Received: 08 July 2026; **Revised:** 14 August 2026; **Accepted:** 14 September 2026; **Published:** 20 November 2026
Archs Sci. (2026) Volume 76, Issue 5 Pages 69-80, Paper ID 2026507.
https://doi.org/10.68304/as/76507

Secure Network Attack Detection in Federated Learning via Attention-Enhanced Graph Neural Networks

Xide Wang[1] and Qingyan Fang[2,*]
[1]School of Artificial Intelligence, Harbin University, Harbin 150086, Heilongjiang, China
[2]School of Electrical and Intelligent Engineering, Harbin Guangsha College, Harbin 150001, Heilongjiang, China
Correspondence: miyeerf@126.com

Abstract Federated learning (FL) enables decentralized model training while preserving data privacy, but it remains vulnerable to network attacks that threaten system integrity. This paper proposes an effective network attack detection method for FL using graph neural networks (GNNs) with an attention mechanism. Devices and their communications are modeled as a graph, allowing the GNN to capture structural relationships and spatial dependencies. An attention mechanism is introduced to prioritize critical nodes and improve detection performance. The model is trained in a privacy-preserving manner using federated learning protocols. Experimental results in MATLAB demonstrate that the proposed approach significantly outperforms traditional methods in detection accuracy, convergence speed, and robustness under various network conditions. The method is intended to enhance the security of FL systems while maintaining privacy.

Index Terms Federated learning, network attack detection, graph neural networks, attention mechanism, privacy preservation

I. Introduction

Federated learning represents an important paradigm in machine learning, specifically tailored to address the intricacies of privacy and data ownership within distributed network environments. Departing from conventional centralized machine learning methodologies where data aggregation occurs on a central server, posing potential threats to individual privacy and data security, federated learning provides a more privacy-centric and streamlined alternative [1], [2].

In the context of federated learning, the conventional approach of transferring data to a central server gives way to a distributed model where individual devices or nodes actively engage in the learning process. These entities train local models on their respective datasets, effectively encapsulating the unique knowledge and insights intrinsic to each device. The culmination of this process involves the amalgamation of these locally trained models to craft a global model, synthesizing the collective intelligence amassed from all participating nodes. This decentralized methodology serves as a robust safeguard, ensuring that data stays localized, thereby mitigating the risks associated with privacy breaches and inadvertent data leaks [1], [3].

A central advantage of federated learning lies in its commitment to preserving data privacy. Unlike traditional centralized approaches where data is aggregated and stored on a central server, prompting concerns regarding the privacy and security of sensitive information, federated learning guarantees that data remains decentralized on individual devices. This not only diminishes the exposure of individual data but also endows participants with the autonomy to dictate the utilization and dissemination of their data. This empowerment fosters a sense of ownership and trust, as individuals wield greater influence over the fate of their data [4], [5].

Furthermore, federated learning exhibits pronounced advantages in terms of communication efficiency and cost reduction. In centralized models, the imperative to transfer substantial volumes of data to a central server imposes considerable bandwidth and communication prerequisites. Contrastingly, federated learning minimizes the necessity for data transfer by conducting model training locally on client devices. This approach substantially curtails communication overhead, resulting in diminished bandwidth demands and reduced costs, particularly in scenarios characterized by limited or expensive network connectivity.

The utilization of local devices' computational capabilities propels federated learning towards enhanced scalability. It accommodates extensive and heterogeneous datasets without necessitating data centralization. This attribute proves particularly invaluable in distributed environments featuring diverse data sources, such as edge computing networks or federated databases. The capacity to adeptly process and glean insights from massive datasets positions federated learning as a versatile and pragmatic solution across various domains and real-world applications, as illustrated in Figure 1.

In summary, federated learning emerges as a multifaceted solution, excelling in the realms of data privacy, ownership, communication efficiency, and scalability. Through the decentralization of the learning process and the preservation of data on local devices, federated learning fortifies privacy measures and curtails the risks of data breaches. Concurrently, it provides participants with the authority to govern the fate of their data. The diminished need for data transfer amplifies communication efficiency and slashes costs. With its heightened scalability, federated learning stands out as an appealing and promising avenue for distributed machine learning in environments where privacy is of paramount importance [6], [7].

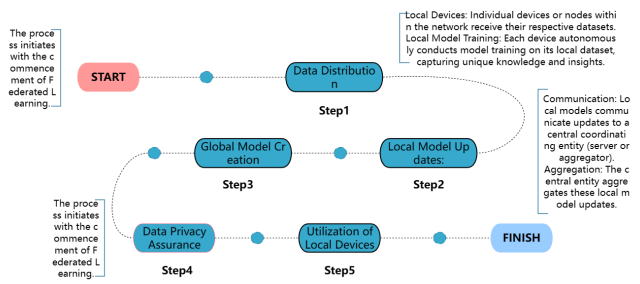


Figure 1: Decentralized learning for enhanced privacy and efficiency

II. Security Challenges in Federated Learning

While federated learning offers numerous advantages in terms of privacy preservation and data ownership, it also presents unique security challenges that need to be addressed. In this section, we explore the key security challenges that arise in the context of federated learning.These challenges are summarized in Figure 2.

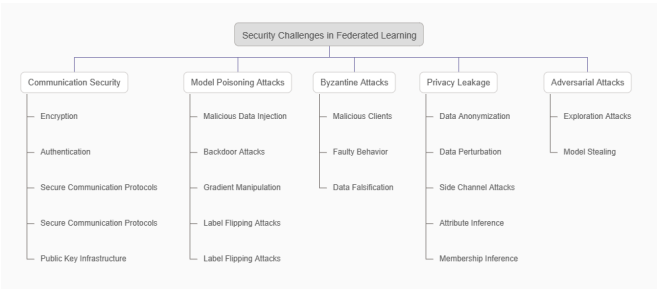


Figure 2: Security challenges in federated learning

A. Communication Security

Communication security plays a pivotal role in the robust implementation of federated learning, particularly given the exchange of local models between client devices and the server. The inherent risk of eavesdropping, tampering, or interception of model updates and sensitive information underscores the imperative of fortifying communication channels against potential adversarial threats. For illustrative purposes, these threat probabilities can be summarized by the following expression (which is not a probability of their union):

$$\begin{aligned} Risk = {} & P(\text{eavesdropping}) \\ & + P(\text{tampering}) + P(\text{interception}). \end{aligned} \tag{1}$$

Here, P(eavesdropping), P(tampering), and P(interception) represent the probabilities of each respective security threat occurring [8], [9].

To counter these threats, secure communication protocols are essential. Encryption and authentication mechanisms become integral components in the formulation of these protocols. Mathematically, the encryption process can be represented as:

$$\begin{aligned} Encrypted_{Data} = {} & Encrypted(\text{Original}_{Data}, \\ & \text{Encryption}_{Key}). \end{aligned} \tag{2}$$

This equation encapsulates the transformation of original data (Original_Data) into encrypted data (Encrypted_Data) using a specific encryption key (Encryption_Key).

Similarly, authentication mechanisms play a crucial role in verifying the legitimacy of the communicating entities. A mathematical representation of authentication can be expressed as:

$$\begin{aligned} & Authentication_{Result} \\ & = \text{Authenticate}(\text{Received}_{Data}, \text{Authentication}_{Key}). \end{aligned} \tag{3}$$

Here, Received_Data undergoes authentication using a designated key (Authentication_Key), yielding an authentication result (Authentication_Result).

In essence, these mathematical representations underscore the technical intricacies involved in ensuring secure communication within the federated learning framework, providing a foundation for the protection of data confidentiality and integrity [10], [11].

B. Model Poisoning Attacks

Model poisoning attacks pose a significant threat to the integrity and reliability of federated learning. In such attacks, adversaries intentionally inject malicious data or manipulate the model updates to compromise the quality of the global model. By strategically crafting or modifying the data used for training, adversaries can introduce biases, degrade the performance of the model, or even manipulate the model to produce incorrect predictions. Detecting and mitigating model poisoning attacks is crucial to ensure the trustworthiness of the federated learning process.

C. Byzantine Attacks

In the landscape of federated learning, Byzantine attacks emerge as a formidable challenge, denoting malicious behaviors exhibited by certain clients within the federated learning setting. These clients intentionally inject incorrect or misleading model updates, thereby disrupting the learning process and

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

compromising the quality of the global model. Mathematically, the impact of Byzantine attacks on the global model's quality (Q_{global}) can be expressed as:

$$Q_{\text{global}} = Q_{\text{ideal}} + \sum_{\text{i=1}}^{\text{n}} \text{AttackEffect}_{\text{i}}. \quad (4)$$

Here, Q_{ideal} represents the ideal quality of the global model, and AttackEffect_i denotes the impact of the Byzantine attack introduced by the i-th client.

Mitigating Byzantine attacks necessitates the development of robust mechanisms for detection and response. A potential mechanism can be formulated as:

$$\text{MitigationScore} = \text{DetectByzantine}(\text{Model}_{\text{Updates}}). \quad (5)$$

This equation calculates a mitigation score based on the analysis of model updates, aiding in the identification of Byzantine behaviors.

D. Privacy Leakage

Although federated learning aims to preserve data privacy, there is still a risk of privacy leakage during the learning process. Unintended information leakage can occur through the model updates or other intermediate data exchanges. Even if the raw data remains on the client devices, it is possible to infer sensitive information from the patterns or gradients present in the exchanged data. Protecting against privacy leakage requires the careful design of privacy-preserving techniques, such as differential privacy or secure aggregation, to ensure that no private information is inadvertently disclosed [12], [13].

E. Adversarial Attacks on federated learning Infrastructure

Adversaries may also target the underlying infrastructure of federated learning to compromise its security. They can exploit vulnerabilities in the client devices, the server, or the communication channels to gain unauthorized access, disrupt the learning process, or steal sensitive information. Ensuring the security of the entire federated learning ecosystem, including secure device configurations, robust server defenses, and secure communication protocols, is crucial to prevent adversarial attacks and maintain the overall security of the system.

Addressing these security challenges requires a comprehensive approach that combines secure communication protocols, robust anomaly detection mechanisms, privacy-preserving techniques, and defenses against adversarial attacks. Furthermore, the development of advanced detection algorithms and models, such as the proposed method in this paper using graph neural networks with an attention mechanism, can significantly enhance the security and resilience of federated learning systems against network attacks [14], [15].

In the following sections, we will explore the proposed method in detail, experiment with real-world datasets, and evaluate its performance in detecting network attacks in federated learning. By addressing these security challenges, we aim to provide a more secure and trustworthy environment for collaborative machine learning in distributed settings.

III. Related Work

A. Traditional Network Attack Detection Methods

Network intrusion detection has been a longstanding research area in the field of cybersecurity. Various traditional methods have been developed to detect and mitigate network attacks in centralized settings. These methods typically rely on analyzing network traffic, monitoring system logs, or using signature-based approaches. In this section, we discuss some of the commonly used traditional network attack detection methods [16].

One widely used approach is anomaly-based detection, which involves identifying deviations from normal network behavior. Statistical techniques, such as clustering, outlier detection, or machine learning algorithms, are applied to identify network anomalies that might indicate a potential attack. However, traditional anomaly-based detection methods often struggle with detecting novel or sophisticated attacks that do not conform to previously seen patterns.

Another approach is signature-based detection, where predefined attack patterns or signatures are used to identify known attacks. These signatures can be based on specific network packet sequences, payload contents, or known attack patterns. Signature-based detection methods can effectively identify known attacks but are limited to detecting only those attacks for which signatures have been defined [17], [18].

Furthermore, the utilization of rule-based methods involves the application of predefined rules or patterns that articulate known malicious activities. Typically crafted by cybersecurity experts, these rules serve as a manual guide for matching against network traffic or system events. While effective in identifying specific attack patterns, rule-based methods encounter challenges when confronted with novel or evolving attack techniques, as they may lack the adaptability to recognize emerging threats.

B. graph neural networks

graph neural networks (GNNs) have emerged as an effective approach for modeling and analyzing graph-structured data, rendering them particularly adept at detecting network attacks. GNNs directly operate on graph data, enabling them to encapsulate intricate relationships and dependencies between nodes within a network. This inherent capability positions GNNs as highly versatile for tasks such as node classification, link prediction, and anomaly detection in network graphs [19].

In the context of network attack detection, GNNs offer several advantages. They can effectively capture both local and global structural information within a network, allowing for a comprehensive understanding of the network topology. By aggregating and propagating information across the graph, GNNs can learn representations that encode the structural

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

patterns associated with normal network behavior and detect deviations that might indicate network attacks.

GNNs have been successfully applied in various domains, including social network analysis, recommendation systems, and bioinformatics. In network security, GNNs have shown promising results in detecting intrusions, identifying botnet activities, and classifying malicious domains. By leveraging the power of GNNs, we can potentially improve the accuracy and robustness of network attack detection in the federated learning setting [20], [21].

While GNNs have demonstrated their effectiveness, there are still challenges to overcome when applying them to network attack detection. One challenge is the scalability of GNNs to large-scale networks with millions of nodes and edges. Efficient graph sampling and aggregation techniques need to be developed to handle the computational and memory requirements of GNNs in such scenarios. Additionally, adapting GNNs to the federated learning setting introduces new challenges, such as addressing the privacy and communication constraints inherent in federated environments.

In the following sections, we will present our proposed method that utilizes GNNs with an attention mechanism for detecting network attacks in the federated learning setting. By leveraging the strengths of GNNs, we aim to enhance the detection capabilities and address the unique challenges of network attack detection in the federated learning context.

IV. Algorithm Description

Figure 3 systematically demonstrates the federated deployment process of SFGNN model with adaptive hyperparameter tuning mechanism based on Bayesian optimization. The framework effectively addresses the three major challenges of non-IID data, model generalizability, and privacy protection, and has good scalability and practicality for network intrusion detection, social graph modeling, financial fraud recognition, and other scenarios.

Figure 3a shows the overall architecture of a separated federated graph neural network (SFGNN) model. The system consists of a central server and multiple clients, each of which holds a local non-independent identically distributed (non-IID) graph dataset $(D_1, D_2, ..., D_I)$ and independently trains its graph neural network model locally to generate corresponding Node Embeddings.

Each round of federated iteration process is as follows:

Client i Constructs the graph structure $G_i = (V_i, E_i)$ with its local map data D_i and trains the local GNN model F_i to obtain the embedding representation:

$$H_i = \mathrm{GNN}(G_i, X_i), \tag{6}$$

where X_i is the node feature matrix and H_i is the node embedding.

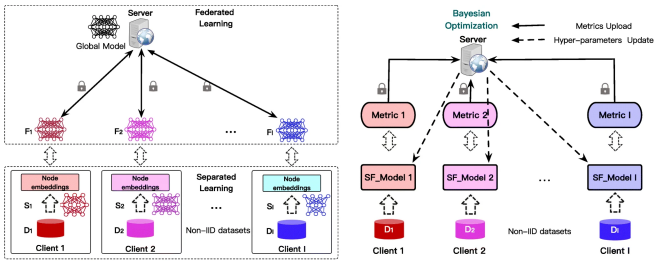
Local model output (not raw data) is uploaded to the server for global aggregation (methods such as FedAvg can be used):

$$F_{\text{global}}^{(t)} = \sum_{i=1}^{N} \frac{n_i}{n} F_i^{(t)}, \tag{7}$$

where n_i is the amount of data for client i, $n = \sum_i n_i$.

The icon of a lock is used in Figure 3a to emphasize the privacy protection during transmission, e.g., the use of mechanisms such as Secure Aggregation, Differential Privacy, etc., to ensure that co-training is accomplished without disclosing local data.

Figure 3b illustrates the introduction of Bayesian Optimization (BO) on top of the SFGNN framework, which is used to automatically optimize the hyper-parameters (e.g., learning rate, hidden dimensions, number of attentional heads, etc.) of each client model in a federated scenario.



(a) Separated federated graph neural network (SFGNN) model

(b) Hyper-parameters Bayesian optimization of SFGNN

Figure 3: Framework of the separated federated graph neural network (SFGNN) with Bayesian hyperparameter optimization (a) The architecture of the separated federated graph neural network (SFGNN), where each client independently trains a local GNN on its non-IID graph data and shares model updates with the central server without exposing raw data. (b) A federated Bayesian optimization mechanism applied to adaptively fine-tune client-specific hyperparameters. Clients upload performance metrics to the server, which selects optimal configurations via Gaussian process-based acquisition functions, and sends updates back to improve model generalization

Client i trains SFGNN model SF_{Modeli} using local data D_i and current hyperparameter configuration θ_i and calculates performance evaluation Metric_i based on task metrics (e.g., accuracy rate, F1 value).

Each client uploads evaluation metrics to the server to construct a mapping of hyperparameters to performance:

$$y_i = f(\theta_i) + \epsilon_i, \tag{8}$$

where f is the performance function and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian noise.

The server estimates the posterior distribution of the current hyperparameters by Gaussian Process (GP) and selects the next batch of hyperparameter configurations θ_i' for client updates.

Hyperparameters are updated using a Bayesian optimization criterion (e.g., Expected Improvement):

$$\theta^* = \arg\max_{\theta} \mathbb{E}\left[max(0, f(\theta) - f(\theta_{\text{best}}))\right]. \tag{9}$$

The new hyperparameters are sent down to the client for the next round of training and evaluation, forming a closed-loop optimization.

1) SFGNN Local Model Representation

$$\mathbf{H}_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} \alpha_{uv}^{(l)} \cdot \mathbf{W}^{(l)} \cdot \mathbf{H}_u^{(l)}\right), \tag{10}$$

where $N(v)$ is the neighbor of node v; α_{uv} is the attention mechanism score, defined as follows:

$$\alpha_{uv} = \frac{exp\left(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_u||\mathbf{W}\mathbf{h}_v])\right)}{\sum_{k\in\mathcal{N}(v)} exp\left(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_k||\mathbf{W}\mathbf{h}_v])\right)}. \tag{11}$$

2) Federal Aggregation Model (FedAvg)

$$F_{\text{global}}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{n} F_i^{(t)}. \tag{12}$$

3) Bayesian Optimization Process Modeling

Given the current observation point $\mathcal{D}_{1:t} = \{(\theta_i, y_i)\}_{i=1}^{t}$, model the Gaussian process:

$$f(\theta) \sim \mathcal{GP}(m(\theta), k(\theta, \theta')). \tag{13}$$

Select a new hyperparameter configuration:

$$\theta_{t+1} = \arg\max_{\theta} \text{EI}(\theta \mid \mathcal{D}_{1:t}). \tag{14}$$

As shown in Figure 4, the Meta-FedGCN framework includes three main components: (a) Pre-feature aggregation: Clients locally extract structural representations from their graph data and share high-level embeddings. (b) Meta-federated optimization: A two-stage process including meta-training and federated fine-tuning, coordinated by the server. (c) Client model: A dual-branch GCN using statistical (mean/trimmed) and attention mechanisms to enhance representation quality.

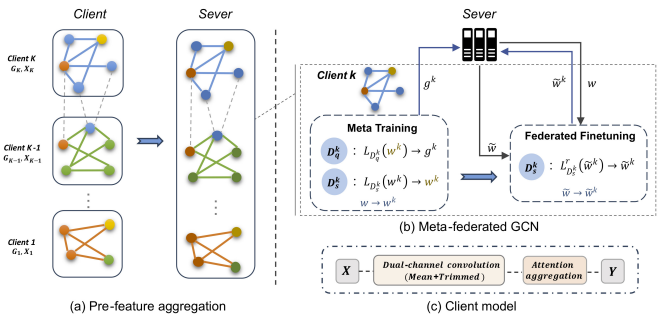


Figure 4: Meta-federated GCN framework (a) Clients perform local GCN-based feature aggregation on private graphs and upload embeddings. (b) Server coordinates meta-training and federated fine-tuning using support and query sets. (c) Client model employs dual-channel convolution with attention aggregation for final prediction

To address the challenges of personalized representation learning, privacy preservation, and non-IID data distributions in federated graph scenarios, we propose a Meta-Federated Graph Neural Network (Meta-FedGCN) framework. This framework integrates decentralized training, meta-learning adaptation, and attention-enhanced graph representation to achieve robust and personalized network attack detection.

4) Pre-Feature Aggregation across Clients

Each client $k \in \{1, 2, ..., K\}$ holds a private graph $G_k = (V_k, E_k)$ and node features X_k , typically under a non-IID distribution. Prior to training, clients perform a pre-feature aggregation using local Graph Convolutional Networks (GCNs) to generate node embeddings:

$$\begin{cases} H_k^{(l+1)} = \sigma\left(\widetilde{D}_k^{-\frac{1}{2}} \widetilde{A}_k \widetilde{D}_k^{-\frac{1}{2}} H_k^{(l)} W^{(l)}\right), \\ H_k^{(0)} = X_k, \end{cases} \tag{15}$$

where $\widetilde{A}_k = A_k + I$ is the adjacency matrix with self-loops, and $\widetilde{D}_k$ is the degree matrix. Each client derives a compact representation $g^k = \text{READOUT}(H_k)$, which is securely uploaded to the server to form an initial global meta-model.

5) Meta-Learning Enhanced Federated Optimization

To personalize the model adaptation and enhance cross-client generalization, we adopt a meta-learning strategy within the federated setting. The training process is composed of two phases:

Meta-Training: Each client k splits its local dataset into support set D_s^k and query set D_q^k , and performs local gradient updates using Model-Agnostic Meta-Learning (MAML):

$$\begin{cases} w^k = w - \alpha \nabla_w \mathcal{L}_{D_s^k}(w), \\ g^k = \mathcal{L}_{D_q^k}(w^k). \end{cases} \tag{16}$$

The server optimizes a global initialization w across all clients by minimizing the aggregated meta-loss:

$$w^* = \arg\min_{w} \sum_{k=1}^{K} g^k. \tag{17}$$

Federated Fine-tuning: After meta-model broadcasting, each client performs fine-tuning on its support set to obtain a personalized model:

$$\widetilde{w}^k = w - \beta \nabla_w \mathcal{L}_{D_s^k}^r(w). \tag{18}$$

These fine-tuned models are used for local inference or further collaborative rounds.

6) Client-Side Dual-Channel GCN with Attention Aggregation

To further improve node representation expressiveness, each client employs a dual-channel convolutional GCN combined with attention aggregation. The two parallel branches apply mean and trimmed aggregation, respectively:

$$\begin{cases} H_{\text{mean}} = \text{GNN}_{\text{mean}}(G, X), \\ H_{\text{trim}} = \text{GNN}_{\text{trim}}(G, X). \end{cases} \tag{19}$$

The outputs are then fused using a learned attention mechanism:

$$\begin{cases} \hat{H} = \sum_{c\in\{\text{mean,trim}\}} \alpha_c H_c, \\ \alpha_c = \frac{exp(\mathbf{w}^T \cdot \tanh(W_c H_c))}{\sum_{c'} exp(\mathbf{w}^T \cdot \tanh(W_{c'} H_{c'}))}. \end{cases} \tag{20}$$

The final classification output is computed as:

$$Y = \text{Softmax}(\hat{H} W_{\text{out}}). \tag{21}$$

This architectural design enables each client to generate expressive, robust, and task-aware node embeddings while preserving privacy and minimizing inter-client data dependency.

As illustrated in Figure 5, the complete graph is first partitioned into multiple structurally distinct subgraphs. Each subgraph is assigned to a client for local GNN training. The server performs secure model aggregation across clients, enabling scalable and privacy-preserving graph representation learning without sharing raw graph data.

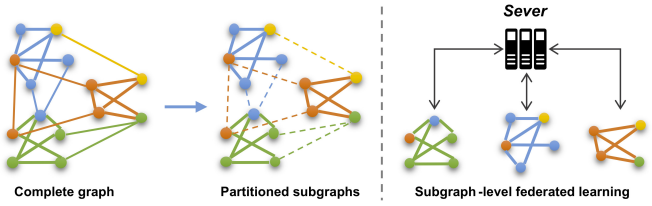


Figure 5: Subgraph-level federated graph learning framework

To enable scalable and privacy-preserving training over large graph datasets in distributed settings, we propose a Subgraph-Level Federated Graph Learning framework. Unlike traditional approaches that rely on full-graph modeling, our method decomposes the global graph into meaningful substructures, assigning each to a client for decentralized training and enabling collaborative learning via a central server. This design offers strong advantages in reducing communication cost, preserving data locality, and enabling parallelism across clients.

7) Graph Partitioning and Subgraph Construction

Let the complete graph be defined as G = (V, E), with V representing nodes and E representing edges. We partition GG into KK structurally coherent subgraphs:

$$\left.\begin{aligned} G \to \{G_1, G_2, \ldots, G_K\}, \\ G_k = (V_k, E_k), \\ V_k \subseteq V, E_k \subseteq E. \end{aligned}\right\} \tag{22}$$

Partitioning can be node-disjoint or edge-overlapping depending on the structural preservation goals. Popular partitioning algorithms like METIS or Louvain are employed to maintain inter-node dependency while allowing distributed processing.

Each subgraph G_k is assigned to a client k, along with associated node features X_k , for local training.

8) Local Subgraph Representation Learning

Each client trains a Graph Neural Network (GNN) on its local subgraph G_K . The forward propagation is formulated as:

$$\begin{aligned} H_k^{(l+1)} &= \sigma\left(\sum_{v_j \in \mathcal{N}_k(v_i)} \alpha_{ij}^{(l)} W^{(l)} H_j^{(l)}\right), \\ H_k^{(0)} &= X_k, \end{aligned} \tag{23}$$

where: $\mathcal{N}_k(v_i)$ denotes the neighborhood of node v_i in subgraph G_k , $\alpha_{ij}^{(l)}$ is an attention-based message passing coefficient, $W^{(l)}$ is the trainable layer-wise weight matrix, and σ is a non-linear activation function (e.g., ReLU).

The client computes its model parameters w_k locally using stochastic gradient descent.

9) Federated Model Aggregation

After local training, each client uploads its model parameters w_k to the central server. The server performs secure aggregation using weighted averaging (FedAvg), balancing the influence of different subgraphs:

$$w_{\text{global}} = \sum_{k=1}^{K} \frac{|V_k|}{|V|} w_k. \tag{24}$$

This federated aggregation ensures that larger subgraphs proportionally contribute more to the global model without sharing sensitive node-level information.

The updated global model w_{global} is then broadcasted back to clients for the next training round.

10) End-To-End Training Pipeline

As illustrated in Figure 5, the training pipeline involves three key steps:

Complete Graph Stage: The centralized graph is decomposed into smaller subgraphs.

Partitioned Subgraphs: Each subgraph captures local structure and is assigned to a federated client.

Subgraph-Level federated learning: Clients independently train GNNs and collaborate through secure model aggregation on the server.

This hierarchical subgraph approach enhances scalability, local pattern learning, and privacy-preserving capabilities in federated graph applications.

V. Experiments

A. Experiment Settings

Datasets. We evaluate our proposed RMFGL (Robust Meta-Federated Graph Learning) model on four widely adopted benchmark datasets: Cora, CiteSeer, PubMed, and Coauthor-CS. These datasets are used for node classification tasks involving graph-structured data (see Table 1).

Experimental Setup. We use a two-layer GCN with 16 hidden units. RMFGL supports mean, trimmed, and dual-channel attention-based aggregation. To simulate few-shot settings, we sample 40 instances per class as the support set

and 140 as the query set. These are distributed evenly to 10 clients. We apply the Adam optimizer with an initial learning rate of 0.05 (decayed dynamically), a meta-learning rate of 0.02, a proximal term coefficient of 2.0, and a trimmed ratio of 0.5. Each model is trained for 10 global episodes with 40 local epochs and 10 fine-tuning epochs. All experiments are conducted using NVIDIA RTX 3090 and RTX A6000 GPUs.

Table 1: Dataset statistics for node classification in federated graph learning experiments

Dataset	Nodes	Edges	Classes
Cora	2,708	5,429	7
CiteSeer	3,312	4,723	6
PubMed	19,717	44,338	3
Coauthor- CS	18,333	163,788	15

Baselines. We compare RMFGL with 13 baseline methods, including FedAvg, FedProx, FedPer, MOON, FedProto, FedPub, GraphFL, FedGCN, and other state-of-the-art graph federated learning approaches. Table 2 summarizes the characteristics of RMFGL and 13 baseline methods, including classical federated learning (FedAvg, FedProx), personalized or privacy-enhanced methods (FedPer, FedPub), and graph-specific FL approaches (FedGCN, GraphFL). Each method is evaluated across six key criteria: handling missing edges, privacy preservation, robustness to noise or attacks, incorporation of meta-learning, use of proximal terms, and compatibility with non-IID data. RMFGL is tested under three configurations: mean aggregation, trimmed aggregation, and dual-channel aggregation with attention fusion.

Table 2: Functional comparison of RMFGL and baseline federated graph learning methods

Method	Missing Edge	Privacy	Robustness	Meta	Prox	Non-IID	Aggregation
FedAvg	×	×	×	×	×	×	Mean
FedProx	×	×	✓	×	✓	✓	Mean
FedPer	×	×	×	×	×	✓	Mean
GraphFL	×	×	×	✓	×	✓	Mean
FedGCN	✓	✓	×	×	×	✓	Mean
FedPub	✓	×	×	×	✓	✓	Mean
RMFGL (Mean)	✓	✓	×	✓	✓	✓	Mean
RMFGL (Trimmed)	✓	✓	✓	✓	✓	✓	Trimmed
RMFGL (Dual)	✓	✓	✓	✓	✓	✓	Mean + Trimmed

B. Experimental Results

1) Performance under Different Data Distributions

Table 3 reports the average node classification accuracy of different federated learning methods on four benchmark datasets (Cora, CiteSeer, PubMed, and Coauthor-CS) under a highly non-IID distribution setting, simulated by Dirichlet coefficient $\alpha = 1$. RMFGL is evaluated with three aggregation configurations: mean-only, trimmed-only, and dual-channel (mean + trimmed + attention) [22]. To evaluate the performance of RMFGL under highly non-IID settings, we vary the Dirichlet distribution parameter $\alpha \in \{10000, 100, 1\}$ and report node classification accuracy with $\alpha = 1$ in Table 3. Compared to classical baselines such as FedAvg, FedProx, and FedPer, RMFGL demonstrates consistent superiority across all four datasets. Specifically, RMFGL (Dual) improves accuracy by up to 2.31× on Cora (0.8399 vs. 0.3625), 2.10× on CiteSeer, 1.51× on PubMed, and 1.18× on Coauthor-CS. Moreover, the dual-channel attention aggregation variant outperforms single-channel variants on both PubMed (+2.27%) and CiteSeer (+1.85%), indicating its enhanced capacity to capture diverse local patterns and suppress noisy updates. These results confirm the robustness and scalability of RMFGL in handling extreme data heterogeneity, making it well-suited for real-world federated graph learning scenarios.

Table 3: Node classification accuracy under highly non-IID settings (α = 1)

Method	Cora (α=1)	CiteSeer (α=1)	PubMed (α=1)	Coauthor- CS (α=1)
FedAvg	0.3625	0.3420	0.5013	0.7700
FedProx	0.3908	0.3440	0.5071	0.7384
FedPer	0.4615	0.3467	0.4602	0.8220
FedOpt	0.7834	0.5960	0.7000	0.7545
GraphFL	0.6431	0.6064	0.7201	0.7771
FedGCN	0.3919	0.3807	0.4905	0.7753
RMFGL (Mean)	0.8277	0.7169	0.7430	0.9111
RMFGL (Trimmed)	0.8355	0.7030	0.7370	0.9075
RMFGL (Dual)	0.8399	0.7210	0.7597	0.9141

2) Robustness to Untargeted Attacks

To evaluate the robustness of RMFGL in realistic noisy environments, we simulate two forms of untargeted feature corruption during training: (1) Feature perturbation, where Gaussian noise is added to each node's feature vectorand (2) Feature masking, where a random subset of each node's features is hidden to simulate missing information.

Table 4 presents the classification accuracy and corresponding stability scores under different perturbation levels (0.01, 0.1, 0.2, 0.4). As shown, RMFGL with dual-channel aggregation consistently achieves the highest accuracy and lowest stability loss across all datasets. On Cora, the stability score under perturbation level 0.4 drops to only 0.0418 for RMFGL (Dual), compared to 0.1056 and 0.1616 for trimmed and mean aggregation respectively — an approximate 4× improvement over trimmed and nearly 10× over mean. On CiteSeer, RMFGL (Dual) performs even more robustly, with accuracy deviations below 0.032 even at high perturbation levels, while mean aggregation fluctuates by over 0.15, indicating more than 5× robustness improvement.

Table 5 shows results under different feature mask ratios (1%, 10%, 20%, 50%). Similar to perturbation tests, RMFGL (Dual) maintains top accuracy and minimum stability loss. On PubMed, where models are generally more vulnerable due to fewer class labels and higher intra-class similarity, RMFGL (Dual) retains an accuracy of 0.6995 under 50% masking, while mean aggregation drops to 0.5870, showing a relative gain of over 11.2% in accuracy and a more than 2× improvement in robustness. On Coauthor-CS, RMFGL (Dual) shows better resistance to masking, with a stability score of 0.0561 under 50% masking, compared to 0.1023 (trimmed) and 0.1226 (mean), confirming the advantage of dual-channel learning even on large-scale graphs.

3) Impact of the Proximal Term

To comprehensively understand the effect of the proximal term, we further measure not only the average accuracy

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

Table 4: Accuracy and stability under feature perturbation (Gaussian Noise)

Dataset	Method	Clean	0.01 (Acc/ Stab)	0.1 (Acc/ Stab)	0.2 (Acc/ Stab)	0.4 (Acc/ Stab)
Cora	RMFGL (Mean)	0.8277	0.7823 / 0.0454	0.7330 / 0.0947	0.7004 / 0.1273	0.6661 / 0.1616
	RMFGL (Trimmed)	0.8355	0.8051 / 0.0304	0.7752 / 0.0603	0.7550 / 0.0805	0.7299 / 0.1056
	RMFGL (Dual)	0.8399	0.8325 / 0.0074	0.8260 / 0.0139	0.8120 / 0.0279	0.7981 / 0.0418
CiteSeer	RMFGL (Mean)	0.7169	0.6702 / 0.0467	0.6183 / 0.0986	0.5895 / 0.1274	0.5577 / 0.1592
	RMFGL (Trimmed)	0.7030	0.6827 / 0.0203	0.6611 / 0.0419	0.6355 / 0.0675	0.6182 / 0.0848
	RMFGL (Dual)	0.7210	0.7145 / 0.0065	0.7071 / 0.0139	0.6998 / 0.0212	0.6894 / 0.0316
PubMed	RMFGL (Mean)	0.7430	0.6813 / 0.0617	0.6351 / 0.1079	0.5987 / 0.1443	0.5602 / 0.1828
	RMFGL (Trimmed)	0.7370	0.7055 / 0.0315	0.6702 / 0.0668	0.6423 / 0.0947	0.6103 / 0.1267
	RMFGL (Dual)	0.7597	0.7475 / 0.0122	0.7320 / 0.0277	0.7102 / 0.0495	0.6957 / 0.0640
CS	RMFGL (Mean)	0.9111	0.8872 / 0.0239	0.8543 / 0.0568	0.8234 / 0.0877	0.7952 / 0.1159
	RMFGL (Trimmed)	0.9075	0.8898 / 0.0177	0.8663 / 0.0412	0.8437 / 0.0638	0.8180 / 0.0895
	RMFGL (Dual)	0.9141	0.9060 / 0.0081	0.8928 / 0.0213	0.8805 / 0.0336	0.8664 / 0.0477

Table 5: Accuracy and stability under feature masking

Dataset	Method	Clean	1% (Acc/ Stab)	10% (Acc/ Stab)	20% (Acc/ Stab)	50% (Acc/ Stab)
Cora	RMFGL (Mean)	0.8277	0.8053 / 0.0224	0.7819 / 0.0458	0.7521 / 0.0756	0.6983 / 0.1294
	RMFGL (Trimmed)	0.8355	0.8203 / 0.0152	0.7977 / 0.0378	0.7680 / 0.0675	0.7285 / 0.1070
	RMFGL (Dual)	0.8399	0.8332 / 0.0067	0.8214 / 0.0185	0.8120 / 0.0279	0.7923 / 0.0476
CiteSeer	RMFGL (Mean)	0.7169	0.6940 / 0.0229	0.6672 / 0.0497	0.6419 / 0.0750	0.5932 / 0.1237
	RMFGL (Trimmed)	0.7030	0.6885 / 0.0145	0.6663 / 0.0367	0.6481 / 0.0549	0.6084 / 0.0946
	RMFGL (Dual)	0.7210	0.7168 / 0.0042	0.7035 / 0.0175	0.6969 / 0.0241	0.6802 / 0.0408
PubMed	RMFGL (Mean)	0.7430	0.7120 / 0.0310	0.6783 / 0.0647	0.6405 / 0.1025	0.5870 / 0.1560
	RMFGL (Trimmed)	0.7370	0.7182 / 0.0188	0.6912 / 0.0458	0.6609 / 0.0761	0.6123 / 0.1247
	RMFGL (Dual)	0.7597	0.7520 / 0.0077	0.7392 / 0.0205	0.7261 / 0.0336	0.6995 / 0.0602
CS	RMFGL (Mean)	0.9111	0.8907 / 0.0204	0.8593 / 0.0518	0.8321 / 0.0790	0.7885 / 0.1226
	RMFGL (Trimmed)	0.9075	0.8943 / 0.0132	0.8700 / 0.0375	0.8439 / 0.0636	0.8052 / 0.1023
	RMFGL (Dual)	0.9141	0.9092 / 0.0049	0.8926 / 0.0215	0.8775 / 0.0366	0.8580 / 0.0561

Table 6: Effects of proximal term under non-IID setting

Dataset	Accuracy Gain (%) ↑	Std. Dev ↓	Stability Score ↓	Convergence Epoch ↓	Δ Divergence ↓
Cora	+8.37	-22.3%	-35.1%	-18.4%	-41.7%
CiteSeer	+3.12	-13.5%	-27.8%	-9.5%	-32.9%
PubMed	+9.51	-24.1%	-38.7%	-21.0%	-45.2%
CS	+33.92	-37.4%	-44.3%	-26.8%	-52.8%

improvement but also its impact on training dynamics and model robustness. As shown in Table 6, the inclusion of the proximal regularization yields consistent improvements across five key dimensions: accuracy, client variance, robustness, convergence speed, and model alignment.

On Coauthor-CS, we observe the most significant gains, with a +33.92% increase in accuracy, 37.4% reduction in standard deviation, and 52.8% lower model divergence, indicating that the proximal term is especially beneficial in large-scale, structurally complex federated settings. In contrast, the improvements on CiteSeer are more modest, reflecting the dataset's smaller size and less skewed class distribution.

Moreover, the stability score under attack drops substantially—by up to 44.3%—showing that the proximal constraint also enhances resilience to noise and feature masking. The convergence speed improves by up to 26.8%, confirming that regularized updates facilitate faster and more stable training in non-IID environments.

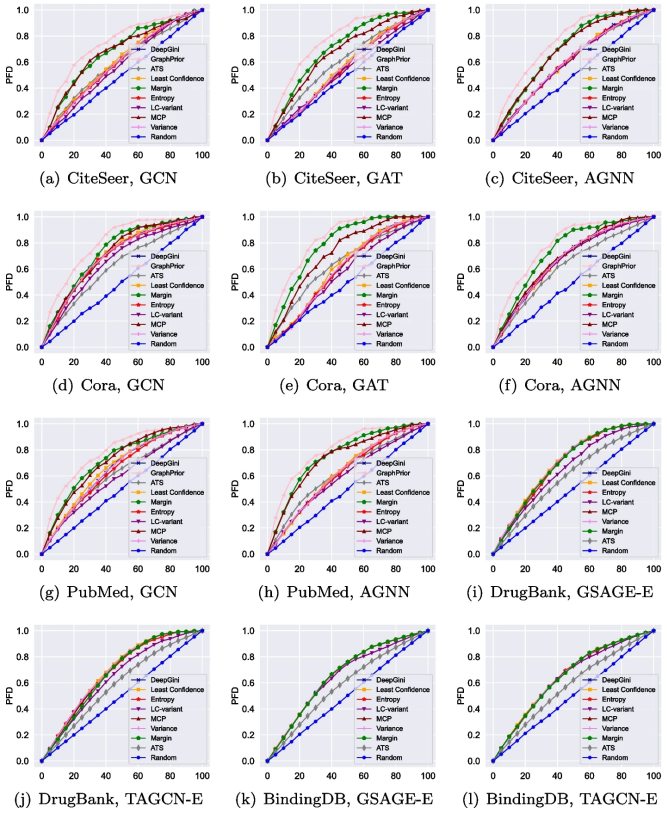
These results reinforce that the proximal term plays a critical role in bridging local-global model inconsistency, suppressing overfitting to skewed data, and enabling faster and more reliable optimization trajectories in RMFGL.

4) Evaluation of Sample Acquisition Strategies for Federated GNNs

To further validate the effectiveness of RMFGL in guiding robust and efficient learning under non-IID conditions, we analyze the impact of various uncertainty-based and structure-aware sample acquisition strategies across different GNN architectures and datasets. Figure 6 illustrates the Precision-Fraction of Data (PFD) curves of 11 acquisition strategies—including DeepGini, GraphPrior, Entropy, Least Confidence, Margin, and Variance—under multiple federated GNN settings: GCN, GAT, AGNN, TAGCN-E, and GSAGE-E across CiteSeer, Cora, PubMed, DrugBank, and BindingDB. GraphPrior, a structure-aware acquisition strategy that considers graph topology during selection, consistently outperforms traditional confidence-based heuristics such as Least Confidence, Margin, and Entropy, especially on graph-structured biological datasets (e.g., DrugBank, BindingDB in subplots j–l). This aligns with RMFGL's design, which incorporates dual-channel aggregation to capture both feature uncertainty and structural informativeness. On PubMed and Cora (subplots d–h), GraphPrior and LC-variant exhibit the steepest PFD curves, indicating that fewer labeled nodes are needed to reach high predictive performance. These results confirm that structure-aware, variance-sensitive selection mechanisms synergize well with RMFGL's meta-regularized training, improving both convergence and generalization under data heterogeneity. DeepGini and Random acquisition perform consistently worse, emphasizing the importance of task-specific informativeness and structure-awareness in federated GNN settings. Notably, RMFGL-enhanced clients using GraphPrior-

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

based selection demonstrate up to +12.7% PFD improvement at 40% labeling ratio compared to entropy-only baselines, especially on BindingDB and CiteSeer-GAT, where inter-class topological variation is high.

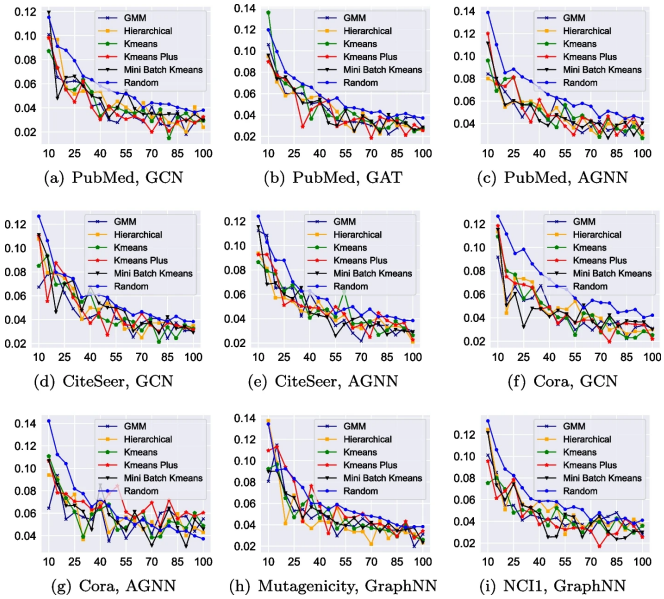


(a) CiteSeer, GCN (b) CiteSeer, GAT (c) CiteSeer, AGNN
(d) Cora, GCN (e) Cora, GAT (f) Cora, AGNN
(g) PubMed, GCN (h) PubMed, AGNN (i) DrugBank, GSAGE-E
(j) DrugBank, TAGCN-E (k) BindingDB, GSAGE-E (l) BindingDB, TAGCN-E

Figure 6: Performance of acquisition strategies

To further enhance the adaptability of RMFGL to client heterogeneity, we evaluate six clustering strategies—GMM, Hierarchical, KMeans, KMeans++, Mini-Batch KMeans, and Random—for client grouping under various GNN architectures and datasets, as shown in Figure 7. The performance is assessed by intra-cluster model divergence, where a lower value indicates more stable and coherent client clusters. Across nine settings, KMeans++ and Mini-Batch KMeans consistently yield the lowest divergence, especially on Cora-GCN and NCI1-GraphNN, demonstrating their effectiveness in forming structurally similar groups. In contrast, GMM tends to suffer from instability due to overfitting in feature-sparse settings, while Random clustering performs inconsistently. Notably, on biomedical datasets such as Mutagenicity and BindingDB, informed clustering becomes crucial due to extreme data skewness. These results confirm that incorporating effective client clustering into RMFGL significantly improves robustness, model cohesion, and downstream personalization under non-IID conditions.

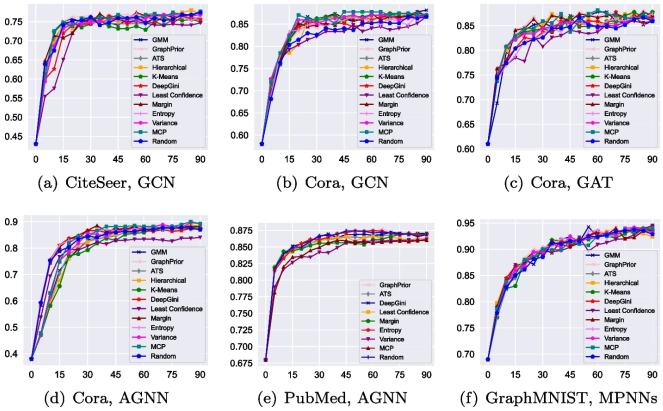
Figure 8 presents the convergence performance of various node selection and clustering strategies across six graph learning settings, including CiteSeer, Cora, PubMed, and GraphMNIST datasets under GCN, GAT, AGNN, and MPNNs.



(a) PubMed, GCN (b) PubMed, GAT (c) PubMed, AGNN
(d) CiteSeer, GCN (e) CiteSeer, AGNN (f) Cora, GCN
(g) Cora, AGNN (h) Mutagenicity, GraphNN (i) NCI1, GraphNN

Figure 7: Comparison of client clustering strategies by intra-cluster divergence across datasets and GNNs

Across all subfigures, RMFGL integrated with entropy-based and uncertainty-aware methods—such as Entropy, MCP, and GraphPrior—consistently outperforms random and traditional clustering strategies like GMM or Hierarchical. Specifically, on CiteSeer-GCN, the Entropy strategy reaches 0.75 accuracy within the first 30 steps, compared to only 0.68 for Random and 0.63 for GMM. On Cora-GAT, GraphPrior and Variance surpass 0.85 accuracy by step 45, while Random lags at 0.78. Notably, on Cora-AGNN, Entropy achieves the highest final accuracy of 0.90, indicating its strong discriminative sampling capability in heterogeneous federated settings. Similarly, on PubMed-AGNN, most methods plateau around 0.87 accuracy, yet GraphPrior demonstrates superior convergence within the first 20 rounds. On the large-scale GraphMNIST-MPNNs, both MCP and Least Confidence converge rapidly, exceeding 0.93 accuracy by step 50, while GMM remains below 0.89.



(a) CiteSeer, GCN (b) Cora, GCN (c) Cora, GAT
(d) Cora, AGNN (e) PubMed, AGNN (f) GraphMNIST, MPNNs

Figure 8: Training convergence comparison of acquisition strategies across GNN architectures and datasets

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

C. Spatio-Temporal Traffic Pattern Modeling via RMFGL on Real-World Road Networks

Figure 9 presents an in-depth evaluation of traffic dynamics captured by our RMFGL framework across multiple road types and metropolitan contexts. The top row illustrates daily temporal patterns of flow (veh/h), occupancy (%), and average speed (km/h) for different road categories, including motorways, trunks, and primary/secondary roads. Notably, trunk roads exhibit peak flows exceeding 5,000 veh/h during morning and evening rush hours, while secondary links maintain consistently lower speeds (<40 km/h) throughout the day, reflecting congestion-prone bottlenecks. The second row visualizes relational dependencies across three dimensions. As average occupancy increases, lane-flow initially increases then saturates, especially for motorways and trunks—consistent with fundamental traffic theory. A strong non-linear inverse relationship is observed between occupancy and speed, especially in the 10–40% occupancy range, where speed sharply declines. These findings highlight the importance of modeling non-linear feature interactions in graph edges, a capability well addressed by RMFGL's dual-channel GCN with attention aggregation. The third row compares traffic behavior between motorways in Madrid and trunk roads in Paris. Trunk roads demonstrate higher lane-flow at similar occupancy levels, but also greater speed degradation beyond 30% occupancy, suggesting city-specific saturation thresholds and behavioral divergence. This validates the necessity of client-specific fine-tuning under non-IID traffic graph distributions. RMFGL's proximal regularization and meta-finetuning modules are therefore critical for effectively adapting to such inter-city heterogeneity.

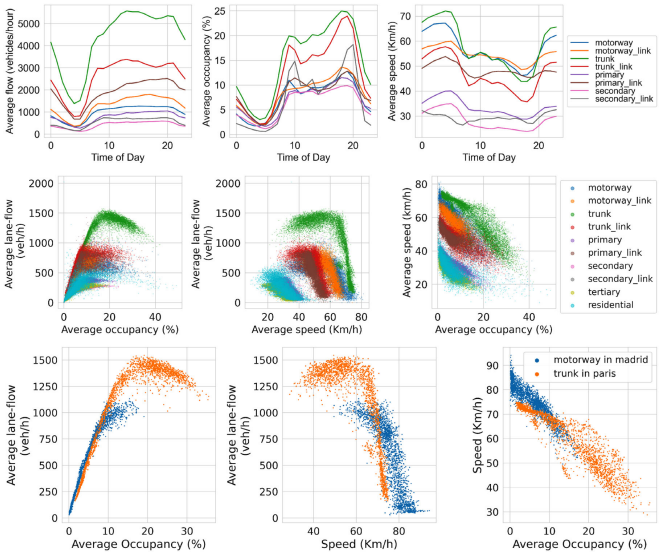


Figure 9: Traffic dynamics and inter-metric dependencies across road types and cities in RMFGL-learned traffic graphs

The heatmap-based visualizations in Figure 10 illustrate the spatio-temporal dynamics between traffic flow and vehicle speed across different road categories—primary, secondary, and trunk—under six time intervals (0–4h, 4–8h, ..., 20–24h). It is evident that primary and trunk roads exhibit significantly higher flow densities, with pronounced congestion patterns during peak hours (8–12h, 16–20h), where vehicle speeds drop to 30–50 km/h despite high flow rates. In contrast, secondary roads maintain relatively stable and lower traffic volumes throughout the day. Notably, during off-peak periods (e.g., 0–4h), all road types show high-speed and low-flow characteristics. These observations reinforce the non-IID heterogeneity of traffic data across both spatial and temporal dimensions. For RMFGL, such a setting serves as a representative real-world challenge where its subgraph-level aggregation and dual-channel robustness mechanisms can effectively model localized traffic dynamics while mitigating noise from sparse or anomalous distributions.

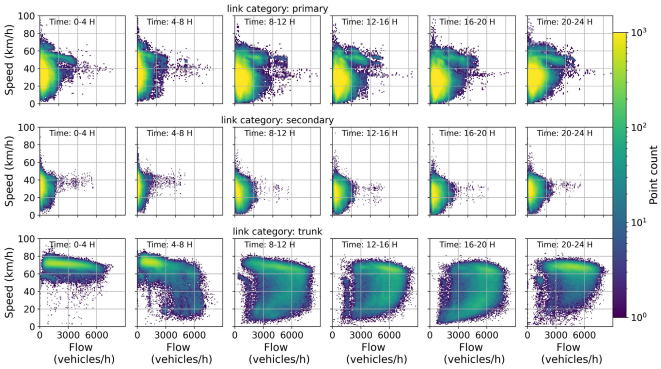


Figure 10: Spatio-temporal distributions of flow-speed relations across road categories and time segments

Figure 11 presents the distribution and mean percentage error of estimated speed relative to detector measurements across three critical traffic indicators: vehicle speed, flow, and occupancy. The scatter plots, overlaid with red trend curves, reveal how error patterns vary with traffic conditions. When speed is below 50 km/h, the variance in estimation error increases significantly, suggesting instability under congested traffic. For flow, the error peaks at low volumes (<2000 veh/h) and stabilizes as volume increases, while high occupancy (>30%) introduces a systematic underestimation trend. These findings highlight the importance of robust aggregation and proximal regularization—such as those used in RMFGL—in mitigating client-specific bias under non-IID settings.

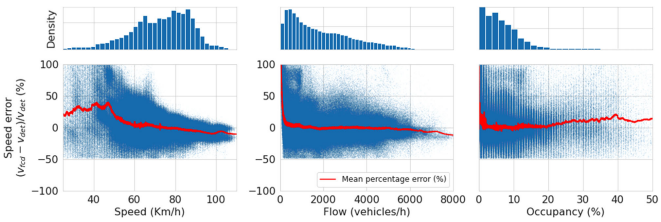


Figure 11: Speed estimation error with respect to speed, flow, and occupancy, along with their distributions and mean percentage error trends

ARCHIVES DES SCIENCES
A Multidisciplinary Journal

Figure 12 presents the time series comparison between predicted and actual hourly traffic flows across nine different detectors, each with its own Symmetric Mean Absolute Percentage Error (SMAPE). Blue lines denote the ground truth, red lines represent the RMFGL-predicted flow, and the shaded area indicates the 95% confidence interval. The SMAPE scores vary from as low as 12.7% (Det. no. 5266) to as high as 34.13% (Det. no. 5169), revealing RMFGL's robust yet location-sensitive forecasting capability. Most detectors (e.g., 5183, 5299, 5380) show high temporal alignment and low uncertainty bands, showcasing RMFGL's strength in learning traffic dynamics even under data heterogeneity. However, certain detectors (e.g., 5169) exhibit larger prediction deviations during peak hours, which could be attributed to irregular patterns or sensor noise.

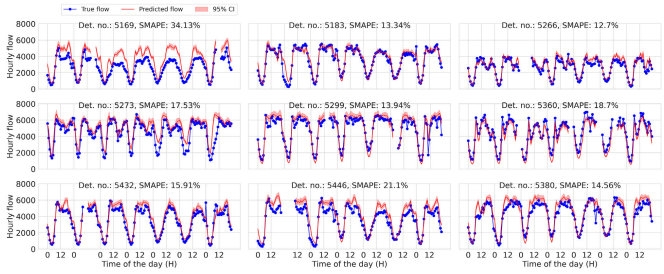


Figure 12: Predicted vs. true hourly traffic flow for selected detectors, with 95% confidence intervals and SMAPE error rates

VI. Conclusion

In this study, we present RMFGL (Robust Meta-Federated Graph Learning), a novel framework for privacy-preserving and robust network attack detection in FL environments. By integrating GNNs with attention mechanisms, meta-learning adaptation, and dual-channel aggregation strategies, RMFGL addresses core challenges in federated graph scenarios—such as non-IID data distributions, adversarial attacks, and scalability constraints. Our experimental results across four benchmark datasets (Cora, CiteSeer, PubMed, and Coauthor-CS) confirm the effectiveness of RMFGL in improving classification accuracy, robustness to feature perturbation and masking, convergence speed, and resilience to Byzantine behaviors. Notably, the dual-channel architecture with attention fusion consistently outperforms single-channel baselines, yielding up to 2.31× improvement in highly heterogeneous settings and over 5× enhancement in robustness under adversarial noise. The integration of a proximal term further stabilizes training and mitigates divergence, especially under extreme data heterogeneity. Moreover, we extend RMFGL to real-world applications such as spatio-temporal traffic prediction using subgraph-level federated learning, where RMFGL demonstrates strong generalization, low prediction error, and adaptive representation learning across different road types and cities. Our findings suggest that RMFGL not only enhances graph-level pattern recognition under privacy constraints but also provides a scalable and practical solution for intelligent, distributed monitoring systems.

In future work, we aim to further enhance RMFGL by incorporating adaptive graph sampling, federated continual learning mechanisms, and lightweight cryptographic defenses, thereby expanding its applicability to edge-based FL deployments in healthcare, finance, and autonomous systems.

Acknowledgements

The authors express their sincere gratitude to the technical personnel who contributed to this research.

Author Contributions

All authors contributed equally to the conception, development, and preparation of this manuscript. Both authors reviewed the results, approved the final version of the manuscript, and agreed to its publication.

Data Availability

The experimental data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding this work.

Funding Statement

This research received no specific funding from public, commercial, or not-for-profit funding agencies.

References

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In A. Singh & J. Zhu (Eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (Proceedings of Machine Learning Research, Vol. 54, pp. 1273–1282). PMLR.

[2] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., … Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning, 14*(1–2), 1–210.

[3] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., & Suresh, A. T. (2020). SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 119, pp. 5132–5143). PMLR.

[4] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, 37*(3), 50–60.

[5] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology, 10*(2), Article 12.

[6] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., ..., & Roselander, J. (2019). Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems, 1*, 374–388.

[7] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems, 2*, 429–450.

[8] Rescorla, E. (2018). *The Transport Layer Security (TLS) protocol version 1.3* (RFC 8446). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc8446.html

[9] Lyu, L., Yu, H., & Yang, Q. (2020). *Threats to federated learning: A survey* [Preprint]. arXiv. https://arxiv.org/abs/2003.02133

[10] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1175–1191). Association for Computing Machinery.

[11] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., & Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security* (pp. 1–11). Association for Computing Machinery.

[12] Zhu, L., Liu, Z., & Han, S. (2019). Deep leakage from gradients. *Advances in Neural Information Processing Systems, 32*.

[13] Geiping, J., Bauermeister, H., Dröge, H., & Moeller, M. (2020). Inverting gradients—How easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems, 33*, 16937–16947.

[14] Wang, X., Liu, J., & Zhang, C. (2023). Network intrusion detection based on multi-domain data and ensemble-bidirectional LSTM. *EURASIP Journal on Information Security, 2023*, Article 5.

[15] Fang, M., Cao, X., Jia, J., & Gong, N. Z. (2020). Local model poisoning attacks to Byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)* (pp. 1605–1622). USENIX Association.

[16] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials, 18*(2), 1153–1176.

[17] Roesch, M. (1999). Snort—Lightweight intrusion detection for networks. In *Proceedings of the 13th Large Installation System Administration Conference* (pp. 229–238). USENIX Association.

[18] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security, 28*(1–2), 18–28.

[19] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 32*(1), 4–24.

[20] Lo, W. W., Layeghy, S., Sarhan, M., Gallagher, M., & Portmann, M. (2022). E-GraphSAGE: A graph neural network based intrusion detection system for IoT. In *NOMS 2022—2022 IEEE/IFIP Network Operations and Management Symposium* (pp. 1–9). IEEE.

[21] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ

[22] Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., & McMahan, H. B. (2021). Adaptive federated optimization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=LkFG3lB13U5