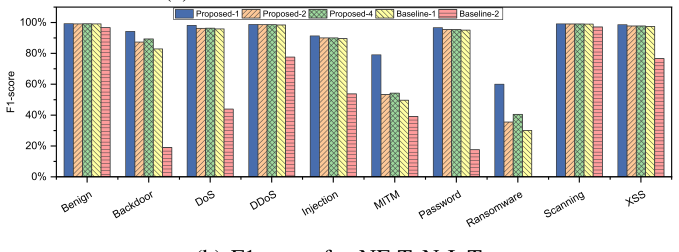
Fig. 5: Recall for benign and individual attack classes.

improvement of more than 5% in Precision. This indicates that the three comparison methods tend to misclassify other traffic categories as the attack class, thereby improving Recall but sacrificing Precision. In contrast, Proposed-1 balances Precision and Recall and reduces false positives. Similarly, for the Fuzzers, Reconnaissance, Shellcode, and Worms attack categories in NF-UNSW-NB15, the Recall of Proposed-1 is lower than that of some comparison methods by less than 3%, whereas its Precision shows a significant improvement.

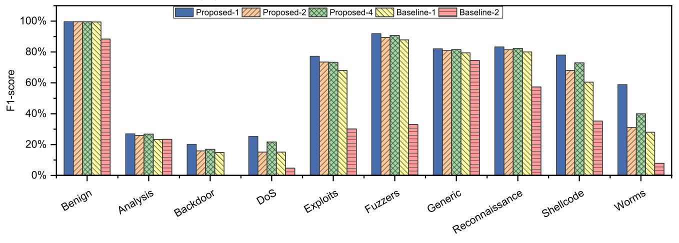
Fig. 6 presents the F1-scores in detecting normal traffic and various attack types. Proposed-1 outperforms other methods in terms of F1-score across almost all attack categories. For certain small and covert attack types, such as Infilteration in CSE-CIC-IDS2018, Backdoor, MITM, and Ransomware in the NF-ToN-IoT dataset, and Backdoor, DoS, Shellcode, and Worms in the NF-UNSW-NB15 dataset, Proposed-2 and Proposed-4 still exhibit high false positive rates and limited improvement in detection, while Proposed-1 further enhances detection performance. In NF-UNSW-NB15, which has the most imbalanced categories, Proposed-1 shows limited im-



(a) F1-score for CSE-CIC-IDS2018



(b) F1-score for NF-ToN-IoT



(c) F1-score for NF-UNSW-NB15

Fig. 6: F1-score for benign and individual attack classes.

provements in Precision for the Analysis and Generic attack types compared with Proposed-2, Proposed-4, and Baseline-1, but its recall decreases significantly or remains unchanged. In this case, the F1-score still demonstrates the superior performance of Proposed-1. Additionally, Proposed-1 shows a slight decrease of about 2% in Precision for the Exploits attack category, but its recall improves by more than 6%, and its F1-score increases by more than 3%.

B. Analysis of Split Iteration Effectiveness

In this experiment, we compared Proposed-1, which incorporates model partitioning, with Proposed-3, which does not, to evaluate the impact of partitioning on detection accuracy during collaborative training. From Fig. 7, the accuracy trends of both methods remain nearly identical as training epochs increase, indicating that model partitioning does not compromise detection performance. Moreover, both approaches exhibit stable performance with only minor fluctuations, suggesting that the split-federated BERT design maintains detection accuracy while improving computational efficiency.

Similar trends between Proposed-1 and Proposed-3 further validate that the partitioning strategy primarily optimizes resource utilization, especially in resource-constrained edge environments, while maintaining the model's detection integrity. The findings highlight the effectiveness of the federated split framework in maintaining high detection accuracy while addressing computational and communication challenges.

C. Training Cost Analysis

This section analyzes the communication cost and computational time of the federated split training strategy based on experimental results. Communication cost refers to the amount of data exchanged between the cloud and edge node i during split-federated training, while computational time reflects the model's time cost for training.

Let I denote the number of edge nodes, R the number of split iterations per global training round, and B the batch size for edge node iterations. $|\mathbf{W}|$ represents the total number of model parameters, and $\mathcal{T}(\mathbf{W})$ and $\mathcal{T}_{mul}(\mathbf{W})$ represent the computation time for the complete model and the federated aggregation time for multi-head attention. In the traditional FL mode in Proposed-3, the entire model $|\mathbf{W}|$ must be uploaded to the cloud server. Since FL involves bidirectional communication, the communication data volume is $2|\mathbf{W}|$. In contrast, in split-federated learning, only the parameters between the split layers need to be transmitted. Specifically, L^E, L^T, and L^H represent the forward propagation outputs of the Embedding, Transformer, and Head layers, respectively, while ∇L^D, ∇L^T, and ∇L^H represent the backward gradient updates for the classification network, Transformer layer, and Head layer. d^E, d^T, and d^H represent the dimensions of the Embedding, Transformer, and Head layers.

In our experiment, the total number of trainable parameters for BERT is 9,632,842, with the Transformer layer having 2 layers, a hidden dimension of 256, and 4 attention heads. Therefore, $d^E = 256$, $d^T = 256$, and $d^H = 10$. The total number of trainable parameters for CGAN is 1,404,979, and the batch size B is set to 100.

For Proposed-1, the communication data volume per split iteration for each edge node is

$$\begin{aligned} L_{11} &= L^E + L^H + \nabla L^D + \nabla L^T \\ &= d^E B + d^H B + d^H B + d^E B \\ &= 2B(d^E + d^H) \end{aligned} \tag{16}$$

The total communication data volume for Proposed-1 in one global training round is

$$L_{12} = I \cdot R \cdot 2B(d^E + d^H) = 2IRB(d^E + d^H) \tag{17}$$

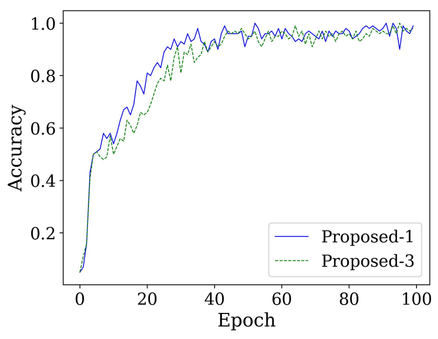
Similarly, in Baseline-3, the communication data volume per split iteration for each edge node is:

$$\begin{aligned} L_{21} &= L^E + L^T + \nabla L^H + \nabla L^T \\ &= d^E B + d^T B + d^T B + d^E B \\ &= 2B(d^E + d^T) \end{aligned} \tag{18}$$

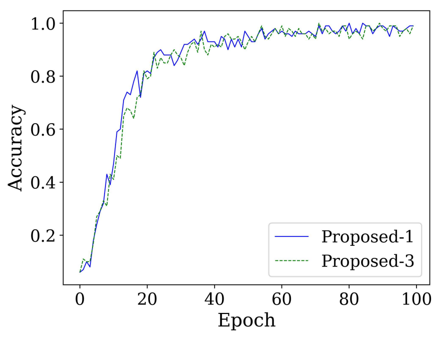
The total communication data volume for Baseline-3 in one global training round is

$$L_{22} = I \cdot R \cdot (2d^E B + 2d^T B) = 2IRB(d^E + d^T) \tag{19}$$

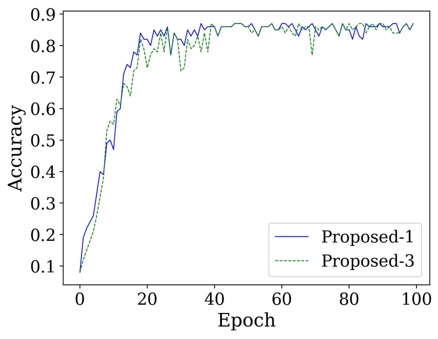
As shown in Fig. 8, when applying the relevant experimental data to the formulas in Table V, it is evident that as R increases, the communication cost for Proposed-1 remains consistently lower than that for Proposed-3 and Baseline-3. Consequently, the split-BERT strategy in Proposed-1 reduces communication costs while maintaining stable detection



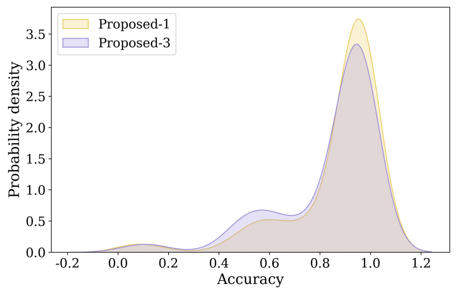
(a) Accuracy changes on CSE-CIC-IDS2018



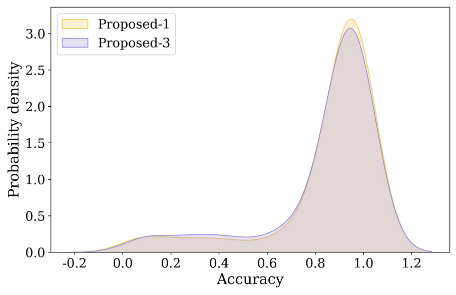
(b) Accuracy variations on NF-ToN-IoT



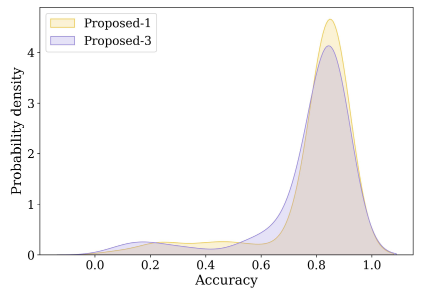
(c) Accuracy variations on NF-UNSW-NB15



(d) Accuracy PDF on CSE-CIC-IDS2018



(e) Accuracy PDF on NF-ToN-IoT



(f) Accuracy PDF on NF-UNSW-NB15

Fig. 7: Impact of model splitting on accuracy and probability density function (PDF).

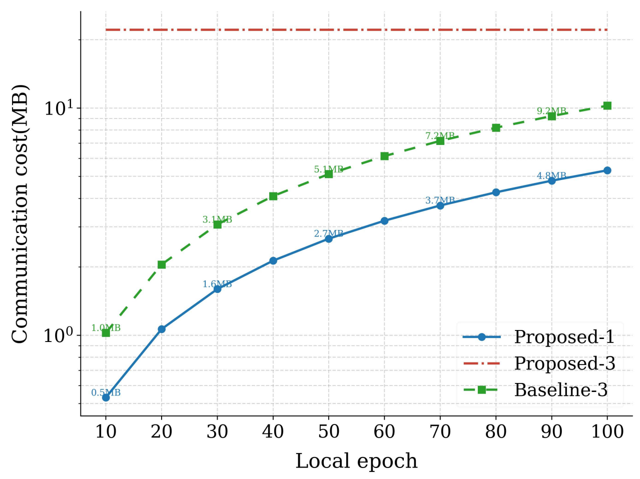


Fig. 8: Communication cost of one global training.

TABLE V: Communication cost of one global training

Model	Edge node	Cloud server
Proposed-1	$2RB(d^E + d^H)$	$2IRB(d^E + d^H)$
Proposed-3	$2\|\mathbf{W}\|$	$2I\|\mathbf{W}\|$
Baseline-3	$2RB(d^E + d^T)$	$2IRB(d^E + d^T)$

TABLE VI: Computation cost of one global training

Model	Edge node	Cloud server
Proposed-1	$\mathcal{T}(\mathbf{W}_b^G) + \mathcal{T}(\mathbf{W}_b^C) + \mathcal{T}(\mathbf{W}_b^E)$	$\mathcal{T}(\mathbf{W}_p^T) + \mathcal{T}(\mathbf{W}_p^H) + \mathcal{T}_{mul}(\mathbf{W}_p^T)$
Proposed-3	$\mathcal{T}(\mathbf{W})$	$\mathcal{T}_{mul}(\mathbf{W})$
Baseline-3	$\mathcal{T}(\mathbf{W}_b^E) + \mathcal{T}(\mathbf{W}_p^H)$	$\mathcal{T}(\mathbf{W}_p^T) + \mathcal{T}_{mul}(\mathbf{W}_p^T)$

performance. For computational costs, $\mathcal{T}(\mathbf{W}_b^E)$, $\mathcal{T}(\mathbf{W}_p^T)$, $\mathcal{T}(\mathbf{W}_p^H)$, $\mathcal{T}(\mathbf{W}_b^G)$, and $\mathcal{T}(\mathbf{W}_b^D)$ represent the computation times for w_b^E, w_p^T, w_p^H, w_b^G, and w_b^C in one global training round. $\mathcal{T}_{mul}(\mathbf{W}_p^T)$ represents the federated aggregation time for w_p^T based on the multi-head attention mechanism in one global training round, and $\mathcal{T}_{avg}(\mathbf{W}_p^T)$ represents the federated averaging time for w_p^T in one global training round.

For Baseline-3, the computation cost for an edge node is represented as

$$\mathcal{T}(\mathbf{W}) = \mathcal{T}(\mathbf{W}_p^T) + \mathcal{T}(\mathbf{W}_p^H) + \mathcal{T}(\mathbf{W}_b^G) + \mathcal{T}(\mathbf{W}_b^D) + \mathcal{T}(\mathbf{W}_b^E) \tag{20}$$

In Proposed-1, the computation time for the top-tier model is

$$\mathcal{T}_1(\mathbf{W}_p) = \mathcal{T}(\mathbf{W}_p^T) + \mathcal{T}(\mathbf{W}_p^H) \tag{21}$$

For the edge node in Proposed-1, the computation time is:

$$\mathcal{T}_1(\mathbf{W}_b) = \mathcal{T}(\mathbf{W}_b^G) + \mathcal{T}(\mathbf{W}_b^D) + \mathcal{T}(\mathbf{W}_b^E) \tag{22}$$

In Baseline-3, the computation time for the top-tier model is

$$\mathcal{T}_2(\mathbf{W}_p) = \mathcal{T}(\mathbf{W}_p^T) \tag{23}$$

For the edge node in Baseline-3, the computation time is:

$$\mathcal{T}_2(\mathbf{W}_b) = \mathcal{T}(\mathbf{W}_b^E) + \mathcal{T}(\mathbf{W}_p^H) \tag{24}$$

From Tables V and VI, since w_p^T is much larger than w_b^E, w_p^H, w_b^G, and w_b^C, $\mathcal{T}(\mathbf{W}_p^T) \gg \mathcal{T}(\mathbf{W}_b^E) + \mathcal{T}(\mathbf{W}_p^H)$ and $\mathcal{T}(\mathbf{W}_p^T) \gg \mathcal{T}(\mathbf{W}_b^G) + \mathcal{T}(\mathbf{W}_b^D)$, thus $\mathcal{T}_1(\mathbf{W}_b) \ll \mathcal{T}(\mathbf{W})$ and $\mathcal{T}_2(\mathbf{W}_b) \ll \mathcal{T}(\mathbf{W})$. The computation cost of Proposed-1 and Baseline-3 at edge nodes is lower than that of Proposed-3.

VI. Conclusion

In this study, we propose a split-federated BERT framework with adversarial training and multi-head attention aggregation for edge intrusion detection. The framework partitions BERT across edge and cloud, enabling collaborative training while reducing communication overhead. The BERT model serves as the traffic classifier, while a CGAN enhances local learning at the edge. The combination of edge-side adversarial enhancement and cloud-side attention-based aggregation improves detection performance while maintaining controlled computation and communication costs. Experiments on multiple datasets demonstrate consistent improvements over baseline methods in accuracy, precision, recall, and F1-score. The framework is applicable to resource-constrained distributed networks such as IoT and edge-based monitoring systems.

References

[1] J. Azimjonov and T. Kim, "A comprehensive empirical analysis of data sets, regression-based feature selectors, and linear SVM classifiers for intrusion detection systems," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34 676–34 693, 2024.

[2] T. Wang, Q. Liu, H. Shen, X. Luo, and G. Bai, "Graph neural network-enhanced auxiliary classifier generative adversarial network framework for robust intrusion detection," *ETRI J.*, 2025, to be published, DOI: 10.4218/etrij.2025-0152.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Adv. Neural Inf. Process.*, vol. 30, pp. 5998–6008, 2017.

[4] H. Shen, X. Li, X. Wang, Y. Dai, T. Wang, and G. Bai, "LLM-augmented contrastive learning for misinformation detection in social networks," *IEEE Trans. Comput. Soc. Syst.*, vol. 13, no. 1, pp. 999–1014, 2026.

[5] C. Liu, K. H. Hettige, Q. Xu, C. Long, S. Xiang, G. Cong, Z. Li, and R. Zhao, "ST-LLM+: Graph enhanced spatio-temporal large language models for traffic prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 37, no. 8, pp. 4846–4859, 2025.

[6] T. Cui, X. Lin, S. Li, M. Chen, Q. Yin, Q. Li, and K. Xu, "TrafficLLM: Enhancing large language models for network traffic analysis with generic traffic representation," *arXiv preprint arXiv:2504.04222*, 2025.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Bidirectional encoder representations from transformers," *arXiv preprint arXiv:1810.04805*, vol. 15, 2018.

[8] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?" in *Proc. 57th Annu. Meet. Assoc. Comput. Linguistics*, 2019, pp. 3651–3657.

[9] M. Cho, Y. Kwon, S. Ahn, S. Kwon, and S. Cho, "A prototypical alignment approach to unknown traffic classification using BERT," *Comput. Networks*, p. 112062, 2026.

[10] Z. Cao, X. Liu, Z. Zhou, L. Ding, and W. Shang, "KD-BERT: A lightweight knowledge distillation bidirectional encoder representations from transformers for IoT network intrusion detection," *IEEE Trans. Ind. Informatics*, vol. 21, no. 11, pp. 8475–8483, 2025.

[11] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf.*, 2022, pp. 633–642.

[12] F. Li, H. Shen, J. Mai, T. Wang, Y. Dai, and X. Miao, "Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection," *Peer-to-Peer Netw. Appl.*, vol. 17, no. 1, pp. 227–245, 2024.

[13] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 8, 2022, pp. 8485–8493.

[14] H. Shen, Y. Yao, T. Wang, and G. Bai, "MobiFormer: Split-federated transfer learning for drone RAN slicing with multi-head attention," *IEEE Trans. Mob. Comput.*, 2025, to be published, DOI: 10.1109/TMC.2025.3627068.

[15] D. Yang, W. Zhang, Q. Ye, C. Zhang, N. Zhang, C. Huang, H. Zhang, and X. Shen, "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5162–5178, 2023.

[16] H. Shen, Y. Zhou, T. Wang, Y. Zhang, J. Huang, G. Bai, and X. Miao, "Blockchain-assisted cross-silo graph federated learning for network intrusion detection," in *Proc. IEEE Global Blockchain Conf. (GBC)*, 2025, pp. 1–8.

[17] R. Lee, M. Kim, D. Li, X. Qiu, T. Hospedales, F. Huszár, and N. Lane, "FedL2P: Federated learning to personalize," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 14 818–14 836, 2023.

[18] J. Li, X. Tong, J. Liu, and L. Cheng, "An efficient federated learning system for network intrusion detection," *IEEE Syst. J.*, vol. 17, no. 2, pp. 2455–2464, 2023.

[19] M. Yang, S. Liu, J. Xu, G. Tan, C. Li, and L. Song, "Achieving privacy-preserving cross-silo anomaly detection using federated XGBoost," *J. Franklin Inst.*, vol. 360, no. 9, pp. 6194–6210, 2023.

[20] J. Shen, N. Cheng, X. Wang, F. Lyu, W. Xu, Z. Liu, K. Aldubaikhy, and X. Shen, "RingsFL: An adaptive split federated learning towards taming client heterogeneity," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5462–5478, 2023.

[21] T. Che, J. Liu, Y. Zhou, J. Ren, J. Zhou, V. Sheng, H. Dai, and D. Dou, "Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2023, pp. 7871–7888.

[22] D.-J. Kim, N. B. Amma, and V. Sarveshwaran, "A novel split learning-based consumer electronics network traffic anomaly detection framework for smart city environment," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 4197–4204, 2024.

[23] A. Abedi and S. S. Khan, "FedSL: Federated split learning on distributed sequential data in recurrent neural networks," *Multimedia Tools Appl.*, vol. 83, no. 10, pp. 28 891–28 911, 2024.

[24] Y. Liao, Y. Xu, H. Xu, L. Wang, Z. Yao, and C. Qiao, "MergeSFL: Split federated learning with feature merging and batch size regulation," in *Proc. IEEE 40th Int. Conf. Data Eng. (ICDE)*, 2024, pp. 2054–2067.

[25] Y. Tian, Y. Wan, L. Lyu, D. Yao, H. Jin, and L. Sun, "FedBERT: When federated learning meets pre-training," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–26, 2022.

[26] R. Deng, X. Du, Z. Lu, Q. Duan, S.-C. Huang, and J. Wu, "HSFL: Efficient and privacy-preserving offloading for split and federated learning in iot services," in *Proc. IEEE Int. Conf. Web Serv. (ICWS)*, 2023, pp. 658–668.

[27] Z. Cheng, X. Xia, M. Liwang, X. Fan, Y. Sun, X. Wang, and L. Huang, "CHEESE: Distributed clustering-based hybrid federated split learning over edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 12, pp. 3174–3191, 2023.

[28] G. Singh, K. Sood, P. Rajalakshmi, D. D. N. Nguyen, and Y. Xiang, "Evaluating federated learning-based intrusion detection scheme for next generation networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 4, pp. 4816–4829, 2024.

[29] C. Park, J. Lee, Y. Kim, J.-G. Park, H. Kim, and D. Hong, "An enhanced AI-based network intrusion detection system using generative adversarial networks," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2330–2345, 2022.

[30] H. Ding, Y. Sun, N. Huang, Z. Shen, and X. Cui, "TMG-GAN: Generative adversarial networks-based imbalanced learning for network intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1156–1167, 2023.

[31] M. G. Constantin, D.-C. Stanciu, L.-D. Ştefan, M. Dogariu, D. Mihăilescu, G. Ciobanu, M. Bergeron, W. Liu, K. Belov, O. Radu *et al.*, "Exploring generative adversarial networks for augmenting network intrusion detection tasks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 21, no. 1, pp. 1–19, 2024.

[32] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, "CGAN-based collaborative intrusion detection for UAV networks: A blockchain-empowered distributed federated learning approach," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 120–132, 2022.

[33] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[34] Q. Zeng, S. Olatunde-Salawu, and F. Nait-Abdesselam, "FGA-IDS: A federated learning and gan-augmented intrusion detection system for UAV networks," in *Proc. IEEE 10th Int. Conf. Collabor. Internet Comput. (CIC)*, 2024, pp. 50–59.

[35] M. Chen, B. Wu, H. Sun, and Z. Wang, "FedGAN-ID: Federated learning-based intrusion detection for in-vehicle network using GANs," *IEEE Internet Things J.*, vol. 12, no. 17, pp. 36 155–36 167, 2025.

[36] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "FedAvg with fine tuning: Local updates lead to representation learning," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 10 572–10 586, 2022.

[37] Z. Li, T. Lin, X. Shang, and C. Wu, "Revisiting weighted aggregation in federated learning with neural networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 19 767–19 788.

[38] H. Aydın, Z. Orman, and M. A. Aydın, "A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment," *Comput. Secur.*, vol. 118, p. 102725, 2022.

Hang Shen (Member, IEEE) received the Ph.D. degree with honors in Computer Science from the Nanjing University of Science and Technology, Nanjing, China, in 2015. He worked as a Full-Time Postdoctoral Fellow with the Broadband Communications Research (BBCR) Lab, Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON, Canada, from 2018 to 2019. He is an Associate Professor with the Department of Computer Science and Technology at Nanjing Tech University, Nanjing, China. His research interests include pre-trained large models and embodied intelligence for cybersecurity and autonomous driving. He serves as an Associate Editor for *Journal of Information Processing Systems*, *Frontiers in Blockchain*, and IEEE ACCESS, and was a Guest Editor for *Peer-to-Peer Networking and Applications*. He is/was a TPC Member for the IEEE Global Communications Conference (GLOBECOM), the IEEE International Conference on High Performance Computing and Communications (HPCC), the Annual International Conference on Privacy, Security and Trust (PST), the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), and the Annual Canadian Conference on Electrical and Computer Engineering (CCECE). He is a member of the IEEE Computer Society, the Communications Society, and the Vehicular Technology Society, an Executive Committee Member of the ACM Nanjing Chapter, and a CCF Senior Member.

Qi Liu received the B.Eng. degree in Computer Science from Shenyang Ligong University, Shenyang, China, in 2023. He is now pursuing his M.Eng. degree in Computer Science at Nanjing Tech University, Nanjing, China. His research interests include dynamic neural networks and large language models in cybersecurity.

Fang Li received the B.Eng. and M.Eng. degrees in Computer Science from Nanjing Tech University, Nanjing, China, in 2022 and 2025, respectively. Her research interests include pre-trained language models and split-federated learning for cybersecurity.

Tianjing Wang (Member, IEEE) holds a B.Sc. in Mathematics from Nanjing Normal University in 2000, an M.Sc. in Mathematics from Nanjing University in 2002, and a Ph.D. in Signal and Information System from the Nanjing University of Posts and Telecommunications (NUPT) in 2009. From 2011 to 2013, she was a Postdoctoral Fellow with the School of Electronic Science and Engineering at NUPT. She was a Visiting Scholar with the Electrical and Computer Engineering Department at the State University of New York at Stony Brook from 2013 to 2014. She is an associate professor in the Department of Communication Engineering at Nanjing Tech University. Her research interests include vehicular networks, cybersecurity and blockchain.

Yuanfei Dai received the Ph.D. degree in Computer Science from Fuzhou University, Fuzhou, China, in 2021. From 2019 to 2020, he was a Visiting Student at Brown University, Providence, RI, USA. He is currently a Lecturer at Nanjing Tech University, Nanjing, China. His research interests include knowledge acquisition and knowledge graph representation. He, together with his student, received the Best Student Paper Award at the 2023 International Conference on Knowledge Science, Engineering and Management (KSEM).

Guangwei Bai received the B.Eng. and M.Eng. degrees in computer engineering from Xi'an Jiaotong University, Xi'an, China, in 1983 and 1986, respectively, and the Ph.D. degree in Computer Science from the University of Hamburg, Hamburg, Germany, in 1999. From 1999 to 2001, he worked as a Research Scientist at the German National Research Center for Information Technology, Bonn, Germany. In 2001, he joined the University of Calgary, Calgary, AB, Canada, as a Research Associate. Since 2005, he has been working at Nanjing Tech University, Nanjing, China, as a Professor in Computer Science. From October to December 2010, he was a visiting professor in the Department of Electrical and Computer Engineering at the University of Waterloo, Waterloo, ON, Canada. His research interests include wireless networking and cybersecurity.