A Experiment Details

In this section, we detail our experiment setup. First, we present 10 baselines in A.1. Then, we provide information on the model architecture, hyperparameters, and experiment environment in A.2, A.3, and A.4, respectively.

A.1 Baseline Descriptions

Since this paper pioneers the study of FCGL, there are no established baselines specific to this setting. To ensure a meaningful comparison, we adapt representative methods from closely related domains. In total, we evaluate 10 baseline methods, which are systematically grouped into three distinct categories based on their methodological principles. The detailed descriptions of these baseline methods are provided as follows:

(1) FL/FGL Fine-tuning. This category includes FedAvg [30], FedSage+ [58], Fed-PUB [2] and FedGTA [22], which simply replace the local client training process of the FL/FGL algorithm with fine-tuning on sequential tasks to adapt to FCGL settings.

FedAvg [30] is a simple yet effective method in FL enabling decentralized model training while preserving data privacy. A central server distributes the global model to clients for local updates. The server then aggregates the clients' models to form a new global model, which is used to update local models in the next round.

FedSage+ [58] integrates node features, link structures, and labels using a GraphSAGE [13] model with FedAvg [30] for FGL over local subgraphs. It also introduces a neighbor generator to handle cross-client missing links, improving robustness and ensuring a more comprehensive graph representation. This enhances the model's generalization across clients in FGL.

Fed-PUB [2] is a framework for personalized subgraph-FL that improves local GNNs independently instead of using a global model. It computes similarities between local GNNs with functional embeddings from random graph inputs, enabling weighted averaging for server-side aggregation. A personalized sparse mask at each client selectively updates subgraph-relevant parameters.

FedGTA [22] combines large-scale graph learning with FGL. Clients encode topology and node attributes, compute local smoothing confidence and mixed moments of neighbor features, and upload them to the server. The server then aggregates models personalized using local smoothing confidence as weights.

(2) Federated CGL. This category includes Fed-ERGNN, FedTWP, and Fed-DSLR, each combining the FedAvg [30] algorithm with a representative CGL method, ERGNN [62], TWP [25], and DSLR [7], respectively. In each case, local training leverages the corresponding CGL technique to handle evolving local graphs, while client-server communication follows the standard FedAvg process.

Fed-ERGNN is the federated variant of ERGNN [62]. ERGNN is a representative CGL approach based on the experience replay mechanism, which stores experience data at the node level, using strategies such as coverage and social influence maximization, and incorporating node classification losses during new task training.

Fed-TWP is the federated variant of TWP [25]. As an effective parameter regularization-based CGL technique, TWP preserves key GNN parameters for node semantics and topology aggregation by constraining updates through regularization.

Fed-DSLR is the federated variant of DSLR [7]. DSLR is another experience replay-based CGL technique, which refines node selection with a diversity-based strategy and improves node topology via graph structure learning.

(3) Federated Continual Learning for CV. This category includes GLFC [9], TARGET [56], and LANDER [41], which are designed for federated continual learning in computer vision. Due to format differences between images and graphs, some modules and workflows need adaptation for FCGL.

GLFC [9] is the first method to learn a global class-incremental model in FL, addressing both local and global catastrophic forgetting. It introduces a prototype communication mechanism to generate images on the server side, using them as a proxy evaluation dataset to select the historical global model with the best performance on previous tasks.

TARGET [56] alleviates catastrophic forgetting by transferring knowledge of old tasks to new ones via the global model. A generator also creates synthetic images to simulate the global data distribution, eliminating the need to store real data.

LANDER [41] is a generative model-based method for federated continual learning. It uses label text embeddings (LTE) from pre-trained language models as anchor points in server-side, data-free knowledge transfer, constraining feature embeddings to mitigate catastrophic forgetting.

A.2 Model Architecture

We employ a 2-layer GAT [42] with 64 hidden units as the backbone of clients and central server. Notably, model-specific baselines like FedSage+ [58] adhere to the custom architectures specified in their original papers. For the POWER framework, the gradient encoding network and prototype reconstruction network are implemented as 4-layer MLPs with 128, 128, and 64 hidden units. To ensure compatibility with graph data or matrix inputs, we replace the visual models used in the baseline comparisons with their respective equivalent layers of GAT or MLP.

A.3 Hyperparameters

In each task, we conduct 10 communication rounds, each comprising 3 local training epochs. These epochs utilize the Adam optimizer [19] with hyperparameters set to a learning rate of 1×10^{-2}, weight decay of 5×10^{-4}, and a dropout rate of 0.5. For all considered baselines, we adopt hyperparameters from the original publications whenever possible. In cases where these are not specified, we employ an automated hyperparameter optimization using the Optuna framework [1]. For the POWER method, specific hyperparameters are fixed as follows: trade-off parameter α at 0.5, number of experience nodes per task b at 1, and the K-Nearest Neighbor parameter at 1, as per Eqs. (3), (5), and (14), respectively. Exploratory hyperparameters include the the threshold distance η set within $\{0.01, 0.1, 0.5\}$ in Eq. (4), trade-off parameter β in Eq. (8) and the past task decay coefficient ϕ varied from 0 to 1 in increments of 0.01 in Eq. (13). Finally, the pseudo prototype reconstruction process in Eq. (12) is optimized with 300 iterations using the LBFGS optimizer [24], which has an initial learning rate of 1.0. Results are reported as the mean and variance across 10 standardized runs.

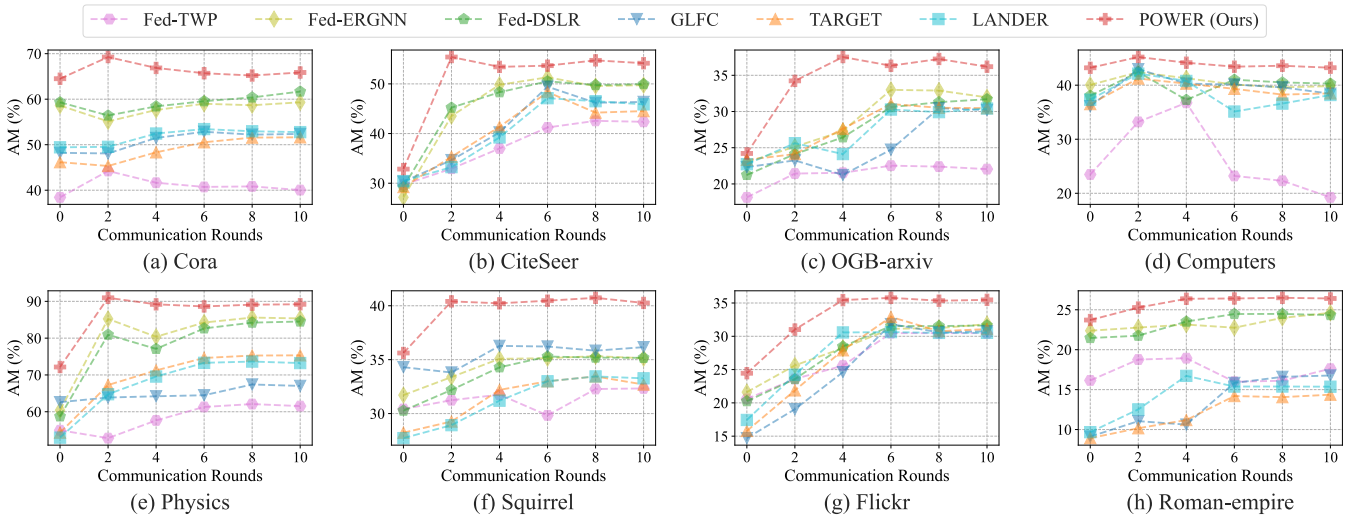

Figure 6: Convergence curves for the proposed POWER framework and six baseline methods across eight benchmark graph datasets observed during training on the final task, where "Round 0" corresponds to the state prior to initiating training on the final task.

Table 4: Theoretical communication overhead comparison of POWER and FedAvg. Notably, FedAvg represents the lowest communication cost of mainstream federated learning processes.

Method	Content	Overhead
FedAvg	GNN Parameters	$O(KTRN_{\Theta_{\text{GNN}}})$
POWER (Ours)	GNN Parameters Prototype Gradients	$O(KTRN_{\Theta_{\text{GNN}}}+$ $KCN_{\Theta_{\text{GE}}})$

Table 5: Running time comparison of POWER and baselines.

Method	Client Execution (s)	Server Execution (s)
FedAvg	0.0519	0.0005
FedSage+	0.7589	0.0005
Fed-PUB	0.0878	0.0029
FedGTA	0.0717	0.0312
Fed-TWP	0.0694	0.0005
Fed-ERGNN	0.0569	0.0005
Fed-DSLR	0.1104	0.0005
GLFC	0.1786	3.9241
TARGET	0.0783	3.1407
LANDER	0.0812	1.5676
POWER (ours)	0.0632	1.9388

A.4 Experiment Environment

The experimental machine is an Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and NVIDIA A100 with 80GB memory and CUDA 12.4. The operating system is Ubuntu 22.04.5 with 251GB memory.

B Additional Experiments

In this section, we conduct additional experiments to answer these questions: **Q4**: Does POWER demonstrate superiority in terms of efficiency (B.1)? and **Q5**: How does POWER perform under sparse settings like missing features/edges/labels and low client participation rate (B.2)?

B.1 Efficient Analysis (Answer for Q4)

To address **Q4**, we provide a detailed analysis of convergence speed, communication overhead, and runing time, as outlined below.

Convergence Speed. We present the convergence curves of POWER alongside six continual learning-related baselines: Fed-TWP, Fed-ERGNN, Fed-DSLR, GLFC, TARGET, and LANDER. Notably, methods such as **FL/FGL Fine-tuning** are excluded from this comparison, as their limitations in FCGL scenarios have already been established in Sec. 5.2. The results, illustrated in Fig. 6, demonstrate that POWER consistently converges faster than all baselines, showcasing its enhanced efficiency. This efficiency is critical for real-world applications, as it requires fewer communication rounds to achieve optimal performance, ultimately reducing communication overhead significantly. We attribute this to the effectiveness of the pseudo prototype reconstruction module in capturing localized expertise from each client. Additionally, the trajectory-aware knowledge transfer mechanism enables the global model to assimilate this expertise more effectively. These factors contribute to POWER's ability to achieve fast convergence.

Communication Overhead. To assess POWER's practicality in real-world settings, we compare its theoretical communication cost with the baseline FedAvg, which also underpins Fed-ERGNN, Fed-TWP, and Fed-DSLR, representing the minimal communication cost among mainstream federated learning approaches. Table. 4 summarizes the results, where K, T, R, C represent the number of clients, tasks, communication rounds, and total classes across all tasks, respectively. Additionally, $N_{\Theta_{\text{GNN}}}$ and $N_{\Theta_{\text{GE}}}$ denote the parameter counts for the local GNN and gradient encoding network. Both POWER and FedAvg require transmitting GNN parameters to the server, incurring a communication cost of $O(KTRN_{\Theta_{\text{GNN}}})$. POWER introduces an additional term for the prototype gradient,

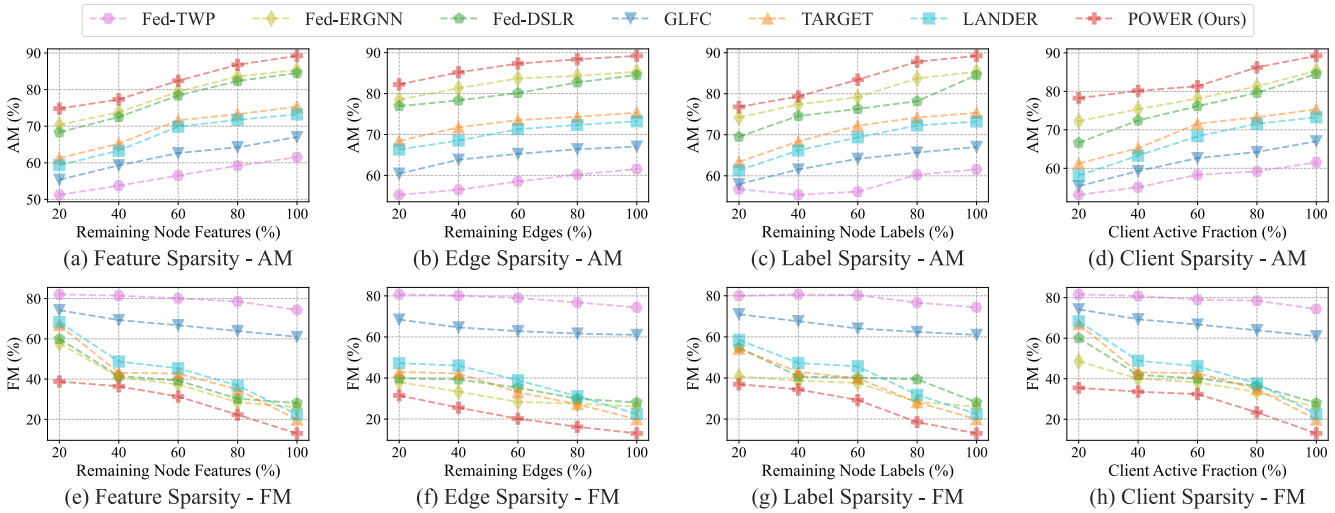

Figure 7: Performance under sparse settings for the proposed POWER framework and six baseline methods on the Physics dataset.

sent from clients to the server, with a cost of $O(KCN_{\Theta_{\text{GE}}})$. Importantly, in practical settings, $O(KCN_{\Theta_{\text{GE}}})$ is typically *far smaller* than $O(KTRN_{\Theta_{\text{GNN}}})$. This reduction can be attributed to several natural factors: **(1)** Real-world graph data tends to be complex and noisy, often requiring a large number of communication rounds R for convergence, while POWER only uploads prototype gradients in the first round of communication for each task. **(2)** Due to the personalized nature of local data sources and collection methods, each client typically lacks data for some categories and, as a result, does not transmit prototypes for all classes C. **(3)** The GNN models in practice have significantly larger parameter counts compared to our lightweight gradient encoding network (i.e., $N_{\Theta_{\text{GE}}} \ll N_{\Theta_{\text{GNN}}}$).

Runing Time. To further evaluate the efficiency of POWER, we measure its runtime against baseline methods on the Cora dataset. The results are summarized in Table. 5. Notably, (1) the single-round client execution time of POWER is lower than that of most baselines; (2) the single-round server execution time of POWER is shorter than that of federated continual learning methods for vision tasks (i.e., GLFC, TARGET, LANDER). Although it exceeds the runtime of a simple federated variant of centralized CGL, this additional overhead enables POWER to effectively mitigate GEC.

B.2 Sparsity Robustness (Answer for Q5)

In response to **Q5**, we conducted experiments to assess the resilience of the POWER framework relative to six baselines across various sparsity conditions: (1) *feature sparsity*, where node features are selectively obscured to mimic incomplete information; (2) *edge sparsity*, which simulates partial graph connectivity by randomly removing edges; (3) *label sparsity*, where only a subset of nodes have available labels to represent scenarios with incomplete supervision; and (4) *client sparsity*, characterized by limited client participation in federated communication due to factors such as network constraints or hardware limitations. These sparsity settings reflect the limitations encountered in real-world FCGL scenarios.

Sparse Feature/Edge/Label. Fig. 7 presents the results for sparse feature/edge/label settings, where panels (a)-(c) present the AM metric, and (e)-(g) present the FM metric. As observed, POWER reveals superior performance across all degrees of sparsity. We attribute this superiority to its pseudo prototype reconstruction mechanism and efficient knowledge transfer processes. Specifically, for *feature sparsity*, POWER calculates prototypes using node features within the same category, effectively compensating for missing features by utilizing the complementary data from multiple nodes. For *edge sparsity*, although POWER does not specifically target missing connections, its superior performance likely stems from the strong overall learning architecture and effective knowledge transfer process. For *label sparsity*, POWER's server-side knowledge transfer process allows each client to share its local insights in a privacy-preserving manner, which enhances the global model's ability to compensate for sparse local supervision, effectively learning from limited label information.

Sparse Client Participation. The results for *client sparsity* settings are depicted in Fig. 7, where panel (d) presents the AM metric, and (h) presents the FM metric. As observed, POWER maintains stable performance across various participation ratios. We attribute this robustness in client-sparse settings to its effective prototype reconstruction mechanism, detailed in the methods section. Specifically, when POWER receives prototype gradients from a participating client, it effectively reconstructs and stores these prototypes in a global buffer. Thus, even if a client discontinues participation in subsequent federation rounds, the knowledge from its prototypes continuously contributes to the knowledge transfer process, thereby optimizing the global model.

In summary, in the FCGL context, POWER outperforms traditional CGL methods and vision-focused federated learning algorithms, demonstrating superiority for sparsity challenges.