Federated Continual Graph Learning

Yinlin Zhu
Sun Yat-sen University
Guangzhou, China
zhuylin27@mail2.sysu.edu.cn

Miao Hu
Sun Yat-sen University
Guangzhou, China
humiao5@mail.sysu.edu.cn

Di Wu*
Sun Yat-sen University
Guangzhou, China
wudi27@mail.sysu.edu.cn

Abstract

Managing evolving graph data presents substantial challenges in storage and privacy, and training graph neural networks (GNNs) on such data often leads to catastrophic forgetting, impairing performance on earlier tasks. Despite existing continual graph learning (CGL) methods mitigating this to some extent, they rely on centralized architectures and ignore the potential of distributed graph databases to leverage collective intelligence. To this end, we propose Federated Continual Graph Learning (FCGL) to adapt GNNs across multiple evolving graphs under storage and privacy constraints. Our empirical study highlights two core challenges: local graph forgetting (LGF), where clients lose prior knowledge when adapting to new tasks, and global expertise conflict (GEC), where the global GNN exhibits sub-optimal performance in both adapting to new tasks and retaining old ones, arising from inconsistent client expertise during server-side parameter aggregation. To address these, we introduce POWER, a framework that preserves experience nodes with maximum local-global coverage locally to mitigate LGF, and leverages pseudo-prototype reconstruction with trajectory-aware knowledge transfer to resolve GEC. Experiments on various graph datasets demonstrate POWER's superiority over federated adaptations of CGL baselines and vision-centric federated continual learning approaches.

CCS Concepts

• **Computing methodologies → Distributed artificial intelligence**; **Machine learning**.

Keywords

Federated Continual Graph Learning; Graph Neural Networks

ACM Reference Format:
Yinlin Zhu, Miao Hu, and Di Wu. 2025. Federated Continual Graph Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3736956

KDD Availability Link:
The source code and further technical report of this paper have been made publicly available at https://doi.org/10.5281/zenodo.15487654.

*Corresponding author

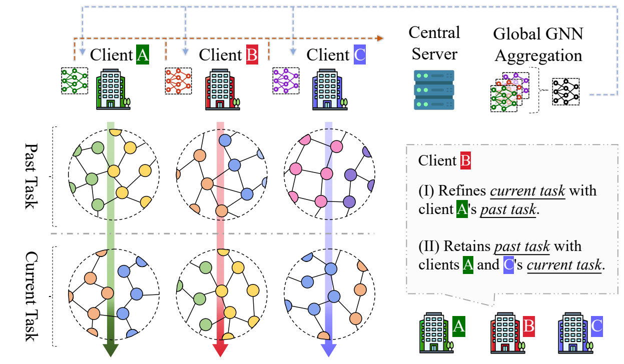

Client A
Client B
Client C
Central Server
Global GNN Aggregation
Past Task
Current Task
Client B
(I) Refines current task with client A's past task.
(II) Retains past task with clients A and C's current task.
A
B
C

Figure 1: Illustration of our FCGL paradigm, which achieves collaborative CGL through multi-round GNN parameter communications between clients and server. Each client's evolving graph is represented by two tasks, with node colors indicating different categories.

1 Introduction

Graph neural networks (GNNs) have emerged as a powerful framework for harnessing relationships in graph-structured data, enabling superior performance across diverse graph-based AI applications, including recommendation [40, 42], biomedical [3, 30], and finance [2, 15]. However, in the era of big data, graph structures often evolve as new entities and relationships emerge, while storing or accessing massive historical node profiles (e.g., features and labels) and topologies is impractical considering database capacity and security [29]. As a result, GNNs often experience unacceptable performance degradation on past tasks (i.e., graphs prior to changes) when adapting to current tasks, known as the *catastrophic forgetting* problem, a consequence of a weak stability-plasticity trade-off [24, 32]. To tackle this issue, inspired by the success of continual learning in vision tasks [22, 23, 38], various continual graph learning (CGL) studies have been proposed in recent years, all of which have demonstrated satisfactory performance [6, 20, 35, 53, 56].

Overlooked Potential: CGL with Collective Intelligence

Despite their effectiveness, current CGL methods are based on the assumption of centralized data storage, where a single institution collects and manages the entire evolving graph. However, this assumption does not always hold in real-world applications. In many practical scenarios, graph-structured data is often distributed across multiple decentralized sources, each capturing and responding to local changes in nodes and topology over time independently.

Motivating Scenario. In the e-commerce domain, various online platforms maintain private evolving product networks, where nodes represent products and edges capture association relationships (e.g., co-purchases or shared reviews), both evolving continuously as new products or associations are introduced [13, 34].

In such decentralized settings, existing CGL methods face **inherent limitations**, as each institution trains its GNN independently, relying solely on its own evolving graph. This isolated learning differs from human learning, which benefits from collective intelligence through group knowledge sharing (e.g., social media, seminars) [33]. Obviously, if an institution could leverage knowledge shared by others, it would be better positioned to mitigate catastrophic forgetting and adapt to new tasks. However, constraints like data privacy and commercial competition make directly aggregating multiple evolving graphs unfeasible, presenting significant challenges to achieving effective CGL with collective intelligence.

First Exploration: Federated Continual Graph Learning

Building on the success of federated graph learning (FGL) [52], we introduce federated continual graph learning (FCGL) as the first practical framework for collaborative CGL in decentralized environments. As shown in Fig. 1, FCGL employs a multi-round communication process. The clients first train local GNNs on their private evolving graphs, and a central server then aggregates these models to form a global GNN, which has improved performance across diverse local graphs. The global model is subsequently broadcast to the clients, enabling the leveraging of collective intelligence without directly sharing massive and sensitive evolving graphs. Notably, although various federated continual learning methods [8, 36, 46, 50] have demonstrated effectiveness in vision tasks, their algorithms rely on augmentation strategies and network architectures tailored specifically for image data, rendering them inapplicable to graph data. Furthermore, for FCGL, the characteristics of multi-client evolving graphs, feasibility, and potential challenges impacting the GNN performance remain unexplored.

Our Efforts: Establishing an Effective FCGL Paradigm

To establish an effective FCGL paradigm, we make these efforts:

E1: In-depth Empirical Investigation of FCGL. We conduct in-depth empirical investigations of FCGL (Sec. 3). From the **Data** perspective, we explore the data distribution of multi-client evolving graphs and identify the phenomenon of divergent graph evolution trajectories, which leads each client to develop expertise in specific classes at different stages, forming the foundation of their collective intelligence. From the **Feasibility** perspective, we observe that even the simplest implementation of FCGL outperforms isolated CGL methods in decentralized scenarios, demonstrating its ability to harness collective knowledge from multi-client evolving graphs. From the **Effectiveness** perspective, we identify two non-trivial challenges impacting GNN performance, including (1) *Local Graph Forgetting* (LGF), where the local GNNs forget the knowledge from the previous tasks when adapting to new ones, and (2) *Global Expertise Conflict* (GEC), where the global GNN performs sub-optimally on both past and current tasks due to graph knowledge conflicts, which occur during parameter aggregation across clients with divergent graph evolution trajectories.

E2: Novel and Effective FCGL Framework. Building on the insights from **E1**, we propose the first FCGL framework, named POWER (graPh evOlution Trajectory-aware knoWledge TransfER). Specifically, POWER is designed with two fundamental objectives: (1) *Addressing LGF at local clients*: POWER selects experience nodes from past tasks with maximum local-global coverage and replays them during local training for future tasks; and (2) *Tackling GEC at central server*: POWER introduces a novel pseudo prototype reconstruction mechanism, enabling the server to capture multi-client graph evolution trajectories. Moreover, a trajectory-aware knowledge transfer process is applied to restore the lost knowledge of global GNN during parameter aggregation.

Our Contributions: (1) **Problem Identification.** To the best of our knowledge, this is the first exploration of the FCGL paradigm, offering a practical solution for continual graph learning under decentralized environments. (2) **In-depth Investigation.** (Sec. 3) We conduct in-depth empirical investigations of FCGL from the perspectives of **Data**, **Feasibility** and **Effectiveness**, providing valuable insights for its development. (3) **Novel Framework.** (Sec. 4) We propose POWER, an novel and effective FCGL training framework that tackles two non-trivial challenges of FCGL named LGF and GEC. (4) **State-of-the-art Performance.** (Sec. 5) Experiments on eight datasets demonstrate that POWER consistently outperforms the federated extension of centralized CGL algorithms and vision-based federated continual learning algorithms.

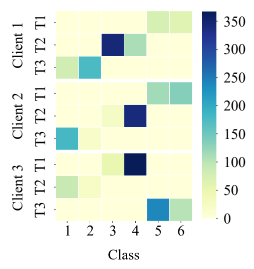
2 Problem Formalization

For FCGL, there is a trusted central server and K clients. Specifically, for the k-th client, we use $\mathcal{T}^k = \{T_1^k, T_2^k, ..., T_M^k\}$ to denotes the set of its tasks, where M is the number of tasks. Then, the local private evolving graph with M tasks can be formalized as follows:

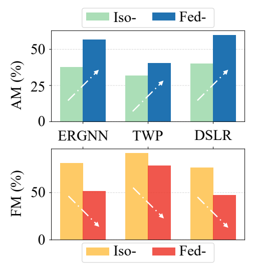
$$\begin{aligned} \mathcal{G}^k &= \{G_1^k, G_2^k, ..., G_M^k\}; \\ G_t^k &= G_{t-1}^k + \Delta G_t^k, \end{aligned} \tag{1}$$

where $G_t^k = (\mathcal{V}_t^k, \mathcal{E}_t^k)$ is the graph given in task T^k, Each node $v_i \in \mathcal{V}_t^k$ has a feature vector $\mathbf{x}_i \in \mathbb{R}^F$ and a one-hot label $\mathbf{y}_i \in \mathbb{R}^C$. The adjacency structure is represented by matrix $\mathbf{A}_t^k \in \mathbb{R}^{N_t \times N_t}$ ($N_t = |\mathcal{V}_t^k|$), and $\Delta G_t^k = (\Delta\mathcal{V}_t^k, \Delta\mathcal{E}_t^k)$ is the change of the node and edge set in task T_t^k. The k-th client aims to train models $\{\text{GNN}_{\Theta_i^k}\}_{i=1}^t$ from streaming data, where Θ_i^k is the GNN parameter in task T_i^k.

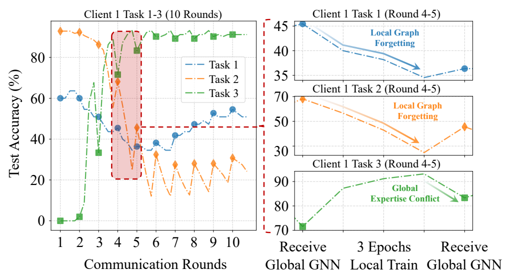
This paper addresses semi-supervised node classification in a class-incremental setting, where new classes emerge as the graph evolves [7]. We assume all clients have M tasks, each introducing c new node categories and undergoing R rounds of FCGL communication. Thus, after task $t = M$, each client encounters $C = c \times M$ categories. We adopt the widely-used FedAvg [25] aggregation strategy from federated learning, adapting it for the FCGL framework as follows: (1) *Initialization.* At the first communication round ($r=1$) of the first task ($t=1$), the central server sets the local GNN parameters of K clients to the global GNN parameters, i.e., $\Theta^k \leftarrow \Theta^g\ \forall k$; (2) *Local Updates.* Each local GNN performs training on the current local data G_t^k to minimize the task loss $\mathcal{L}(G_t^k; \Theta_t^k)$, and then updating the parameters: $\Theta^k \leftarrow \Theta^k - \eta\nabla\mathcal{L}$; (3) *Global Aggregation.* After local training, the server aggregates the local knowledge with respect to the number of training instances, i.e., $\Theta^g \leftarrow \frac{N_k}{N}\sum_{k=1}^K \Theta^k$ with $N = \sum_k N_k$, and distributes the global parameters Θ^g to local clients selected at the next round. For each task t, the process alternates between steps 2 and 3 until reaching the final round R. The iteration continues until the last round of the last task.



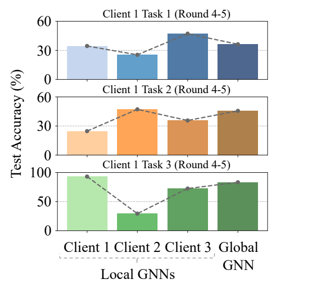
(a) Tasks of Clients



(b) Isolated CGL vs. FCGL



(c) Performance Variation **(Single GNN)**



(d) Performance Comparison **(Multiple GNNs)**

Figure 2: Experimental results of our empirical study. (a) Node label distribution of each client's class-incremental tasks, each exhibiting divergent evolution trajectories. (b) Comparative analysis of three CGL methods in isolated versus federated settings, presenting AM and FM metrics in the upper and lower parts, respectively. (c) Performance variation of Client 1's local GNN (Single GNN) during training on Client 1 Task 3, with markers denoting receiving global parameters from the server. (d) Performance comparison between clients and the server (Multiple GNNs) during training on Client 1 Task 3, indicating that the global GNN can be improved by local GNNs with expertise.

3 Empirical Investigation

In this section, we present a thorough empirical investigation of the proposed FCGL paradigm, structured around three key questions from different perspectives. From the **Data** perspective, to better understand the complexities of graph data in FCGL contexts, we answer **Q1**: What patterns and trends emerge in data distribution across multi-client evolving graphs in FCGL settings? From the **Feasibility** perspective, to demonstrate the advantages of FCGL using collective intelligence, we answer **Q2**: Compared to isolated CGL, does a naive FCGL approach (i.e., CGL integrated with FedAvg) achieves better performance for decentralized evolving graphs? From the **Effectiveness** perspective, to reveal the bottleneck of naive FCGL and realize efficient FCGL paradigm, we answer **Q3**: What non-trivial challenges limit the performance of GNNs within the FCGL framework? Details for the described experimental setup, algorithms, and metrics refer to Sec. 5.1.

To address **Q1**, we simulate multiple decentralized evolving graphs using a two-step approach. First, we partition the Cora dataset [44] into three subgraphs using the Louvain algorithm [4], which is widely utilized in various FGL studies [18, 51, 52]. These subgraphs are distributed among three different clients. Second, following the standard experimental settings of CGL [6, 20, 56], each client's local subgraph is further divided into three class-incremental tasks. Each task comprises two classes of nodes, discarding any surplus classes and removing inter-task edges while preserving only intra-task connections. For each client, we present its tasks with node label distributions in Fig. 2 (a).

Observation 1. Although all these clients eventually learn across six classes, they follow divergent *graph evolution trajectories*, varying both in sample quantity (e.g., Clients 1 and 2) and in the sequence of class learning (e.g., Clients 1 and 3). These clients tend to develop expertise in specific classes at different stages, forming the basis of the collective intelligence harnessed by the FCGL paradigm.

To address **Q2**, we assess the performance of three CGL algorithms, including ERGNN [56], TWP [20], and DSLR [6], under two scenarios: **isolated training** (referred to as Iso-ERGNN, Iso-TWP, and Iso-DSLR) and **federated training** using the FedAvg [25] algorithm (referred to as Fed-ERGNN, Fed-TWP, and Fed-DSLR). In the isolated setting, no client-server communication occurs, while the federated variants implement naive FCGL strategies that simply aggregate parameters from local GNNs. We assessed their performance using two metrics: the accuracy mean (AM) and the forgetting mean (FM), where higher AM values denote better adaptation to new tasks, and lower FM values denote less forgetting of previously learned tasks. The training protocol for each task includes 10 communication rounds, with 3 local epochs per round. As depicted in Fig. 2 (b), the federated variants consistently outperformed their isolated counterparts in both AM and FM metrics. This enhanced performance is attributed to FCGL's capability to leverage diverse knowledge from evolving graph data across clients, thereby reducing task forgetting through effective cross-client knowledge transfer (e.g., alleviating the knowledge forgetting of Client 1 Task 1 and Client 2 Task 1 through insights gained from Client 3 Task 3).

Observation 2. Even a basic implementation of FCGL can substantially enhance the performance of existing CGL algorithms on decentralized evolving graphs.

To address **Q3**, we evaluate the naive FCGL algorithm's performance across Client 1's three tasks during the training of Task 3. For readability, we present the results obtained from Fed-ERGNN, as similar trends are also shown in Fed-TWP and Fed-DSLR. Our analysis is further concentrated on rounds 4 and 5. From the **Single GNN** aspect, we analyze performance variations in Client 1's local GNN, particularly during the critical phases of *global parameter reception* and *local training*, motivated by findings in existing literature [20, 46], which suggest that shifts in parameter values play a significant role in causing knowledge forgetting. This perspective enables a closer examination of how individual GNNs react to external updates. As shown in Fig. 2 (c), each *global parameter reception* phase is marked with scatter points, while lines represent the 3-epoch *local training* phases. From the **Multiple GNNs** aspect, we compare the performance of all local GNNs and the aggregated global GNN. The results are depicted in Fig. 2 (d).

Observation 3. Building on these two aspects, we reveal the following two **Non-trivial Challenges** limit the performance of GNNs within the FCGL framework: (1) *Local Graph Forgetting (LGF)*. From the **Single GNN** aspect, local training on the current task (i.e.,

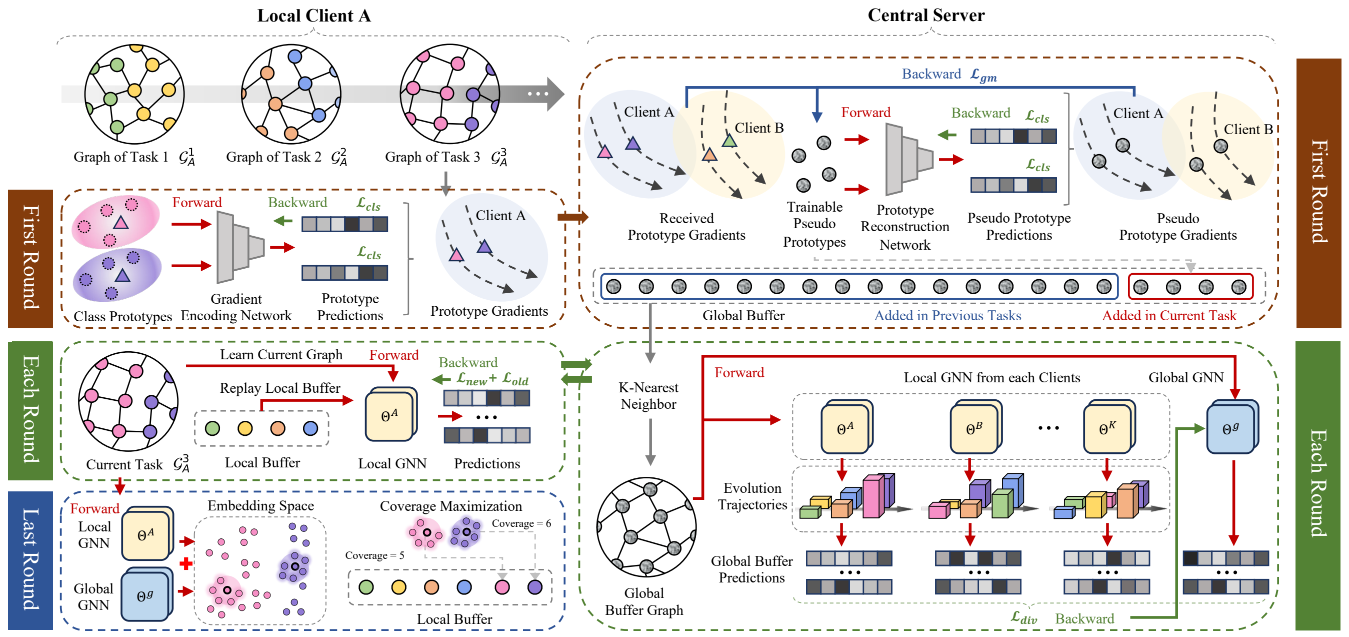

Local Client A
Central Server
Graph of Task 1 $\mathcal{G}_A^1$
Graph of Task 2 $\mathcal{G}_A^2$
Graph of Task 3 $\mathcal{G}_A^3$
First Round
Forward
Backward $\mathcal{L}_{cls}$
Class Prototypes
Gradient Encoding Network
Prototype Predictions
Client A
Prototype Gradients
Backward $\mathcal{L}_{gm}$
Client B
Received Prototype Gradients
Trainable Pseudo Prototypes
Prototype Reconstruction Network
Pseudo Prototype Predictions
Pseudo Prototype Gradients
Global Buffer
Added in Previous Tasks
Added in Current Task
Each Round
Learn Current Graph
Replay Local Buffer
Backward $\mathcal{L}_{new}+\mathcal{L}_{old}$
Current Task $\mathcal{G}_A^3$
Local Buffer
Local GNN
Predictions
K-Nearest Neighbor
Local GNN from each Clients
Global GNN
Evolution Trajectories
Global Buffer Predictions
Global Buffer Graph
$\mathcal{L}_{div}$ Backward
Last Round
Local GNN
Global GNN
Embedding Space
Coverage Maximization
Coverage = 6
Coverage = 5

Figure 3: Overview of our proposed POWER framework. We follow a class-incremental setting, where each client collects an evolving graph into which unseen classes are continually introduced. Node colors indicate different labels. (1) To tackle LGF, POWER stores experience nodes with maximum local-global coverage in *the last round of the old task* and replays them in *each round of the new task*; (2) To tackle GEC, POWER reconstructs pseudo prototypes via client-server collaboration in *the first round of each task*. Based on this, POWER constructs a global buffer graph and applies trajectory-aware knowledge transfer to recover the global GNN's lost knowledge in *each round of each task*.

Client 1 Task 3) significantly impairs performance on earlier tasks (i.e., Client 1 Tasks 1 and 2). This demonstrates that in the FCGL scenario, the local GNN **experiences local graph forgetting similar to that of centralized CGL**; and (2) *Global Expertise Conflict (GEC)*. From the **Single GNN** aspect, incorporating global parameters can degrade the performance on the current task (i.e., Client 1 Task 3) while improving the performance on previous tasks (i.e., Client 1 Tasks 1 and 2). However, these improvements are limited when considering **Multiple GNNs** aspect. For example, in Task 1, the global GNN is outperformed by the local GNN of Client 3; in Task 2, it is outperformed by the local GNN of Client 2. This phenomenon occurs because each client's local GNN develops expertise shaped by its unique evolution trajectory, and this expertise leads to conflicts during parameter aggregation, where **the global GNN fails to capture it adequately**. Thus, GEC hinders FCGL's ability to adapt to new tasks and mitigate forgetting of previous ones.

In summary, both **LGF** and **GEC** present non-trivial challenges that critically restrict the performance of GNNs in FCGL, which is crucial to be addressed in building an effective FCGL framework.

4 POWER: The Proposed Framework

In this section, we introduce POWER, the first FCGL framework designed for learning on multiple decentralized evolving graphs. We first provide an overview of POWER in Fig. 3. Afterward, we delineate the architecture of POWER based on its two design objectives. Specifically, in Sec. 4.1, we detail the maximum local-global coverage-based experience node selection and replay mechanisms that POWER employs to *address LGF at local clients*. In Sec. 4.2, we expound on the pseudo prototype reconstruction mechanism and the graph evolution trajectory-aware knowledge transfer process, which is employed to *tackle GEC at central server*.

4.1 Local-Global Coverage Maximization

To tackle the challenge of LGF, POWER employs a strategic selection and storage of experience nodes from prior tasks, enabling targeted replay during local training on new tasks. We begin by introducing the experience node selection strategy.

Experience Node Selection. To ensure that selected experience nodes capture knowledge from previous tasks, we build upon the Coverage Maximization (CM) strategy [56] with further modifications tailored to FCGL settings. The conventional CM is based on the intuitive assumption that nodes within the same category share similar properties in the embedding space, with many nodes of the same class naturally gathering near their class's representative nodes. As for FCGL, we can utilize the embeddings from both the local and global GNNs to fully consider both local and global knowledge. Specifically, consider the k-th client training on its t-th local task T_t^k; the local graph is denoted as G_t^k. At the last training round, we calculate the local-global node embeddings $\mathbf{Z}$ as follows:

$$\begin{aligned}\mathbf{H} &= \mathrm{GNN}(\mathbf{X}_t^k, \mathbf{A}_t^k|\Theta^k),\\ \mathbf{H}^g &= \mathrm{GNN}(\mathbf{X}_t^k, \mathbf{A}_t^k|\Theta^g),\\ \mathbf{Z} &= \alpha\mathbf{H}+(1-\alpha)\mathbf{H}^g,\end{aligned} \tag{2}$$

where $\mathbf{H}$ and $\mathbf{H}^g$ are node embeddings from local and global GNNs with parameters Θ^k and Θ^g, respectively. α is the trade-off factor for combining local and global embeddings. The local-global coverage $C(v_i)$ for each node v_i in the training set $\mathcal{V}^{\text{lbl}}$ can be calculated as:

$$C(v_i) = |\{v_j \mid v_j \in \mathcal{V}^{\text{lbl}}, \mathbf{y}_i = \mathbf{y}_j, d(\mathbf{z}_i, \mathbf{z}_j) < \epsilon E(v_i)\}|, \tag{3}$$

where $d(\cdot, \cdot)$ measures the Euclidean distance between the embeddings of v_i and v_j, ϵ serves as a threshold distance, and $E(v_i)$ denotes the average pairwise distance between the embeddings of training nodes in the same class as v_i and the central node v_i.

Subsequently, for each class c in the current task, we select b experience nodes from the set of nodes in class c with maximum local-global coverage. The experience node set of class c is denoted as $\mathcal{B}_c$, which is formulated as follows:

$$\mathcal{B}_c = \operatorname{argmax}_{\{v_{c_1},\ldots,v_{c_b} \mid v_{c_1},\ldots,v_{c_b} \in \mathcal{V}_c^{\text{lbl}}\}} \sum\nolimits_{i=1}^{b} C(v_{c_i}), \tag{4}$$

where $\mathcal{V}_c^{\text{lbl}} = \{v_i | v_i \in \mathcal{V}^{\text{lbl}}, \mathbf{y}_i = c\}$, b represents the buffer size allocated per task and class, constrained by the storage capacity budget. In our experiments, we set $b = 1$ as the default. Eq. (4) poses an NP-hard challenge, which we address using a greedy algorithm. At each iteration, the algorithm picks the node with the highest local-global coverage from the remaining unselected nodes in $\mathcal{V}_c^{\text{lbl}}$, repeating until b nodes are selected.

Finally, for each class c in task T_t^k, the experience node set $\mathcal{B}_c$ is stored into the local buffer $\mathcal{B}$ for replay (i.e., $\mathcal{B} = \mathcal{B} \cup \mathcal{B}_c$).

Experience Node Replay. POWER's local training process pursues two primary objectives. First, to adapt to the current task T_t^k, the local GNN minimizes the cross-entropy loss over the labeled training set $\mathcal{V}^{\text{lbl}}$, which can be calculated as follows:

$$\begin{aligned} \hat{\mathbf{Y}} &= \text{GNN-CLS}(\mathbf{X}_t^k, \mathbf{A}_t^k \mid \Theta^k), \\ \mathcal{L}_{\text{new}} &= -\sum\nolimits_{v_i \in \mathcal{V}^{\text{lbl}}} \mathbf{y}_i \log \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \end{aligned} \tag{5}$$

where $\hat{\mathbf{Y}}$ denotes the soft labels predicted by the local GNN for the current task nodes, Θ^k represents the local GNN parameters, and $\mathbf{y}_i$ is the ground-truth label for a specific training node v_i.

When the current task is not the first task (i.e., $t \neq 1$), the client also optimizes a second objective, aiming at retaining knowledge from previously learned tasks. To achieve this, the client replays experience nodes stored in the local buffer $\mathcal{B}$, which captures representative samples from prior tasks. The replay objective is the cross-entropy loss over these stored nodes, as formulated below:

$$\begin{aligned} \hat{\mathbf{Y}} &= \text{GNN-CLS}(\mathbf{X}_{\mathcal{B}}, \mathbf{I} \mid \Theta^k), \\ \mathcal{L}_{\text{old}} &= -\sum\nolimits_{v_i \in \mathcal{B}} \mathbf{y}_i \log \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \end{aligned} \tag{6}$$

where $\hat{\mathbf{Y}}$ represents the soft labels predicted by the local GNN for the buffer nodes, while $\mathbf{X}_{\mathcal{B}}$ denotes the feature matrix of nodes in the local buffer, and $\mathbf{y}_i$ is the ground truth for a specific buffer node v_i. Notably, since we store only the node features without including any topological structures, the adjacency matrix for buffer nodes is set as the identity matrix $\mathbf{I}$, meaning that each buffer node is predicted in isolation.

Finally, the loss function of local training is defined as:

$$\mathcal{L} = \beta \mathcal{L}_{\text{new}} + (1 - \beta) \mathcal{L}_{\text{old}}, \tag{7}$$

where β denotes the trade-off parameter to flexibly balance adaptation to new tasks with a replay of previous tasks.

4.2 Trajectory-aware Knowledge Transfer

As discussed in Sec. 3, GEC arises from knowledge conflicts during the aggregation of parameters from local GNNs trained on divergent graph evolution trajectories. Consequently, for certain classes, the global model exhibits sub-optimal performance compared to clients with expertise in those classes. Motivated by this, POWER employs two modules to address GEC: **(1)** *Pseudo Prototype Reconstruction*, which enables the server to discern each client's evolution trajectory and expertise, and **(2)** *Trajectory-aware Knowledge Transfer*, which utilizes multi-client trajectories and expertise to recover the lost knowledge of the global GNN.

Pseudo Prototype Reconstruction Mechanism. First, to capture the expertise of each client, POWER calculates the prototype (i.e., averaged node feature) for each class on the client side. Specifically, consider the k-th client on its first communication round of t-th local task T_t^k. We denote the labeled node set as $\mathcal{V}^{\text{lbl}}$, with $\mathcal{V}_c^{\text{lbl}} = \{v_i \mid v_i \in \mathcal{V}^{\text{lbl}}, y_i = c\}$ representing the subset of nodes labeled as class c. The prototype of class c is denoted as P_c, formulated as:

$$P_c = \frac{1}{|\mathcal{V}_c^{\text{lbl}}|} \sum\nolimits_{v_i \in \mathcal{V}_c^{\text{lbl}}} \mathbf{x}_i, \tag{8}$$

However, directly transmitting class prototypes to the server poses significant privacy risks. To mitigate this, POWER adopts a privacy-preserving strategy where clients send prototype gradients instead, allowing the server to reconstruct pseudo prototypes via gradient matching. This enhances security for two reasons: (1) prototype gradients come from a randomly initialized network, making it hard to extract meaningful information without additional context, and (2) pseudo prototypes mimic the gradient patterns of true prototypes but lack identifiable numerical features that can be recognized by humans. Specifically, the client feeds the computed prototype $\mathbf{P}_c$ into an L-layer randomly initialized gradient encoding network Γ with parameters $\{\Omega_i\}_{i=1}^{L}$ to obtain predictions. We then backpropagate to compute the gradient $\nabla\Gamma_c$, whose i-th element ∇_{Ω_i} can be computed as follows:

$$\nabla_{\Omega_i}\Gamma_c = \nabla_{\Omega_i}(-\mathbf{y}_c \log \hat{\mathbf{y}}_c + (1 - \mathbf{y}_c) \log(1 - \hat{\mathbf{y}}_c)), \tag{9}$$

where $\hat{\mathbf{y}}_c = \Gamma(P_c | \{\Omega_i\}_{i=1}^{L})$ denotes the predicted soft label and $\mathbf{y}_c$ is the one-hot ground-truth vector representing class c.

When the server receives the prototype gradient $\nabla\Gamma$, it reconstructs pseudo prototypes via gradient matching. Specifically, consider a trainable pseudo prototype $\hat{\mathbf{P}}$ initialized by a standard Gaussian noise $\mathcal{N}(0, 1)$, the server obtains the pseudo prototype prediction by feeding $\hat{\mathbf{P}}$ into a prototype reconstruction network Γ, which is the same as the gradient encoding network used by all local clients (i.e., also parameterized by $\{\Omega_i\}_{i=1}^{L}$). Subsequently, the server backpropagates the cross-entropy loss to compute the pseudo prototype gradients $\nabla\hat{\Gamma}$, whose i-th element ∇_{Ω_i} is formulated as follows:

$$\nabla_{\Omega_i}\hat{\Gamma} = \nabla_{\Omega_i}(-\mathbf{y} \log \hat{\mathbf{y}} + (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}})), \tag{10}$$

where $\hat{\mathbf{y}} = \Gamma(P | \{\Omega_i\}_{i=1}^{L})$ is the predicted soft label of pseudo prototype and $\mathbf{y}$ is the one-hot vector of a specific class c. Notably, the class c of prototype gradient $\nabla\Gamma$ can be inferred with the symbol of prototype gradient (i.e., $c = \operatorname{argmin}_i \nabla_{\Omega_{L_i}} \Gamma$).

Based on Eqs. (9) and (10), the trainable pseudo prototype is optimized via the gradient matching loss $\mathcal{L}_{\text{gm}}$, defined as:

$$\mathcal{L}_{\text{gm}} = \sum\nolimits_{i=1}^{L} \|\nabla_{\Omega_i}\Gamma - \nabla_{\Omega_i}\hat{\Gamma}\|^2. \tag{11}$$

Finally, the reconstructed pseudo prototypes are stored in global buffer $\mathcal{B}^{\text{g}}$ (i.e., $\mathcal{B}^{\text{g}} = \mathcal{B}^{\text{g}} \cup \{\hat{\mathbf{P}}\}$).

Second, we define the evolution trajectory of each client through a cumulative label distribution, which aggregates task-specific label distributions across the client's task sequence, applying a gradual decay to earlier tasks to simulate LGF during local training. Formally, the evolution trajectory for the k-th client on task T^t is represented as $\mathbf{q}_t^k$, defined as follows:

$$\begin{gathered}\forall i \in \{1, ..., t\}, \mathbf{p}_i^k \propto \{\sum\nolimits_{v_j \in \mathcal{V}^{\text{lbl}}} \mathbf{1}_{y_j = c}\}_{c \in C}, \\ \mathbf{q}_t^k = \sum\nolimits_{i=1}^{t} \phi^{t-i} \mathbf{p}_i^k,\end{gathered} \tag{12}$$

where ϕ represents the decay coefficient that attenuates the contributions of past tasks, C denotes the total classes, $\mathbf{q}_i^k$ denotes the label distribution of task T_i^k, and $\mathbf{1}_{y_j=c}$ takes the value 1 if $y_j = c$ holds, and 0 otherwise. Notably, Eq. (12) can be computed iteratively without storing nodes from previous tasks.

Trajectory-aware Knowledge Transfer. Our key insight is to utilize multi-client expertise and trajectories to recover the lost knowledge of the global GNN. First, based on the global buffer $\mathcal{B}^{\text{g}}$, the server constructs a global buffer graph G^{g} via the K-Nearest Neighbor strategy to explore the global input space. Specifically, the feature matrix of the global buffer is denoted as $\mathbf{X}^{\text{g}}$, the constructed adjacency $\mathbf{A}^{\text{g}}$ can be calculated as:

$$\begin{gathered}\mathbf{H} = \sigma(\mathbf{X}^{\text{g}} \mathbf{X}^{\text{g}^T}), \\ \mathbf{A}^{\text{g}}[u, v] = \begin{cases} 1, \text{if } v \in \text{TOPK}(\bar{k}, \mathbf{H}[u, :]), \\ 0, \text{otherwise}, \end{cases}\end{gathered} \tag{13}$$

where $\sigma(\cdot)$ represents the element-wise sigmoid function, and $\text{TOPK}(\bar{k}, \cdot)$ selects the index of the first $\bar{k}$ largest items.

Subsequently, POWER leverages the global buffer graph $\mathcal{G}^{\text{g}}$ along with the set of clients' evolution trajectories $\mathbf{q}_t = \{\mathbf{q}_t^k\}_{k=1}^K$ to facilitate knowledge transfer. The extent to the global model's integration of expertise from each local GNN is guided by the client's evolution trajectory. Specifically, the knowledge transfer loss for training on task T_t is defined as:

$$\begin{gathered}\forall k \in \{1, ..., K\}, \hat{\mathbf{Y}}^k = \text{GNN-CLS}(\mathbf{X}^{\text{g}}, \mathbf{A}^{\text{g}} | \Theta^k), \\ \hat{\mathbf{Y}}^{\text{g}} = \text{GNN-CLS}(\mathbf{X}^{\text{g}}, \mathbf{A}^{\text{g}} | \Theta^{\text{g}}); \\ \mathcal{L}_{\text{div}} = \sum_{c=1}^{C} \sum_{k=1}^{K} \sum_{v_i \in \mathcal{V}^{\text{g}}} \mathbf{1}(\mathbf{y}_i = c) \frac{\mathbf{q}_t^k(c)}{\sum_{j=1}^K \mathbf{q}_t^j(c)} \text{KL}(\hat{\mathbf{y}}_i^{\text{g}} \| \hat{\mathbf{y}}_i^k),\end{gathered} \tag{14}$$

where C is the set of all classes, $\mathcal{V}^{\text{g}}$ represents the node set of the global buffer, $\mathbf{y}_i$ denotes the ground truth for the i-th node in the global buffer graph, and $\hat{\mathbf{y}}_i^g$ and $\hat{\mathbf{y}}_i^k$ are the soft labels predicted by the global GNN and the local GNN of the k-th client, respectively. The term $\frac{\mathbf{q}{t}^k(c)}{\sum{j=1}^K \mathbf{q}_t^j(c)}$ provides the proportion of class c within the evolution trajectory of the k-th client relative to all clients, and $\text{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence. The full algorithm for POWER is outlined in Algorithm 1.

Algorithm 1: POWER-Clients and Server Execution

```
Input: Number of tasks $T$, each task communication rounds $R$,
       local epochs $E$, global epochs $E_g$, global GNN parameter $\Theta^{\text{g}}$
Output: Local GNN parameters $\{\Theta^k\}_{k=1}^K$
for each training task $t$ in $1, ..., T$ do
    for each communication round $r$ in $1, ..., R$ do
        for each client $k = 1, ..., K$ do
            $\Theta^k, \nabla\Gamma_t^k, \mathbf{q}_t^k \leftarrow$ C_Exec($k, t, r, \Theta^{\text{g}}$)
        end
        $\Theta^{\text{g}} \leftarrow$ S_Exec($t, r, \{\nabla\Gamma_t^k\}_{k=1}^K, \{\mathbf{q}_t^k\}_{k=1}^K, \{\Theta^k\}_{k=1}^K$)
          // Receive uploaded messages
    end
end
/* Client Execution                                          */
Function C_Exec($k, t, r, \Theta^{\text{g}}$):
    $\Theta^k \leftarrow \Theta^{\text{g}}$                          // Update local GNN parameters
    for each local training epoch $e$ in $1, ..., E$ do
        Perform local training and replaying.                  // Eq. (7)
    end
    if $r = R$ then
        for each class $c$ in task $T_t^k$ do
            Select experience nodes $\mathcal{B}_c$    // Eqs. (2), (3), (4)
            $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}_c$                 // Store to local buffer
        end
    end
    if $r = 1$ then
        $\nabla\Gamma_t^k \leftarrow \{\}$
        for each class $c$ in task $T_t^k$ do
            Compute prototype gradients $\nabla\Gamma_c$             // Eq. (9)
            $\nabla\Gamma_t^k \leftarrow \nabla\Gamma_k^t \cup \nabla\Gamma_c$
        end
        Compute evolution trajectory $\mathbf{q}_t^k$               // Eq. (12)
        return $\Theta^k, \nabla\Gamma_t^k, \mathbf{q}_t^k$
    else
        return $\Theta^k$, None, None
    end
end
/* Server Execution                                          */
Function S_Exec($t, r, \nabla\Gamma_t, \mathbf{q}_t, \Theta$):
    if $r = 1$ then
        for $\nabla\Gamma_t^k$ in $\nabla\Gamma_t$ do
            for $\nabla\Gamma_{t_c}^k$ in $\nabla\Gamma_t^k$ do
                Reconstruct $\hat{\mathbf{P}}$ via $\nabla\Gamma_{t_c}^k$       // Eqs. (10), (11)
                /* Store to global buffer                    */
                $\mathcal{B}^{\text{g}} \leftarrow \mathcal{B}^{\text{g}} \cup \{\hat{\mathbf{P}}\}$
            end
        end
    end
    for each global training epoch $e$ in $1, ..., E_g$ do
        Construct global buffer graph $G^{\text{g}}$              // Eq. (13)
        Perform Knowledge transfer                     // Eq. (14)
    end
    return Global GNN parameters $\Theta^{\text{g}}$
end
```

5 Experiments

In this section, we present a comprehensive evaluation of POWER. We first introduce the settings of our experiments (Sec. 5.1). After that, we aim to address the following questions: **Q1**: Compared with state-of-the-art baselines, can POWER achieve better predictive performance under FCGL scenario (Sec. 5.2)? **Q2**: How much does each module in POWER contribute to the overall performance (Sec. 5.3)? **Q3**: Can POWER remain overall robust under changes in hyperparameters (Sec. 5.4)? Moreover, we also provide further experimental investigations about the efficiency (**Q4**) and sparsity robustness (**Q5**) of POWER in [57] **??** and **??**, respectively.

Table 1: Statistical information of the experimental datasets, where "#Classes pT" denotes the number of classes per task.

Datasets	#Nodes	#Features	#Edges	#Classes	#Classes pT	#Tasks	#Clients	Train/Val/Test	Description
Cora	2,708	1,433	5,429	7	2	3	3	20%/40%/40%	Citation Network
CiteSeer	3,327	3,703	4,732	6	2	3	3	20%/40%/40%	Citation Network
OGB-arxiv	169,343	128	231,559	40	10	4	5	60%/20%/20%	Citation Network
Computers	13,381	767	245,778	10	2	5	3	20%/40%/40%	Co-purchase Network
Physics	34,493	8,415	247,962	5	2	2	10	20%/40%/40%	Co-authorship Network
Squirrel	5,201	2,089	216,933	5	2	2	10	48%/32%/20%	Wiki-page Network
Flickr	89,250	500	899,756	7	2	3	10	60%/20%/20%	Image Network
Roman-empire	22,662	300	32,927	18	6	3	5	50%/25%/25%	Article Syntax Network

5.1 Experimental Setup

We introduce 8 benchmark graph datasets across 6 domains and the FCGL simulation strategy, followed by 10 state-of-the-art baselines and detailed metrics descriptions. For further statistics and experimental details, see [57] **??**.

Datasets and Simulation Strategy. We evaluate POWER on 8 benchmark graph datasets across 6 domains, including citation networks (Cora, Citeseer [44], OGB-arxiv [13]), co-purchase (Computers [34]), co-authorship (Physics [34]), wiki-page (Squirrel [27]), image (Flickr [48]), and article syntax networks (Roman-empire [28]). The decentralized evolving graph simulation strategy is detailed in Sec. 3. For each dataset, client counts, the class counts of each task, and data splits are defined based on node and class counts of the original dataset.

Simulation Details. As mentioned in Sec. 3 and Sec. 5.1, the decentralized evolving graph simulation strategy is a 2-step process, which first uses the Louvain [4] algorithm to partition the original graph dataset into multiple subgraphs. These subgraphs are subsequently distributed to different clients, all of which are further divided into multiple class-incremental tasks. Each task comprises multiple classes of nodes, discarding any surplus classes and removing inter-task edges while preserving intra-task connections. For each dataset, the number of clients, class per task, and data splits are defined based on the node and class counts of the original graph, as provided in Table. 1.

Baselines. We summarize 10 baselines into 3 categories as follows: **(1) FL/FGL Fine-tuning**, including FedAvg [25], FedSage+ [52], Fed-PUB [1] and FedGTA [18], which simply replace the local client training process of the FL/FGL algorithm with fine-tuning on sequential tasks; **(2) Federated CGL**, including Fed-ERGNN, FedTWP and Fed-DSLR, which combines the naive FedAvg algorithm with three representative CGL methods, ERGNN [56], TWP [20] and DSLR [6], respectively; **(3) Federated Continual Learning for CV**, which includes GLFC [8], TARGET [50] and LANDER [36].

Evaluation Metrics. We adopt two primary metrics commonly used in continual learning benchmarks [6, 20, 56] to evaluate model performance across sequential tasks: **(1) Accuracy Mean** (AM), defined as AM $= \frac{1}{T}\sum_{i=1}^{T}\mathbf{A}_{T,i}$, and **(2) Forgetting Mean** (FM), given by FM $= \frac{1}{T-1}\sum_{i=1}^{T-1}(A_{i,i} - A_{T,i})$, where T represents the total number of tasks. In standard CGL, $\mathbf{A}_{i,j}$ denotes the accuracy on task j following the completion of task i. In our FCGL framework, each client's local GNN predicts its own task, and we compute a weighted average of $\mathbf{A}_{i,j}$ across clients, based on their sample counts. AM measures the average accuracy on each task after training on the final task, with higher values indicating stronger performance. FM captures the mean performance drop across tasks as new ones are learned, with lower values indicating better retention.

5.2 Performance Comparison (*Answer for* Q1)

To answer **Q1**, we present a performance comparison of our proposed POWER framework against various baselines in Table. 2. As shown, POWER consistently outperforms all baselines across both AM and FM metrics. We provide a detailed discussion of the reasons behind the sub-optimal performance of each baseline category.

Comparison to FL/FGL Fine-tuning. These methods lack targeted designs to alleviate LGF across tasks, resulting in poor retention of prior performance and consistently low AM and high FM. Algorithms originally designed for FGL (e.g., FedSage+, Fed-PUB, FedGTA) often perform worse than the simpler FedAvg, due to their complex focus on current tasks, which can lead to overfitting and worsen forgetting of previous tasks as local graphs evolve.

Comparison to Federated CGL. These methods combine the centralized CGL algorithm with FedAvg, mitigating LGF but not effectively addressing GEC. While they retain some memory of previous tasks, they still show a significant performance gap compared to POWER (e.g., Fed-ERGNN vs. POWER on Cora), mainly due to GEC, as discussed in Sec. 3. Notably, FedTWP often matches FedAvg, losing most ability to retain previous task knowledge. We attribute this to its regularization of task-sensitive parameters, disrupted by the replacement of local parameters with the global model in FCGL.

Comparison to Federated Continual Learning for CV. These methods are tailored for federated continuous learning in computer vision but show significant shortcomings in the FCGL scenario. While they outperform most fine-tuning methods, they lag behind Fed-ERGNN and POWER. This is due to their heavy reliance on image data augmentation or the need for complex generative models to create pseudo images. In FCGL, simply removing data augmentation or using a basic GNN-based model doesn't yield satisfactory results. Moreover, these methods fail to effectively address GEC.

Table 2: Performance comparison of POWER and baselines, where the best and second results are highlighted in bold and underline.

Dataset	Cora		CiteSeer		OGB-arxiv		Computers		Physics		Squirrel		Flickr		Roman-empire	
Methods	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓	AM ↑	FM ↓
FedAvg [25]	40.14 ±1.01	78.92 ±1.84	42.50 ±0.73	59.30 ±1.11	21.90 ±0.99	57.57 ±1.54	19.81 ±0.07	94.94 ±1.71	60.63 ±1.80	75.98 ±3.58	32.33 ±0.95	59.03 ±2.67	27.46 ±2.00	37.35 ±3.02	17.19 ±0.92	42.11 ±1.60
FedSage+ [52]	30.41 ±2.39	87.63 ±0.28	33.70 ±4.51	58.76 ±6.29	15.22 ±0.83	56.87 ±1.21	12.22 ±1.77	95.19 ±2.35	58.24 ±2.18	79.82 ±3.04	29.52 ±2.18	79.82 ±3.04	25.58 ±2.54	36.12 ±2.07	13.35 ±1.33	40.27 ±3.15
Fed-PUB [1]	30.02 ±1.28	90.88 ±0.35	33.48 ±2.34	73.66 ±2.38	15.17 ±0.15	56.94 ±0.74	11.93 ±1.54	95.46 ±0.46	58.75 ±0.62	79.72 ±1.24	28.82 ±0.42	43.52 ±0.56	25.47 ±1.10	36.27 ±1.27	11.56 ±0.20	41.60 ±1.02
FedGTA [18]	38.62 ±0.66	80.99 ±1.31	47.00 ±0.84	54.05 ±1.65	16.80 ±0.62	62.89 ±1.83	19.48 ±0.60	96.40 ±3.35	61.62 ±2.07	74.10 ±4.09	30.84 ±0.38	42.47 ±2.29	27.78 ±1.82	40.05 ±1.74	14.16 ±1.29	46.54 ±0.66
Fed-TWP [20]	40.34 ±1.10	78.67 ±1.63	42.38 ±1.33	58.94 ±1.48	22.44 ±0.35	57.31 ±1.42	19.02 ±1.81	94.11 ±3.65	61.43 ±2.22	74.38 ±4.45	32.34 ±0.55	57.08 ±3.84	30.43 ±4.20	37.41 ±6.37	17.58 ±0.86	42.02 ±1.36
Fed-ERGNN [56]	60.40 ±2.78	45.68 ±4.20	49.92 ±2.02	39.19 ±2.41	32.63 ±3.48	48.78 ±5.75	39.79 ±7.00	20.27 ±13.82	85.23 ±2.74	26.02 ±5.56	35.13 ±2.12	41.85 ±4.38	31.79 ±3.93	22.52 ±5.85	24.71 ±0.87	27.87 ±3.56
Fed-DSLR [6]	61.21 ±3.25	43.77 ±5.10	49.55 ±2.16	40.92 ±3.27	31.69 ±4.32	50.32 ±3.22	40.32 ±5.30	18.56 ±8.31	84.75 ±5.74	28.12 ±6.56	35.21 ±3.15	41.33 ±5.27	31.55 ±2.47	24.18 ±6.30	24.32 ±2.87	28.22 ±4.13
GLFC [8]	52.04 ±5.34	55.80 ±7.13	46.30 ±3.27	51.24 ±4.35	30.27 ±4.34	52.43 ±5.25	38.43 ±7.62	23.56 ±3.27	67.01 ±4.87	60.97 ±4.22	35.99 ±6.78	40.21 ±4.28	30.80 ±3.74	35.27 ±6.03	16.86 ±3.07	41.52 ±4.85
TARGET [50]	51.64 ±4.18	56.69 ±2.37	44.62 ±2.14	56.95 ±3.15	30.52 ±5.19	53.02 ±4.20	37.75 ±1.54	26.82 ±1.79	75.48 ±2.06	19.88 ±3.18	32.75 ±2.03	51.59 ±3.11	31.07 ±3.24	25.12 ±2.20	14.37 ±1.11	45.02 ±1.95
LANDER [36]	52.85 ±5.07	53.57 ±3.24	45.93 ±3.20	52.45 ±4.33	30.45 ±4.12	54.31 ±3.73	38.13 ±2.64	24.27 ±3.28	73.24 ±4.51	22.46 ±4.20	33.21 ±3.35	52.03 ±5.17	30.58 ±4.12	28.32 ±3.27	15.42 ±2.35	44.27 ±3.53
POWER (Ours)	**65.74 ±5.11**	**28.10 ±4.95**	**54.47 ±5.21**	**28.94 ±3.19**	**36.19 ±3.17**	**30.52 ±4.22**	**43.26 ±2.94**	**12.59 ±3.03**	**89.17 ±2.54**	**13.03 ±4.53**	**40.27 ±3.22**	**35.24 ±2.98**	**35.79 ±4.15**	**18.13 ±3.24**	**26.54 ±2.35**	**16.23 ±2.26**

Table 3: Ablation study results on the Cora dataset.

Components	AM ↑	FM ↓
POWER	**65.74 ±5.11**	**28.10 ±4.95**
w/o LGF	43.42 ±2.27	74.25 ±3.34
w/o GEC	61.50 ±2.71	42.08 ±4.42
w/o LGF & GEC	40.14 ±1.01	78.92 ±1.84
Local CM	63.22 ±4.27	30.21 ±3.14
Non-cumulative	62.25 ±2.77	31.36 ±4.81

5.3 Ablation Study (Answer for Q2)

To address **Q2**, we analyze the composition of POWER, focusing on two key modules: **(1) LGF module**, which combines local-global coverage maximization (Eqs. (2-4)) with experience node replay (Eq. (7)); and **(2) GEC module**, which integrates pseudo prototype reconstruction (Eqs. (9-12)) and trajectory-aware knowledge transfer (Eqs. (13-14)). To evaluate each component's contribution, we conduct an ablation study on the Cora dataset. Additionally, we examine variations within each module: for the **LGF module**, we use local coverage maximization (*Local CM*), and for the **GEC module**, we replace the cumulative label distribution with a non-cumulative version (*Non-cumulative*). The results are presented in Table. 3.

Inter-module Ablation. Removing the LGF Module causes significant performance drops of 22% and 46% in AM and FM, highlighting its critical role in FCGL's effectiveness. Excluding the GEC Module leads to smaller reductions of 4% and 14%, indicating it limits further performance gains but is less impactful than LGF. Removing both modules reduces POWER to a basic FedAvg algorithm.

Intra-module Ablation. Delving deeper, we examine the internal design choices of each module. The **Local CM** variant, which forgoes global insight during coverage maximization, leads to performance degradation, confirming that our local-global coverage maximization strategy benefits from global insights to select more representative experience nodes. Similarly, the **Non-cumulative** configuration weakens POWER's ability to encode graph evolution trajectory, confirming that the cumulative label distribution is key to preserving the evolving knowledge across local GNNs over tasks.

In summary, both these modules and their designed components play a pivotal role in shaping the overall performance.

5.4 Sensitivity Analysis (Answer for Q3)

To address **Q3**, we perform a sensitivity analysis of hyperparameters in the POWER framework. First, we examine the impact of replay buffer size b (Eq. (4)), as shown in Fig. 4. As observed, larger b improves both AM and FM. Our default $b = 1$ minimizes storage costs, but in scenarios with sufficient storage, increasing b can enhance performance. Next, we analyze the sensitivity of POWER to the trade-off parameter β (Eq. (7)) and decay coefficient ϕ (Eq. (12)), both affecting task knowledge retention. Fig. 5 shows results on the Cora dataset, focusing on values near the optimal range. A large β may underplay old task knowledge, while a small ϕ may fail to capture it effectively. POWER performs stably with appropriate β and ϕ combinations, maintaining holistic performance.

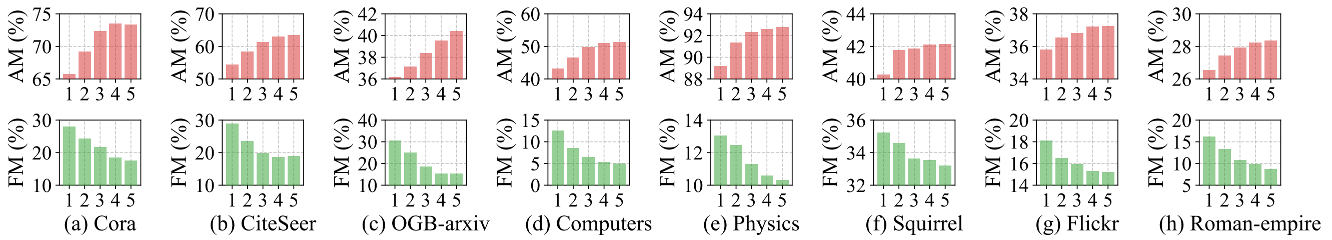


Figure 4: Sensitivity analysis for the replay buffer size b across eight benchmark graph datasets.

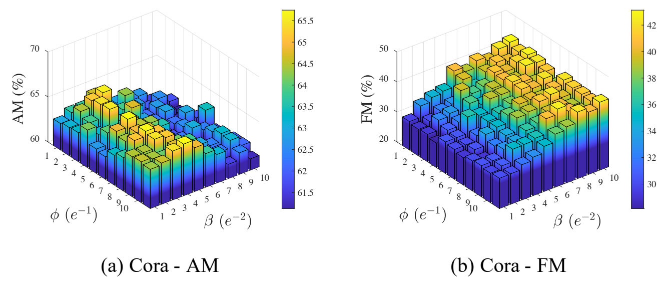


Figure 5: Sensitivity analysis for various combinations of the trade-off parameters β and the decay coefficient ϕ on the Cora dataset.

6 Related Work

Continual Graph Learning (CGL). CGL introduces a paradigm where GNNs learn from evolving graphs while retaining prior knowledge. The main challenge, catastrophic forgetting, causes performance degradation on past tasks as GNNs adapt to new data. CGL research is mainly categorized into three approaches: (1) **Experience Replay** selects and stores a minimal set of data from previous tasks, aiming to efficiently retain representative data within memory constraints. This involves deciding which data to replay and how to use it for future tasks. Early works like ERGNN [56] store experience data at the node level, using strategies such as coverage and social influence maximization, and incorporating node classification losses during new task training. DSLR [6] refines node selection with a diversity-based strategy and improves node topology via graph structure learning. Recent studies [11, 16, 21, 53–55] focus on experience data selection at the subgraph or ego-network level, preserving topological information and achieving good performance. (2) **Parameter Regularization** alleviates catastrophic forgetting by adding a regularization term during new task training to preserve parameters crucial for prior tasks. For example, TWP [20] introduces a regularization term to preserve key GNN parameters for node semantics and topology aggregation. To address forgetting caused by structural shifts, SSRM [35] introduces a regularization-based mitigation strategy. (3) **Architecture Isolation** allocates dedicated GNN parameters to each task, preventing interference when learning new tasks. Studies [31, 47] expand the GNN architecture to accommodate new graphs when its capacity is insufficient.Moreover, TPP [26] leverages Laplacian smoothing to profile graph tasks and learns a separate prompt-based classifier for each, thereby eliminating the need for replay and mitigating catastrophic forgetting in CGL.

Federated Graph Learning (FGL). Motivated by the success of federated learning in computer vision and natural language processing [43] and the demand for distributed graph learning, FGL has gained increasing attention. From the data and task perspectives, FGL studies are categorized into two settings: (1) **Graph-FL**, where each client collects multiple graphs for graph-level tasks, like graph classification. The main challenge is avoiding interference between clients' graph datasets, especially in multi-domain settings. For example, GCFL+ [41] introduces a GNN gradient pattern-aware technique for dynamic client clustering to reduce conflicts from structural and feature heterogeneity. (2) **Subgraph-FL**, where each client holds a subgraph of a global graph for node-level tasks like node classification. The key challenges are *subgraph heterogeneity* and *missing edges*. Fed-PUB [1] addresses heterogeneity by enhancing local GNNs with random graph embeddings and personalized sparse masks for selective aggregation. FedGTA [18] encodes topology into smoothing confidence and graph moments to improve model aggregation. Other studies [14, 17, 37, 58] also achieve strong results on this challenge. To address missing edges, FedSage+ [52] integrates node representations, topology, and labels across subgraphs, training a neighbor generator to restore missing links and achieve robust subgraph-FL. Other works [5, 45, 51] also excel in this area. Detailed insights into FGL research are available in surveys [9, 10, 49] and benchmark studies [12, 19, 39].

7 Conclusion

This paper pioneers the exploration of federated continual graph learning (FCGL) for node classification, bridging the gap between idealized centralized continual graph learning (CGL) setups and real-world decentralized challenges. Through empirical analysis, we investigate the data characteristics, feasibility, and effectiveness of FCGL, identifying two critical challenges: local graph forgetting (LGF) and global expertise conflict (GEC). To address these, we propose a novel FCGL framework named POWER to mitigate LGF by selectively replaying experience nodes with maximum local-global coverage, and resolve GEC through pseudo-prototype reconstruction and trajectory-aware knowledge transfer.

Acknowledgments

This work was supported by the Shenzhen Science and Technology Program (Grant No. KJZD20230923113901004), and the Fundamental Research Funds for the Central Universities, Sun Yat-sen University under Grant 24lgqb020-2.

References

[1] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized Subgraph Federated Learning. (2023).

[2] Vicente Balmaseda, María Coronado, and Gonzalo de Cadenas-Santiagoc. 2023. Predicting Systemic Risk in Financial Systems Using Deep Graph Learning. *Intelligent Systems with Applications* (2023), 200240.

[3] Dongmin Bang, Sangsoo Lim, Sangseon Lee, and Sun Kim. 2023. Biomedical knowledge graph learning for drug repurposing by extending guilt-by-association to multiple layers. *Nature Communications* 14, 1 (2023), 3570.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[5] Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. 2021. FedGL: federated graph learning framework with global self-supervision. *arXiv preprint arXiv:2105.03170* (2021).

[6] Seungyoon Choi, Wonjoong Kim, Sungwon Kim, Yeonjun In, Sein Kim, and Chanyoung Park. 2024. DSLR: Diversity Enhancement and Structure Learning for Rehearsal-based Graph Continual Learning. In *Proceedings of the ACM on Web Conference 2024*. 733–744.

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.

[8] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. 2022. Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10164–10173.

[9] Dongqi Fu, Wenxuan Bao, Ross Maciejewski, Hanghang Tong, and Jingrui He. 2023. Privacy-Preserving Graph Machine Learning from Data to Computation: A Survey. *ACM SIGKDD Explorations Newsletter* 25, 1 (2023), 54–72.

[10] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2022. Federated graph machine learning: A survey of concepts, techniques, and applications. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 32–47.

[11] Xiaoxue Han, Zhuo Feng, and Yue Ning. 2024. A topology-aware graph coarsening framework for continual graph learning. *Advances in Neural Information Processing Systems* 37 (2024), 132491–132523.

[12] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, S Yu Philip, Yu Rong, et al. 2021. FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks. In *International Conference on Learning Representations, ICLR Workshop on Distributed and Private Machine Learning.*

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems, NeurIPS* 33 (2020), 22118–22133.

[14] Wenke Huang, Guancheng Wan, Mang Ye, and Bo Du. [n. d.]. Federated Graph Semantic and Structural Learning. ([n. d.]).

[15] Woochang Hyun, Jaehong Lee, and Bongwon Suh. 2023. Anti-Money Laundering in Cryptocurrency via Multi-Relational Graph Neural Network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 118–130.

[16] Jialu Li, Yu Wang, Pengfei Zhu, Wanyu Lin, and Qinghua Hu. 2024. What matters in graph class incremental learning? An information preservation perspective. *Advances in Neural Information Processing Systems* 37 (2024), 26195–26223.

[17] Xunkai Li, Zhengyu Wu, Wentao Zhang, Henan Sun, Rong-Hua Li, and Guoren Wang. 2024. AdaFGL: A New Paradigm for Federated Node Classification with Topology Heterogeneity. *arXiv preprint arXiv:2401.11750* (2024).

[18] Xunkai Li, Zhengyu Wu, Wentao Zhang, Yinlin Zhu, Rong-Hua Li, and Guoren Wang. 2024. Fedgta: Topology-aware averaging for federated graph learning. *arXiv preprint arXiv:2401.11755* (2024).

[19] Xunkai Li, Yinlin Zhu, Boyang Pang, Guochen Yan, Yeyu Yan, Zening Li, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. OpenFGL: A Comprehensive Benchmarks for Federated Graph Learning. *arXiv preprint arXiv:2408.16288* (2024).

[20] Huihui Liu, Yiding Yang, and Xinchao Wang. 2021. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8653–8661.

[21] Yilun Liu, Ruihong Qiu, and Zi Huang. 2023. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1157–1162.

[22] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. 2021. Representational continuity for unsupervised continual learning. *arXiv preprint arXiv:2110.06976* (2021).

[23] Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7765–7773.

[24] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Vol. 24. Elsevier, 109–165.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics* (2017), 1273–1282.

[26] Chaoxi Niu, Guansong Pang, Ling Chen, and Bing Liu. 2024. Replay-and-Forget-Free Graph Class-Incremental Learning: A Task Profiling and Prompting Approach. *arXiv preprint arXiv:2410.10341* (2024).

[27] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations, ICLR.*

[28] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: are we really making progress? *International Conference on Learning Representations, ICLR* (2023).

[29] Xiao Qin, Nasrullah Sheikh, Chuan Lei, Berthold Reinwald, and Giacomo Domeniconi. 2023. Seign: A simple and efficient graph neural network for large dynamic graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2850–2863.

[30] Zongshuai Qu, Tao Yao, Xinghui Liu, and Gang Wang. 2023. A Graph Convolutional Network Based on Univariate Neurodegeneration Biomarker for Alzheimer's Disease Diagnosis. *IEEE Journal of Translational Engineering in Health and Medicine* (2023).

[31] Appan Rakaraddi, Lam Siew Kei, Mahardhika Pratama, and Marcus De Carvalho. 2022. Reinforced continual learning for graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1666–1674.

[32] Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review* 97, 2 (1990), 285.

[33] Juho Salminen. 2012. Collective intelligence in humans: A literature review. *arXiv preprint arXiv:1204.3401* (2012).

[34] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[35] Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. 2023. Towards robust graph incremental learning on evolving graphs. In *International Conference on Machine Learning*. PMLR, 32728–32748.

[36] Minh-Tuan Tran, Trung Le, Xuan-May Le, Mehrtash Harandi, and Dinh Phung. 2024. Text-enhanced data-free approach for federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 23870–23880.

[37] Guancheng Wan, Wenke Huang, and Mang Ye. 2024. Federated graph learning under domain shift with generalizable prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15429–15437.

[38] Shuzhe Wang, Zakaria Laskar, Iaroslav Melekhov, Xiaotian Li, and Juho Kannala. 2021. Continual learning for image-based camera localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3252–3262.

[39] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. FederatedScope-GNN: Towards a Unified, Comprehensive and Efficient Package for Federated Graph Learning. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD* (2022), 4110–4120.

[40] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. 2023. Graph-less Collaborative Filtering. In *Proceedings of the ACM Web Conference, WWW*. 17–27.

[41] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[42] Liangwei Yang, Shengjie Wang, Yunzhe Tao, Jiankai Sun, Xiaolong Liu, Philip S Yu, and Taiqing Wang. 2023. DGRec: Graph Neural Network for Recommendation with Diversified Embedding Generation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM*. 661–669.

[43] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13, 3 (2019), 1–207.

[44] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML*. 40–48.

[45] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. 2024. FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks. *Advances in neural information processing systems* 36 (2024).

[46] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. 2021. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*. PMLR, 12073–12086.

[47] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* (2017).

[48] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *International conference on learning representations, ICLR.*

[49] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated graph learning–a position paper. *arXiv preprint arXiv:2105.11099* (2021).

[50] Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. 2023. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4782–4793.

[51] Ke Zhang, Lichao Sun, Bolin Ding, Siu Ming Yiu, and Carl Yang. 2024. Deep Efficient Private Neighbor Generation for Subgraph Federated Learning. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 806–814.

[52] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[53] Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. 2024. Topology-aware Embedding Memory for Continual Learning on Expanding Networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4326–4337.

[54] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Sparsified subgraph memory for continual graph representation learning. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1335–1340.

[55] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2023. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[56] Fan Zhou and Chengtai Cao. 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4714–4722.

[57] Yinlin Zhu, Miao Hu, and Di Wu. 2025. POWER Technical Report. https://github.com/zyl24/FCGL_POWER/blob/main/POWER_Technical_Report.pdf. Accessed: 2025-05-22.

[58] Yinlin Zhu, Xunkai Li, Zhengyu Wu, Di Wu, Miao Hu, and Rong-Hua Li. 2024. FedTAD: Topology-aware Data-free Knowledge Distillation for Subgraph Federated Learning. *arXiv preprint arXiv:2404.14061* (2024).