GECKO: A DIAGNOSTIC BENCHMARK FOR FEDERATED CONTINUAL LEARNING ON A SHARED GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated continual graph learning (FCGL) studies how distributed clients learn a sequence of graph prediction tasks without forgetting earlier knowledge. Evaluating FCGL is challenging because the way graph data are assigned to clients can change both the prediction examples and the graph context available locally, while task order introduces a separate source of variation over time. We introduce GECKO (**G**raph **E**valuation of **C**lient-**K**nowledge and **O**rder Sequences), a diagnostic benchmark for FCGL on a fixed graph distributed across clients. GECKO defines matched evaluation settings across node classification, link classification, and link prediction under Task-, Class-, and Domain-Incremental Learning. It varies client allocation and task order under explicit comparison rules, combining construction checks with diagnostics of task acquisition, forgetting, and cross-client support. Across the evaluated settings, both factors substantially affect final performance and forgetting, and their effects differ across methods and prediction problems. These results show that FCGL methods should be assessed across multiple allocation and task-order conditions rather than a single partition and sequence. GECKO provides reusable protocols and diagnostics for interpreting these comparisons.

1 INTRODUCTION

Continual learning studies how models acquire new knowledge without forgetting what was learned before (Parisi et al., 2019; De Lange et al., 2021). Continual graph learning (CGL) extends this problem to graph data, where predictions also depend on the relational context used by graph neural networks (GNNs) (Gilmer et al., 2017; Hamilton et al., 2017; Zhang et al., 2022a; Ko et al., 2025). Most CGL settings assume centralized graph access, whereas real-world graph data are often distributed across clients. Federated graph learning enables collaboration without centralizing raw data (McMahan et al., 2017; Zhang et al., 2021), but does not address continual task progression and forgetting. FCGL therefore combines federated collaboration with continual adaptation.



(a) Node-wise (b) Topology-aware

Figure 1: **Client assignment determines the graph available locally.** A node-wise split can cut more cross-client edges, whereas a topology-aware split preserves more local connectivity. Solid edges are retained within clients; dashed edges are cut because their endpoints belong to different clients.

Recent FCGL methods address forgetting through experience replay (Zhu et al., 2025), generative replay (Feng et al., 2025), topology preserving graph coarsening (Wan et al., 2026), and spectral knowledge preservation (Guo et al., 2026a). These methods tackle complementary sources of forgetting, including loss of historical knowledge, structural information degradation, and conflicting updates across clients and tasks. However their evaluations use different combinations of datasets, task construction, client partitions, and model configurations. Results are therefore difficult to compare across studies, and conclusions under one protocol need not hold under another. Moreover, these works center on node classification, leaving less evidence for link-level continual learning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Existing CGL and FGL benchmarks provide standardized settings for their respective problems, but not a common protocol for continual learning across distributed local graphs (Ko et al., 2025; Li et al., 2024c).

In graph-structured data settings, federated clients are formed by partitioning a global graph while meeting prescribed per-client class-proportion constraints (He et al., 2021; Li et al., 2024c). As shown in Figure 1, clients with similar label distributions can still produce very different local connectivity patterns. For link-level prediction, partitioning can also remove candidate supervision signals, because a link is useful for training only if both endpoint nodes are assigned to the same client. Even with the same data allocation, clients may see tasks in different orders, which changes which tasks contribute to federated optimization at each stage. A benchmark should therefore specify both (i) the local graph conditions under which client learning occurs and (ii) the task-ordering regimes through which client knowledge is aggregated.

This setting leads to three research questions. We first ask ***(RQ1)*** *How do client allocation and task order affect final performance and forgetting across methods?* To interpret these outcomes, we examine ***(RQ2)*** *When does lower forgetting reflect weaker task acquisition rather than better retention?* Finally, we ask ***(RQ3)*** *When can updates from clients still learning a task benefit clients that encountered it earlier?*

Our experiments validate the constructed client data, compare representative continual and federated methods across allocation and task-order settings, and analyze acquisition, forgetting, and the usefulness of updates from other clients. We find that matching allocation targets does not guarantee similar local graph structure, and method rankings can remain stable even when performance changes substantially. Task-order effects vary with the method and allocation, while lower forgetting can accompany weaker acquisition and worse final performance. Further comparisons show that updates from other clients can help at the evaluated checkpoints, but the benefit is not consistent across conditions. In summary, our contributions are:

- We define an FCGL evaluation framework on a fixed graph that specifies client access, task progression, and what is held fixed or changed in each comparison.
- We construct reusable benchmark conditions for selected node classification, link classification, and link prediction scenarios, with checks on the resulting client allocation, retained graph structure, and usable prediction examples.
- We provide empirical evidence on how allocation and task order affect FCGL methods, together with analyses that relate forgetting to acquisition and test the usefulness of updates from other clients.

2 Related Work

Graph continual and federated evaluation. Graph continual learning (GCL) has primarily studied task- and class-incremental node classification (Zhou & Cao, 2021; Liu et al., 2021; 2023; Choi et al., 2024), including methods that preserve relational structure and graph information across tasks (Zhang et al., 2022b; Han et al., 2024; Li et al., 2024a). Other settings include continual knowledge-graph embeddings (Daruna et al., 2021) and graph domain-incremental learning (Guo et al., 2026b), while CGLB and BeGin broaden graph continual-learning evaluation across prediction problems and incremental-learning settings, with BeGin covering node classification, link classification, and link prediction under centralized graph access (Zhang et al., 2022a; Ko et al., 2025). Federated graph learning (FGL), in contrast, studies how graph data are distributed and learned across clients, including missing cross-client neighbors (Zhang et al., 2021), personalized or topology-aware aggregation (Baek et al., 2023; Li et al., 2024b), and spectral or homophily heterogeneity (Tan et al., 2024; 2025). Federated graph-learning frameworks further provide reusable evaluation settings and client constructions (He et al., 2021; Wang et al., 2022; Li et al., 2024c).

Federated continual graph learning. FCGL couples continual adaptation with cross-client knowledge sharing, motivating methods that preserve graph knowledge under heterogeneous client data and task histories (Zhu et al., 2025; Feng et al., 2025; Wan et al., 2026; Guo et al., 2026a). Client trajectories and label skew have been studied for their effects on transfer and aggregation, but the corresponding evaluations remain concentrated on node classification (Zhu et al., 2025; Wan et al.,

Table 1: Comparison of GECKO with representative CGL, FGL, and FCGL studies. ✓ indicates that the corresponding factor is explicitly covered in the reported evaluation, while × indicates that it is outside the reported evaluation scope.

Work	Federated graph	Continual learning	Multiple prediction families	Multiple IL settings	Client allocation	Client task order
BeGin	×	✓	✓	✓	×	×
OpenFGL	✓	×	✓	×	✓	×
POWER	✓	✓	×	×	×	✓
MOTION	✓	✓	×	×	✓	×
GECKO (Ours)	✓	✓	✓	✓	✓	✓

2026). How these findings extend to link-level prediction and domain-incremental learning remains less well established under federation. GECKO adapts established scenarios to a common diagnostic protocol on client-local graphs, with matched allocation and task-order comparisons. It also includes domain-incremental link classification to examine distribution changes among observed interactions without class expansion. Table 1 summarizes these distinctions.

3 GECKO: A Matched-Condition Evaluation Framework

3.1 Prediction Settings and Scenarios

To examine these prediction problems and incremental-learning settings under a common federated protocol, GECKO defines seven scenarios on a fixed graph distributed across clients. Each scenario combines a prediction problem with an incremental learning setting, as summarized in Table 2. In node classification (NC), each client predicts labels of its assigned nodes. In link classification (LC), it predicts labels of observed interactions whose endpoints lie on the same client. In link prediction (LP), it predicts held-out links between nodes on the same client, ranked against fixed negative candidates. Therefore validation and test positive links are excluded from the message-passing graph to prevent their existence from being revealed to the model. Additional feature definitions are given in Appendix A.1.

In Task-IL, the evaluated task is known and prediction is restricted to that task. In Class-IL, task identity is hidden and prediction covers classes encountered so far. In Domain-IL, the output space remains fixed while the data distribution changes. We retain the established Task-/Class-/Domain-IL on NC, Task-/Class-IL on LC and Domain-IL on LP (Ko et al., 2025). We add LC Domain-IL setting to examine a complementary form of change at the observed link level: the prediction space remains fixed while interactions shift across label-independent structural regimes. This separates distributional change from the class expansion and allows the effect of local graph structure on labeled interactions to be studied under the same FCGL protocol. LP Task-IL and Class-IL are outside the current scope because LP retains a fixed binary edge-existence output, making class expansion less informative than in NC or LC. Time-IL is excluded as it requires graph evolution, introducing an additional source of variation beyond the fixed-graph allocation and task-order factors studied here. Every client encounters each global task once, although task order may differ across clients. Exact task mappings and inference rules are reported in Appendices A.2 and A.3. For LP, we focus on Domain-IL, where the link-existence output remains fixed while the link distribution changes. Within each scenario, GECKO varies client allocation and task order to form the comparisons summarized in Figure 2. Each condition is constructed before training and reused across methods.

3.2 What Changes Across Conditions

Client allocation. For each training group z, client proportions are drawn as

$$\mathbf{p}_z \sim \mathrm{Dir}_K(\alpha \mathbf{1}), \qquad \alpha \in \{0.1, 1, 10, 100\}.$$

Smaller α favors more concentrated allocations, while larger values favor more even allocations. The effect depends on the prediction problem. For NC and LC, changing α can change the client

Table 2: GECKO scenarios and evaluation status. Full grid denotes learner evaluation over all eight allocation-task order conditions at the scenario level. Matched anchors denote evaluation at the four endpoint conditions. Construction only denotes scenario definition and construction checks without learner-level results.

ID	Scenario	Task or domain	Tasks	Clients	Evaluation
S_1	NC-Task	8 groups of 5 classes	8	10	Matched anchors
S_2	NC-Class	Same groups as NC-Task	8	10	Full grid
S_3	NC-Domain	8 species domains	8	10	Construction only
S_4	LC-Task	3 groups of 2 labels	3	10	Matched anchors
S_5	LC-Class	Same groups as LC-Task	3	10	Full grid
S_6	LC-Domain	4 degree-based domains	4	10	Four-anchor extension
S_7	LP-Domain	2 source-domain groups	2	3	Full grid

partition, local graph structure, and prediction examples that remain valid. We therefore use α as a client-allocation parameter rather than a direct measure of label skew. For LP, the client partition and local base graph remain fixed across all values of α, and only the selected client-local training positives change. These positives may enter the training graph, so the graph context used during learning can still change.

Task order. Task order is changed after client data have been fixed. Synchronized clients encounter tasks in the same order, whereas unsynchronized clients follow predefined client-specific orders. The client partition, local data, evaluation examples, and participation schedule remain unchanged within each paired comparison. NC permutes tasks within two four-task blocks, LC permutes its three or four tasks, depending on the scenario and LP uses a fixed two-task mismatch profile. Training remains synchronous in both settings: each client receives new supervision for one task per stage, while the server may combine updates from different tasks. Unlike Joint Training, clients are not given supervision for all tasks from the start. Paired-condition definitions are given in Appendix B.1, with exact task-order and participation rules in Appendix B.4.

Constructing Client Data NC and LC define tasks and prediction examples before forming the client partition. Dirichlet allocation sets client-level training targets, while topology-aware construction assigns connected nodes together whenever possible subject to these targets and client sizes. LP instead forms the client partition from a base graph first, fixes within-client validation/test positives and their negative candidates, and then selects client-local training positives according to α. We characterize the resulting conditions through their realized allocation, retained graph structure, and usable prediction examples, linking the intended comparisons to the data available for learning. Full construction details are provided in Appendices B.2 and B.3.

4 Evaluation Setup

We distinguish scenario coverage from the primary full-grid evaluation. We use S_2, S_5, and S_7 as the core scenarios, providing one primary setting for each prediction family: NC-Class, LC-Class, and LP-Domain. Class-IL is used for NC and LC because prediction is performed without task identity, whereas LP uses Domain-IL as its binary edge-existence output remains fixed. The remaining scenarios serve complementary roles: S_1 and S_4 provide matched Task-IL references, S_6 evaluates distribution shift for observed links, and S_3 provides NC-Domain construction and validation without learner-level evaluation.

To assess the stability of learner behavior across client allocations and task orders, we evaluate the core scenarios on $\alpha \in \{0.1, 1, 10, 100\}$ under synchronized and unsynchronized task orders, yielding eight conditions per scenario. We treat α as an allocation-control parameter rather than a common measure of heterogeneity across prediction families, and therefore report the realized

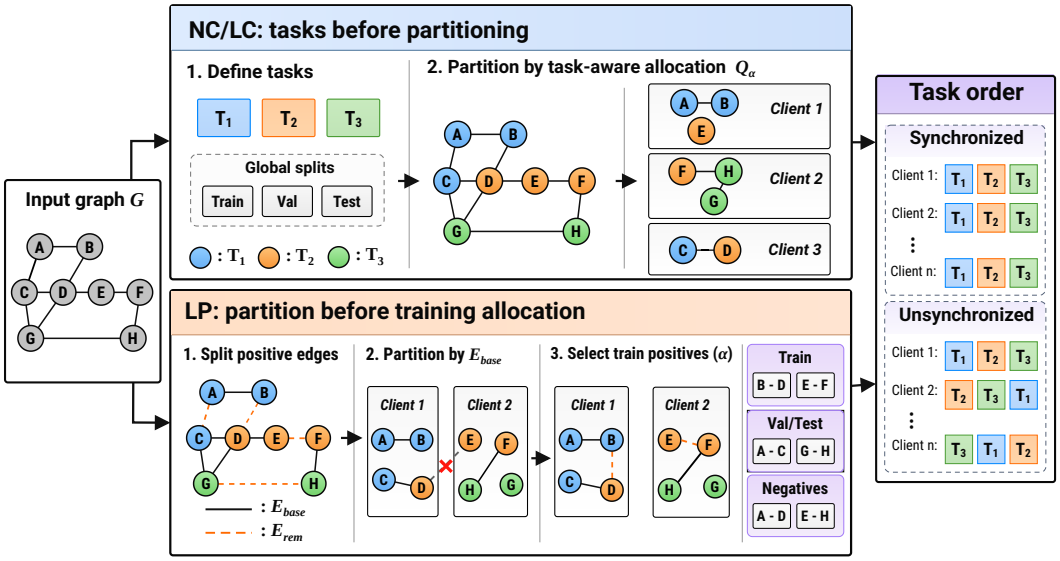


Figure 2: Overview of the GECKO construction. Changing allocation may change ownership and local graph structure for node classification (NC) and link classification (LC), while link prediction (LP) keeps ownership and the base graph fixed. Task-order comparisons use the same client data and change only the order in which tasks are encountered.

allocation, local connectivity, and usable supervision separately. Within each condition, all methods use the same constructed client data, task order, participation schedule, and training budget of 50 communication rounds and one local epoch per selected client.

Datasets. GECKO uses OGBN-Arxiv and OGBN-Proteins for NC (Hu et al., 2020), Bitcoin-OTC for LC (Kumar et al., 2016), and GEMSEC Facebook for LP (Rozemberczki et al., 2019). These datasets provide node-, edge-, and link-level predictions.

Method comparison. We train methods without a native federated optimization rule using FedAvg (McMahan et al., 2017), and retain the original federated rule for methods that define one. this statement sounds passive. We therefore compare end-to-end method behavior in terms of final performance and forgetting under matched benchmark conditions, rather than isolating individual aggregation components. Each method uses one fixed configuration across allocation levels, task orders, and seeds, based on published defaults, benchmark-specific adaptations, or available validation results. We do not retune methods for individual conditions. For eligible methods, Appendix E.2 additionally compares FedAvg with Local-only and FedProx (Li et al., 2020) training. Full method settings and resource accounting are reported in Appendices C.3 and C.4.

Metrics. For each allocation-task order condition, we report final performance P and average forgetting AFḞinal performance is measured by Accuracy for NC and LC and Hits@50 for LP, where each held-out positive link is ranked against a fixed set of negative candidates. Forgetting measures the decrease from the best performance attained after a client's first exposure to a task to its performance at the end of training, excluding pre-exposure evaluations. Across the eight-condition grid, we summarize each method by mean final performance $\overline{P}$, mean forgetting $\overline{AF}$, and the lowest final performance over the tested conditions P_{worst}. Exact definitions and aggregation rules are provided in Appendix C.2.

Repetitions. All summaries are computed separately for each seed before reporting the mean and sample standard deviation. NC and LC use three paired seeds with 10 clients, while LP uses five paired seeds with 3 clients. LP uses fewer clients to ensure that each client retains sufficient positive test links for reliable Hits@50 evaluation under the strict-local protocol. We therefore increase the number of seeds to five in LP to improve the reliability of the reported estimates. The same

seed set is reused across methods within each scenario to preserve pairing in condition and method comparisons. Detailed training and repetition settings are given in Appendix C.1.

5 Results Across Benchmark Conditions

5.1 Performance Across Conditions

To address ***RQ1***, we first examine performance variation across benchmark conditions before assessing the effects of allocation and task order. Figure 3 compares each learner's performance at the base condition ($\alpha = 100$, Synchronized) with its performance across the eight allocation-task order conditions. The base condition is not uniformly representative of the full grid. For some learners it lies near the upper end of the observed range, whereas for others it is closer to the grid mean or to lower performing conditions. By comparing the base condition with the remaining grid, we observe that the same learner can exhibit different performance as client allocation and task order vary. This indicates that evaluation under a near-uniform allocation alone may not capture its sensitivity to heterogeneous client data. Thus, performance measured under a single allocation-task order condition can provide a different view of learner behavior than evaluation across the complete grid.

Table 3: Performance across the eight allocation-order conditions. For each seed, $\overline{P}$ is the average final performance across the eight conditions, while P_{worst} is the minimum among them. Values are mean±sample SD over three seeds for NC/LC and five for LP.

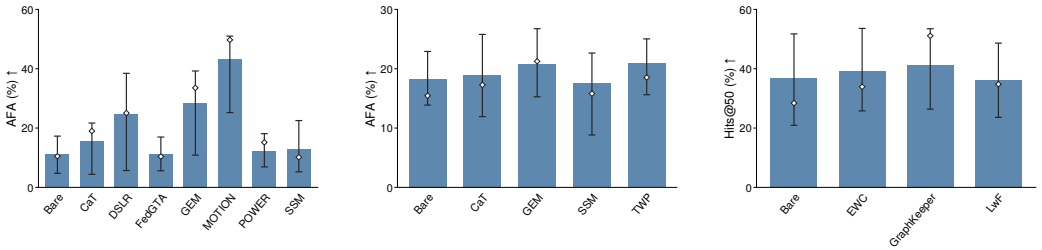
Task	Method	$\overline{P}\uparrow$	$P_{\text{worst}}\uparrow$	$\overline{\text{AF}}\downarrow$
NC OGBN-Arxiv Class-IL	Bare	11.09 ± 2.01	4.74 ± 2.37	58.57 ± 2.90
	CaT	15.46 ± 0.99	4.40 ± 2.11	31.65 ± 0.55
	DSLR	24.80 ± 1.29	5.68 ± 0.81	43.56 ± 0.52
	FedDC	5.91 ± 0.58	2.39 ± 0.25	45.32 ± 0.76
	FedGTA	11.25 ± 0.65	5.60 ± 0.68	61.76 ± 1.91
	FedPUB	1.94 ± 0.78	1.04 ± 0.59	27.50 ± 1.38
	GEM	28.38 ± 1.68	10.87 ± 5.02	39.97 ± 0.67
	MOTION	$\mathbf{43.07 \pm 1.48}$	$\mathbf{25.20 \pm 2.52}$	$\mathbf{14.27 \pm 1.49}$
	POWER	12.06 ± 0.97	6.90 ± 0.71	29.91 ± 0.65
	SSM	13.01 ± 0.63	5.24 ± 1.61	45.09 ± 0.98
	TWP	3.99 ± 0.92	1.13 ± 0.52	28.96 ± 1.10
LC Bitcoin-OTC Class-IL	Bare	18.30 ± 0.26	13.87 ± 0.71	30.68 ± 0.25
	CaT	18.91 ± 0.90	11.92 ± 2.28	$\mathbf{18.80 \pm 2.20}$
	SSM	17.51 ± 0.93	8.83 ± 1.53	21.06 ± 2.25
	TWP	$\mathbf{20.86 \pm 1.67}$	$\mathbf{15.62 \pm 3.66}$	28.68 ± 0.63
	GEM	20.85 ± 0.88	15.27 ± 1.07	24.41 ± 1.67
LP GEMSEC Facebook Domain-IL	Bare	36.84 ± 3.30	20.93 ± 7.28	5.74 ± 3.92
	EWC	39.16 ± 4.31	25.79 ± 9.45	$\mathbf{5.26 \pm 4.65}$
	LwF	36.23 ± 6.40	23.60 ± 4.70	7.91 ± 4.90
	GraphKeeper	$\mathbf{41.22 \pm 5.81}$	$\mathbf{26.36 \pm 8.85}$	7.84 ± 2.28

We next examine whether an advantage in average performance is preserved under the weakest tested condition. Table 3 shows that the separation between methods can narrow from $\overline{P}$ to P_{worst}. On NC, DSLR exceeds CaT by 9.34 percentage points in $\overline{P}$, but by only 1.28 points in P_{worst}. On LP, the gap between GraphKeeper and EWC similarly decreases from 2.06 to 0.57 points. Thus, a method's average advantage across the grid does not necessarily reflect the margin it retains under less favorable allocation-task order conditions. Considering $\overline{P}$ together with P_{worst} therefore reveals whether an observed average advantage remains substantial across the tested conditions.

5.2 Effects of Allocation and Task Order

Having established that performance can vary across benchmark conditions, we next ask whether this variation is associated with client allocation, task order, or both. We isolate the two factors

through paired comparisons. For allocation, we compare $\alpha = 0.1$ with $\alpha = 100$ under the same task order and average the two contrasts across synchronized and unsynchronized settings. For task order, we compare synchronized and unsynchronized runs on the same client data and average the contrasts across the two allocation endpoints. Differences are computed within paired seeds before averaging.



(a) S_2 (NC Class) (b) S_5 (LC Class) (c) S_7 (LP Domain)

Figure 3: Final performance across eight allocation-order conditions: (a) S_2 (NC Class), (b) S_5 (LC Class), and (c) S_7 (LP Domain). Colored bars show the grid mean, capped horizontal lines span the mean seed-wise minimum and maximum, white diamonds mark the base condition ($\alpha = 100$, synchronous task order). All summaries are computed within each seed and then averaged across seeds.

Figure 4 shows that allocation and task order affect final performance and forgetting in different ways. In NC, a more even allocation generally improves final performance, with gains of $+23.0$ percentage points (pp) for MOTION, $+22.2$ pp for GEM, and $+21.3$ pp for DSLR, but its effect on forgetting is less consistent as the same allocation change reduces forgetting for MOTION ($+12.8$ pp in $-\Delta AF$) but increases it for FedGTA (-15.6 pp). Unsynchronized task orders often reduce AF, while their effect on final performance varies across methods.

In LP, the two factors can even produce opposite responses for the same method. GraphKeeper improves under a more even allocation ($\Delta P = +6.2$ pp, $-\Delta AF = +3.8$ pp), but degrades under an unsynchronized task order ($\Delta P = -6.9$ pp, $-\Delta AF = -4.5$ pp). Bare shows the opposite order response, improving both final performance ($+13.6$ pp) and forgetting ($+9.2$ pp in $-\Delta AF$). These opposing responses show why allocation and task order should be assessed as distinct factors, without assuming that task-order mismatch is uniformly harder across learners. Higher final performance also need not coincide with lower AF. Moreover, lower AF can arise when a task was learned poorly to begin with. We therefore turn to ***RQ2*** and examine task acquisition alongside final performance and forgetting.

	Client allocation		Task order	
	ΔP	−ΔAF	ΔP	−ΔAF
MOTION	+23.0	+12.8	+2.1	+4.3
GEM	+22.2	+6.0	+3.3	+13.7
DSLR	+21.3	-1.6	+11.2	+20.5
FedGTA	+7.5	-15.6	+3.4	+18.7
TWP	+0.8	+5.6	-0.7	+19.6

(a) S_2 (NC Class)

	Client allocation		Task order	
	ΔP	−ΔAF	ΔP	−ΔAF
Bare	-1.8	+3.2	+3.2	+17.3
CaT	-3.5	-6.5	-0.3	-1.5
GEM	+2.1	+3.1	+1.7	+12.7
SSM	+2.0	+0.4	-1.3	-1.6
TWP	+1.0	+0.4	+4.6	+22.2

(b) S_5 (LC Class)

	Client allocation		Task order	
	ΔP	−ΔAF	ΔP	−ΔAF
Bare	-2.7	-1.9	+13.6	+9.2
EWC	+1.0	-2.9	+5.0	+6.5
GraphKeeper	+6.2	+3.8	-6.9	-4.5
LwF	+0.7	-4.4	+6.2	+5.6

(c) S_7 (LP Domain)

Figure 4: Paired effects of client allocation and task order across (a) S_2 (NC Class), (b) S_5 (LC Class), and (c) S_7 (LP Domain). $\Delta P = P_{\text{after}} - P_{\text{before}}$ and $\Delta AF = AF_{\text{after}} - AF_{\text{before}}$. Positive values indicate higher final performance or less forgetting. Allocation effects average the change from $\alpha = 0.1$ to 100 over both task orders, while order effects average the change from synchronized to unsynchronized order over both allocation endpoints.

5.3 Learning and Forgetting

To distinguish poor task acquisition from subsequent forgetting, we examine two anchor conditions from the eight-condition allocation-order grid: Base condition ($\alpha = 100$, Synchronized) and joint

condition ($\alpha = 0.1$, Unsynchronized). Table 4 compares four references: Prior, a non-learning predictor based on the client's observed training-label frequencies, Bare, which learns only from the current task, Simple Replay, which augments the current task with a small memory of earlier examples and All-task Training, which makes all local task supervision available from the start. Acquisition is measured at the end of the first stage when a client encounters a task, whereas All-task Training serves as a learnability reference rather than a compute-matched upper bound. Train fit is the mean training accuracy over client-task pairs after all tasks have been learned. Full protocol definitions are given in Appendix F.1.

At the base condition, the NC continual references acquire tasks well but lose much of that performance by the end of training. At joint condition, acquisition and final performance are lower, and AF also decreases, indicating that lower forgetting does not necessarily imply better learning. LC reveals a different limitation: final training fit remains weak even under All-task Training, so sequential forgetting alone cannot explain the low final performance. Under the tested configuration, weak task fitting and subsequent performance loss coexist. All-task Training also fails to exceed the non-learning Prior in mean final accuracy at either LC anchor, reinforcing that removing sequential task availability alone does not resolve the observed difficulty under this setup. Further analysis is provided in Appendix F.2.

Table 4: Learnability calibration at base and joint condition values are mean±sample SD over three paired seeds. Performance and train fit are in percent, AF is in percentage points (pp). All-task Training has no acquisition or AF.

Task	Condition	Reference	Acquisition ↑	P ↑	AF ↓	Train fit ↑
NC	Base condition	Prior	9.91 ± 0.54	9.91 ± 0.54	0.00 ± 0.00	—
		Bare	$\mathbf{86.03 \pm 0.70}$	10.49 ± 0.14	75.55 ± 0.61	11.12 ± 0.16
		Simple Replay	84.08 ± 0.17	16.26 ± 1.83	$\mathbf{67.82 \pm 1.72}$	19.84 ± 2.22
		All-task Training	—	$\mathbf{48.98 \pm 1.73}$	—	$\mathbf{66.63 \pm 1.42}$
	Joint condition	Prior	19.66 ± 3.12	9.78 ± 0.72	9.88 ± 3.04	—
		Bare	35.70 ± 4.62	5.58 ± 2.33	$\mathbf{37.48 \pm 2.54}$	6.45 ± 1.66
		Simple Replay	$\mathbf{41.92 \pm 3.95}$	11.99 ± 1.97	38.34 ± 4.01	20.82 ± 1.67
		All-task Training	—	$\mathbf{35.13 \pm 3.54}$	—	$\mathbf{60.74 \pm 1.96}$
LC	Base condition	Prior	29.56 ± 1.10	$\mathbf{23.91 \pm 1.87}$	5.65 ± 2.72	—
		Bare	$\mathbf{54.18 \pm 3.48}$	15.46 ± 2.54	38.72 ± 1.51	17.84 ± 2.40
		Simple Replay	36.85 ± 8.13	21.84 ± 2.25	$\mathbf{20.18 \pm 3.12}$	25.93 ± 2.87
		All-task Training	—	21.46 ± 0.61	—	$\mathbf{26.72 \pm 4.31}$
	Joint condition	Prior	38.49 ± 3.44	20.27 ± 2.15	18.22 ± 2.93	—
		Bare	$\mathbf{41.74 \pm 7.05}$	20.45 ± 0.49	24.61 ± 5.75	21.92 ± 2.88
		Simple Replay	39.02 ± 1.90	$\mathbf{21.92 \pm 4.48}$	$\mathbf{21.23 \pm 4.49}$	$\mathbf{28.69 \pm 3.64}$
		All-task Training	—	20.08 ± 5.00	—	25.23 ± 8.62

The relationship between acquisition and forgetting can be made explicit for the NC diagnostic. Averaging over the same client-task pairs gives

$$\mathrm{AF} = \mathrm{Acq} - P + \Gamma, \qquad \Gamma \geq 0, \tag{1}$$

where Γ measures the average improvement from acquisition to the best performance reached thereafter. Thus, a lower AF can arise not only from better retention, but also from weaker initial acquisition. To further examine this effect, Appendix F.3 provides a balanced-participation analysis across synchronized and unsynchronized task orders. We apply it only to NC because LP uses different weighting for final performance and AF. Taken together, Table 4 and equation 1 show that low final performance can arise from different failure modes: NC exhibits strong initial acquisition followed by substantial performance loss, whereas LC also shows limited task fitting even when all local task supervision is available in All-task Training. Thus, AF should not be interpreted in isolation, retention must be considered jointly with task acquisition and final performance. Finally, to address ***RQ3***, we examine whether updates from clients still learning a task can benefit clients that encountered it earlier.

5.4 CROSS-CLIENT TASK SUPPORT

Related benefits have been observed in FCL with heterogeneous task sequences (Cox et al., 2023), and cross-client knowledge transfer has been linked to reduced forgetting in FCGL (Zhu et al., 2025). Under synchronized order, participating clients receive the same current task; replay or regularization can still involve earlier knowledge. Here, we isolate support from current-task training using Bare+FedAvg. For a past task t, call clients still learning t *donors* and clients that have already learned it *receivers*. Let f_t be the mean receiver loss, evaluated on fixed local graphs and queries. Output classes and saved normalization statistics are held fixed. Write δ_t for the donors' combined contribution with the original FedAvg weights, and $x = \theta_{-t}$ for the model after adding all other clients' contributions. If f_t is differentiable with a locally β_t-Lipschitz gradient along $[x, x + \delta_t]$, the standard smoothness bound gives

$$\underbrace{f_t(x) - f_t(x + \delta_t)}_{G_t^L} \geq -\langle \nabla f_t(x), \delta_t \rangle - \frac{\beta_t}{2} \|\delta_t\|_2^2. \tag{2}$$

The first term measures alignment with the receiver's descent direction, while the second accounts for curvature over a finite update. Loss decreases if the alignment term exceeds this penalty. The receiver gradient and donor contribution depend on their respective local graphs, which can affect their alignment. Local smoothness is an assumption of the bound and accuracy effects are measured separately. We examine this support at saved checkpoints using two controls. *Drop* removes the same-task contribution and keeps the remaining FedAvg weights unchanged. *Other task* replaces it with an update from the same donors on another earlier task, scaled to the same norm. Each comparison keeps the receiver's evaluation fixed.

Table 5 provides the empirical counterpart to equation 2. At $\alpha = 100$, retaining the same-task donor contribution improves old-task accuracy relative to both removing that contribution ($+6.41$ pp) and replacing it with a norm-matched update from another task ($+11.40$ pp), with positive effects in all three seeds. At $\alpha = 0.1$, the corresponding gains are smaller and less consistent. Together, the local alignment condition in equation 2 and these matched-control comparisons show that updates from clients still training task t can support clients that encountered t earlier, and that the strength of this support depends on the allocation condition. This evidence concerns intermediate checkpoints rather than final retention, which also depends on task acquisition and subsequent updates.

Table 5: **Cross-client task support.** Old-task accuracy gain from same-task donors over matched controls (pp; mean±SD over three seeds).

Allocation	vs. Drop	vs. Other task
$\alpha = 0.1$	$+0.84 \pm 1.09$	$+2.34 \pm 3.05$
$\alpha = 100$	$+6.41 \pm 2.31$	$+11.40 \pm 4.75$

6 CONCLUSION

GECKO contributes a diagnostic benchmark for FCGL that combines explicit client access rules, validated graph construction, matched allocation and task order comparisons, and learning diagnostics. Our results show that allocation counts alone do not capture the resulting local learning problem, that similar rankings can hide large performance changes across conditions, and that lower forgetting can partly reflect weaker task acquisition. Task order effects are also method and allocation dependent, while matched checkpoint tests show that same task client updates can provide useful cross client support.

A limitation of GECKO is the relatively small number of datasets. We prioritize datasets that permit controlled and meaningful construction of the considered FCGL scenarios, rather than expanding coverage through substantial artificial modification of the underlying data. The benchmark can naturally incorporate additional datasets as suitable graph data become available. In addition, GECKO is intended for diagnosing the effects of data allocation and task order, rather than evaluating privacy mechanisms. Its local-graph protocol defines the information available to each client, while formal privacy guarantees remain complementary to, and can be incorporated independently of, the benchmark.

AI USE STATEMENT

AI tools were used only to assist with language editing and improve the clarity of the manuscript. All scientific content, analyses, results, and conclusions were produced and verified by the authors.

ETHICS STATEMENT

This work does not involve new human-subject data collection or deployment decisions. We use existing research datasets and follow their established research-use protocols.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce the GECKO benchmark construction and reported evaluations. Dataset, task, and prediction definitions are given in Appendix A; benchmark construction and validation are described in Appendix B; training protocols, evaluation metrics, method configurations, and resource accounting are provided in Appendix C. Complete results and sensitivity analyses are reported in Appendices D-E, with additional acquisition and task-order analyses in Appendix F. Appendix G describes the benchmark records and planned release of the construction and evaluation code, experiment configurations, environment specifications, and result-generation scripts.

REFERENCES

Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *International conference on machine learning*, pp. 1396–1415. PMLR, 2023.

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

Seungyoon Choi, Wonjoong Kim, Sungwon Kim, Yeonjun In, Sein Kim, and Chanyoung Park. Dslr: Diversity enhancement and structure learning for rehearsal-based graph continual learning. In *Proceedings of the ACM Web Conference 2024*, pp. 733–744, 2024.

Bart Cox, Jeroen Galjaard, Aditya Shankar, Jérémie Decouchant, and Lydia Y Chen. Parameterizing federated continual learning for reproducible research. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 478–486. Springer, 2023.

Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135, 2021.

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

Jiyuan Feng, Xu Yang, Dongyi Zheng, Weihong Han, Binxing Fang, and Qing Liao. Overcoming catastrophic forgetting in federated continual graph learning for resource-limited mobile devices. *IEEE Transactions on Mobile Computing*, 24(10):11151–11163, 2025.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Hanyao Guo, Zihan Tan, Wenke Huang, Bin Yang, and Mang Ye. Fedfst: Mitigating spectral catastrophic forgetting in federated graph continual learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1404–1414, 2026a.

Zihao Guo, Qingyun Sun, Ziwei Zhang, Haonan Yuan, Huiping Zhuang, Xingcheng Fu, and Jianxin Li. Graphkeeper: Graph domain-incremental learning via knowledge disentanglement and preservation. *Advances in Neural Information Processing Systems*, 38:1145–1171, 2026b.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

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

Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

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

George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Jihoon Ko, Shinhwan Kang, Taehyung Kwon, Heechan Moon, and Kijung Shin. Begin: Extensive benchmark scenarios and an easy-to-use framework for graph continual learning. *ACM Transactions on Intelligent Systems and Technology*, 16(1):1–22, 2025.

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 221–230. IEEE, 2016.

Jialu Li, Yu Wang, Pengfei Zhu, Wanyu Lin, and Qinghua Hu. What matters in graph class incremental learning? an information preservation perspective. *Advances in Neural Information Processing Systems*, 37:26195–26223, 2024a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

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

Xunkai Li, Yinlin Zhu, Boyang Pang, Guochen Yan, Yeyu Yan, Zening Li, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. Openfgl: A comprehensive benchmark for federated graph learning. *arXiv preprint arXiv:2408.16288*, 2024c.

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

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

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

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*, pp. 65–72, 2019.

Zihan Tan, Guancheng Wan, Wenke Huang, and Mang Ye. Fedssp: Federated graph learning with spectral knowledge and personalized preference. *Advances in Neural Information Processing Systems*, 37:34561–34581, 2024.

Zihan Tan, Guancheng Wan, Wenke Huang, He Li, Guibin Zhang, Carl Yang, and Mang Ye. Fedspa: Generalizable federated graph learning under homophily heterogeneity. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15464–15475. IEEE, 2025.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Frank Wan, Fengyuan Ran, Ruikang Zhang, Wenke Huang, Xuankun Rong, Guibin Zhang, Yuxin Wu, Bo Du, and Mang Ye. Motion: Multi-sculpt evolutionary coarsening for federated continual graph learning. *Advances in Neural Information Processing Systems*, 38:70280–70304, 2026.

Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4110–4120, 2022.

Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *Advances in neural information processing systems*, 34:6671–6682, 2021.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Cglb: Benchmark tasks for continual graph learning. *Advances in Neural Information Processing Systems*, 35:13006–13021, 2022a.

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

Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4714–4722, 2021.

Yinlin Zhu, Miao Hu, and Di Wu. Federated continual graph learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 4203–4213, 2025.

A Data and Prediction Definitions

This appendix specifies the data, task and domain mappings, and prediction rules underlying the scenarios summarized in the main paper. Construction effects such as retained topology and retained client-local supervision are reported with the construction analysis rather than repeated here.

A.1 Datasets and Prediction Data

The benchmark uses OGBN-Arxiv and OGBN-Proteins for node classification (NC) (Hu et al., 2020), Bitcoin-OTC for link classification (LC) (Kumar et al., 2016), and GEMSEC Facebook for link prediction (LP) (Rozemberczki et al., 2019). Dataset-setting pairings and inherited task cardinalities follow CGLB and BeGin where they are compatible with strict-local federated evaluation (Zhang et al., 2022a; Ko et al., 2025).

Node data. We use the standard OGB release of OGBN-Arxiv, containing 169,343 nodes, 40 classes, and the supplied 128-dimensional node features. The benchmark uses the official split, adds reverse arcs for message passing, and includes one self-loop per node. Labels, split indicators, and publication years are not model inputs. OGBN-Proteins contains 132,534 nodes with 112 binary outputs and eight edge-feature channels. The benchmark uses the fixed within-domain split shown in Table 6. Node features are obtained by averaging outgoing edge features over the edges visible inside each owned subgraph. The resulting Domain-IL evaluation is therefore not directly comparable with the official OGB leaderboard protocol.

Interaction and link data. Bitcoin-OTC contains 35,592 directed interactions over 6,005 nodes. Ratings, timestamps, and the structural-domain score are not supplied as model inputs. GEMSEC Facebook provides eight source categories with 1,380,293 positive rows. The reported data version contains 134,826 nodes because consecutive category boundaries share seven node indices, it is therefore not a disjoint union of the eight category graphs. The reported LP results correspond to this data version.

Table 6: Prediction populations before client ownership for NC and LC. Task- and Class-IL share the same populations. Counts denote prediction targets after scenario-level eligibility filtering, client-local retention after ownership is a construction outcome.

Scenario	Eligible targets	Eligible train pool	Validation	Test
S_1/ S_2	169,343	90,941	29,799	48,603
S_3	132,534	53,016	13,253	66,265
S_4/ S_5	9,982	8,002	970	1,010
S_6	35,592	28,474	3,559	3,559

For S_4/S_5, the 8,002 training interactions form the pool available before client allocation. We then keep a fixed training budget of 398 interactions, corresponding to about 5% of each target class. Client quotas are defined on these 398 selected interactions. Therefore, allocation fidelity is measured against 398 requested training targets rather than all 8,002 eligible interactions.

LP uses a partition-first design and does not follow the same fixed global train/validation/test split. For seed 0, 138,298 positive edges are used to build the base graph, while 1,241,995 positives remain available before client ownership is applied. Training positives are then selected from the positive-edge pools available within each client. For evaluation, we use one validation positive for each client-task pair, giving six validation positives in total, and 1,000 test positives per task, giving 2,000 test positives in total. Negative candidates are generated separately and are not included in these positive-edge counts. The reported LP runs use fixed training configurations and evaluate the model after each stage. The validation positives are not used for early stopping or for selecting the best checkpoint within a run.

Structural features. For LC, we use global in-degree and out-degree computed before client ownership is assigned. For LP, we use global in-degree computed from the full positive graph before separating base and held-out positive edges. After client partitioning, each client keeps the degree

values of its owned nodes. Thus, message passing and prediction use only client-local nodes and edges, but the degree features still contain information from the full graph. In LP, these degree values also include held-out positive edges. The reported LP results therefore use global degree information and should not be interpreted as a setting where all structural features exclude held-out positives.

A.2 Task and Domain Definitions

Task- and Class-IL. NC and LC labels are grouped by Label-Preserving Task (LPT) balancing using training support only. Given C target labels and T tasks, each task contains

$$q = \left\lfloor \frac{C}{T} \right\rfloor \tag{3}$$

labels. LPT assigns higher-support labels first to the task with the smallest accumulated support, subject to this capacity. The resulting mapping is fixed across allocation levels, task orders, and learning methods. No remainder labels arise for the reported NC grouping or the six target classes used by S_4/ S_5. Table 7 gives the exact target groups. Task-IL and Class-IL use the same groups.

Table 7: Exact Task-/Class-IL target groups. Task IDs are global and do not change with client task order.

Family	Task ID	Original classes or ratings
NC	0	28, 29, 17, 35, 12
	1	16, 18, 21, 32, 15
	2	24, 37, 22, 0, 7
	3	34, 9, 23, 33, 11
	4	10, 36, 20, 3, 1
	5	30, 26, 39, 38, 14
	6	8, 4, 27, 13, 25
	7	2, 5, 19, 31, 6
LC	0	$\{3\}$ and $\{4\}$
	1	$\{-10\}$ and $\{-9, \ldots, -1\}$
	2	$\{6, \ldots, 10\}$ and $\{5\}$

For LC, ratings -10, $-9, \ldots, -1$, 3, 4, 5, and $6, \ldots, 10$ define the six Task-/Class-IL target classes represented in the table. Ratings 0, 1, and 2 form a seventh class that is not a Task-/Class-IL prediction target but remains part of the seven-class S_6 output.

Domain-IL. Domain-IL keeps the prediction space fixed while changing the distribution of examples across tasks. The source of this shift is prediction-family specific: species for NC, interaction structure for LC, and source category for LP.

NC: species domains. OGBN-Proteins treats each of its eight species as one domain while retaining the same 112-dimensional multi-label output across all tasks. Hence, the continual change is induced by species rather than by a changing label space.

LC: structural domains. Bitcoin-OTC defines domains from endpoint connectivity rather than interaction ratings. For an interaction (u, v), let $d(\cdot)$ denote total degree in the full directed interaction graph and define

$$s(u, v) = \ln(1 + d(u)) + \ln(1 + d(v)). \tag{4}$$

The training interactions are divided into four domains using the quartiles of this score:

$$b_j = Q_{j/4}\left(\{s(u, v) : (u, v) \in \mathcal{E}_{\text{train}}\}\right), \qquad j \in \{1, 2, 3\}, \tag{5}$$

where Q_p denotes the p-quantile. The four domains are $s \leq b_1$, $b_1 < s \leq b_2$, $b_2 < s \leq b_3$, and $s > b_3$. All domains share the same seven-class rating space, and the boundaries are fixed before client ownership is constructed.

LP: source domains. GEMSEC Facebook groups links by their source category while retaining the same binary link-prediction objective. The largest source category, *Artist*, defines one domain, and

the remaining seven categories define the other. This grouping is fixed before client ownership is constructed.

A.3 Prediction and Inference Rules

A prediction example is valid only when the information required to evaluate it is available within one client. NC requires ownership of the target node, LC and LP require ownership of both endpoints. These locality rules apply to training, validation, and test examples. Task-IL provides the queried global task ID and restricts prediction to that task's labels. Class-IL hides the task ID and predicts over the union of labels from tasks already encountered by the evaluating client. This is a client-local seen-class rule: a label does not become available because another client has encountered its task. Domain-IL hides the domain ID and retains a fixed output space.

Node classification. Client k performs message passing on its induced graph

$$\mathcal{G}_k = \mathcal{G}[\mathcal{V}_k]. \tag{6}$$

Validation and test nodes remain available as unlabeled graph context, but their target labels are not model inputs.

Link classification. Each original directed Bitcoin-OTC interaction defines one LC example when both endpoints have the same owner. The original direction is retained and no reverse interaction is introduced for message passing, an original reciprocal interaction remains a separate example. The target interaction may remain in the local graph at validation and test time, but its target label and target-derived attributes are hidden. LC is therefore transductive with respect to interaction structure. Only selected training interactions provide labeled supervision, other retained interactions may provide unlabeled context.

Link prediction. Validation and test positives, together with their negative candidates, are fixed across allocation levels, task orders, and learning methods. Each client-task pair uses a fixed pool of 1,000 validation negatives and another 1,000 test negatives. Candidate endpoints must belong to the same client, self-pairs and pairs that are known positives in either direction are excluded. Training uses one fixed negative per positive. Each evaluated positive is ranked against the corresponding negative pool. Let q^-_{50} denote its 50th-largest negative score. The positive counts as a hit only when its score is strictly greater than q^-_{50}, ties are counted as misses. LP distinguishes the fixed local base topology from task-specific training context. For client k and queried task t, evaluation at stage s uses

$$\mathcal{G}_{\text{eval}}(k, s, t) = \big(\mathcal{V}_k,\, E_{\text{base}}[\mathcal{V}_k] \cup E^{k,t}_{\text{train}},\, \mathbf{X}_k\big). \tag{7}$$

Here $E^{k,t}_{\text{train}}$ includes the reverse arcs used for message passing. Validation and test positives are never added to this graph. For a fixed benchmark condition and queried task, the evaluation context is unchanged across stages: past tasks are evaluated with their own task-specific training context rather than an accumulated union across tasks. Future-task training positives are not supplied to the current training view. Strict-local access in this benchmark is an evaluation constraint rather than a formal privacy guarantee, it does not include cross-client neighborhood retrieval or subgraph exchange.

B Construction Algorithms and Validation

This section specifies the changes made between benchmark conditions and validates their realized allocation, retained graph context, and available supervision.

B.1 Conditions and Information Use

The benchmark crosses $\alpha \in \{0.1, 1, 10, 100\}$ with synchronized (p_{S}) and unsynchronized (p_{U}) task orders. The four endpoint anchors are

$$\begin{aligned} \text{Base} &= (100, p_{\text{S}}), \quad \text{Alloc.} = (0.1, p_{\text{S}}), \\ \text{Order} &= (100, p_{\text{U}}), \quad \text{Joint} = (0.1, p_{\text{U}}). \end{aligned} \tag{8}$$

These anchors define comparisons, not an easy-to-hard ordering. Table 8 specifies their fixed and changed components. The same α need not produce the same allocation concentration across prediction families.

Table 8: Components held fixed and changed in each comparison. Local data include training and evaluation examples and LP negative candidates.

Comparison	**Held fixed**	**Changed**
Methods	Tasks, ownership, local data, order, participation, round budget, and paired initialization where architectures permit.	Learning method, including its prescribed federated rule.
Task order	Tasks, ownership, local data, participation, and training budget.	Client task permutations.
Allocation: NC/LC	Global tasks and data splits, task order, participation, and training budget.	Requested allocation and ownership, local graphs and retained examples may change.
Allocation: LP	Ownership, base graph, local positive pools, evaluation data, training-positive totals, task order, and participation.	Training-positive allocation, selected edges, and the resulting training context.

NC and LC. Ownership uses graph structure and training class or domain labels. Held-out example locations, split membership, and predefined task membership may be used to ensure client-task coverage. Individual held-out class labels within those task groups are excluded, as are NC node features and target-derived LC attributes. Because task membership can identify a label group, this is task-aware construction rather than fully label-blind partitioning.

LP. Ownership proposals are formed from the base graph and the construction seed. Acceptance additionally uses remaining-positive endpoints and task membership to check local support before the held-out split is formed. Thus proposal generation is topology-based, but acceptance is support-dependent. Neither NC/LC nor LP ownership uses model predictions or validation scores, LP evaluation negatives are formed only after ownership is fixed. These construction permissions are distinct from learner access.

B.2 NC AND LC CONSTRUCTION

NC and LC fix task definitions and global data splits before client ownership. For each training group z, client proportions are drawn as

$$\mathbf{p}_z \sim \mathrm{Dir}_K(\alpha \mathbf{1}), \tag{9}$$

where groups are classes in Task-/Class-IL and domains in Domain-IL. The sampled proportions are converted to integer client quotas while preserving group totals and balanced client training loads. For S_4/ S_5, these quotas apply to the fixed selected-supervision budget rather than to the entire eligible training pool.Ownership is guided by graph structure: S_1/ S_2 and LC use Louvain with METIS refinement (Blondel et al., 2008; Karypis & Kumar, 1998), while S_3 uses METIS directly. LC additionally requires both endpoints of a supervised interaction to share one owner, so interaction selection and endpoint ownership are determined jointly. Each client-task pair must retain at least five training, one validation, and one test example. Conditions that do not satisfy the prescribed quotas, support, or locality requirements are excluded rather than approximated.

B.3 LP CONSTRUCTION

LP reverses the construction order: ownership is fixed before training-positive allocation. A fixed rule first reserves approximately 10% of positive edges as the base topology. Ownership is formed from this base graph and retained only when every client-task pair has sufficient remaining positive support. Ownership, local base graphs, validation/test positives, and negative candidates then remain fixed across all allocation levels. Allocation affects only the training positives selected from these

fixed client-local pools. Let $C_{k,t}$ denote the available training positives for client k and task t. The total training budget for task t is

$$B_t = \min\left(\sum_k C_{k,t}, \max\left(5K, \left\lfloor 0.001 \sum_k C_{k,t} \right\rfloor\right)\right). \tag{10}$$

This total is fixed across α, while the client allocation follows an α-dependent Dirichlet draw subject to local capacity and a minimum of five positives per client. For client k and task t, the resulting graph context is

$$E^{k,t}_{\text{context}} = E_{\text{base}}[\mathcal{V}_k] \cup E^{k,t}_{\text{train}}. \tag{11}$$

Thus LP allocation changes the selected training edges and their task-specific context, but not ownership, base topology, or evaluation data.

B.4 Task Orders and Participation

Every client encounters every task once. Synchronized clients share one task sequence, unsynchronized clients use different predefined permutations. NC permutations are restricted within two four-task blocks, S_4/ S_5 permutes three tasks, S_6 permutes four, and the three LP clients use the two opposite orders. Within each paired comparison, ownership, local data, evaluation candidates, and participation are unchanged.

Each stage contains 50 rounds. NC and LC select five of ten clients per round, whereas LP selects two of three. Every client participates at least once per stage, and all clients advance together at the stage boundary. Unsynchronized order therefore changes task composition within a round, not the synchronization of federated optimization. In NC, unsynchronized rounds contain 3.21 ± 0.15 distinct tasks on average versus exactly one under synchronized order, this verifies the intended task-order mismatch without attributing a mechanism to its learning effect.

B.5 Construction Validation

Table 9: Constructor fidelity under identical requested client-by-group counts. Values are mean±sample SD over three seeds. S_3 provides construction-only evidence.

Scenario	α	Allocation error (%) ↓		Edge retention (%) ↑		Example retention (%) ↑	
		Node	Topo.	Node	Topo.	Node	Topo.
S_2	0.1	0.00 ± 0.00	0.00 ± 0.00	22.85 ± 0.52	54.11 ± 0.59	100.00 ± 0.00	100.00 ± 0.00
	1	0.00 ± 0.00	0.00 ± 0.00	17.43 ± 0.41	58.95 ± 0.99	100.00 ± 0.00	100.00 ± 0.00
	10	0.00 ± 0.00	0.00 ± 0.00	16.23 ± 0.02	60.25 ± 1.50	100.00 ± 0.00	100.00 ± 0.00
	100	0.00 ± 0.00	0.00 ± 0.00	16.10 ± 0.02	59.61 ± 0.83	100.00 ± 0.00	100.00 ± 0.00
S_3	0.1	0.00 ± 0.00	0.00 ± 0.00	13.56 ± 0.27	54.32 ± 1.94	100.00 ± 0.00	100.00 ± 0.00
	1	0.00 ± 0.00	0.00 ± 0.00	10.96 ± 0.04	44.06 ± 1.29	100.00 ± 0.00	100.00 ± 0.00
	10	0.00 ± 0.00	0.00 ± 0.00	10.24 ± 0.02	40.72 ± 0.71	100.00 ± 0.00	100.00 ± 0.00
	100	0.00 ± 0.00	0.00 ± 0.00	10.16 ± 0.00	40.40 ± 0.38	100.00 ± 0.00	100.00 ± 0.00
S_5	0.1	16.08 ± 2.45	0.00 ± 0.00	10.09 ± 0.18	36.51 ± 0.32	19.68 ± 0.91	48.58 ± 0.34
	1	7.75 ± 2.44	0.00 ± 0.00	10.09 ± 0.18	35.20 ± 3.59	22.47 ± 2.03	48.29 ± 2.40
	10	3.56 ± 1.26	0.00 ± 0.00	10.09 ± 0.18	37.32 ± 2.56	23.87 ± 1.57	49.02 ± 1.72
	100	2.51 ± 0.78	0.00 ± 0.00	10.09 ± 0.18	35.38 ± 1.57	24.22 ± 1.43	49.31 ± 1.35

Allocation fidelity and graph retention. The node-wise reference ignores graph connectivity, whereas topology-aware construction uses graph structure under the same requested client-by-group allocation. LP is excluded from this comparison because its ownership does not change with α.

For requested and realized training counts $q_{k,z}$ and $n_{k,z}$, allocation error is

$$E_{\text{quota}} = \frac{100}{2N} \sum_{k,z} |n_{k,z} - q_{k,z}|, \qquad N = \sum_{k,z} q_{k,z}, \tag{12}$$

where zero denotes exact matching. Edge retention is

$$R_E = 100 \, \frac{\sum_k |E[\mathcal{V}_k]|}{|E|}. \tag{13}$$

Example retention measures prediction examples that remain valid after ownership. For S_4/ S_5, the denominator is the prescribed training budget together with all validation and test examples:

$$R_{\text{ex}}^{\text{LC}} = 100 \, \frac{N_{\text{train}}^{\text{selected}} + N_{\text{val}}^{\text{local}} + N_{\text{test}}^{\text{local}}}{398 + 970 + 1010}. \tag{14}$$

The eligible pool of 8,002 training interactions is therefore not the denominator of this retention measure.

Realized allocation. Allocation concentration is measured by the Jensen-Shannon divergence between each training group's client distribution and the uniform distribution. Higher values indicate more concentrated allocation, this quantity measures realized heterogeneity rather than quota error. Table 10 reports this diagnostic across allocation levels for Scenarios S_1/ S_2 (NC), S_4/ S_5 (LC) and S_7 (LP).

Table 9 shows that both NC constructions match the requested allocation, while topology-aware ownership retains substantially more graph context. For LC, topology-aware construction also retains more supervision and matches the requested quotas more closely. The LC comparison therefore changes both local topology and realized supervision and should not be interpreted as an edge-only intervention.

Table 10: Realized allocation concentration, measured by Jensen-Shannon divergence from uniform client allocation (nats). Values are mean±sample SD over seeds 0-2, higher values indicate greater concentration. LP uses three clients.

Scenario	$\boldsymbol{\alpha = 0.1}$	$\boldsymbol{\alpha = 1}$	$\boldsymbol{\alpha = 10}$	$\boldsymbol{\alpha = 100}$
S_1/ S_2	0.3630 ± 0.0033	0.0954 ± 0.0060	0.0117 ± 0.0006	0.00113 ± 0.00014
S_4/ S_5	0.1908 ± 0.0113	0.0827 ± 0.0061	0.0131 ± 0.0016	0.00122 ± 0.00018
S_7	0.1637 ± 0.0471	0.0960 ± 0.0271	0.0086 ± 0.0031	0.00139 ± 0.00048

Across all three prediction families, realized concentration decreases as α increases. For LP, ownership and the base graph are fixed, so retained base topology and the eligible local positive population do not change with α, only the selected training positives do.

Validity and prediction populations. The reported conditions satisfy the prescribed ownership, allocation, support, locality, and paired-comparison checks. These checks apply to retained benchmark conditions and are not an estimate of the success rate over all attempted constructions.

NC retains the same global prediction population across allocation levels, and LP fixes its validation/test positives and candidates. LC instead has allocation-dependent evaluation populations: between $\alpha = 0.1$ and 100, validation-set Jaccard overlap ranges from 0.53 to 0.79 and test overlap from 0.58 to 0.76 across the reported seeds. LC allocation effects therefore describe a change in the overall client-local evaluation condition, not only a reassignment of a fixed interaction set.

C Training, Evaluation, and Resource Accounting

This section specifies the training conditions, evaluation metrics, and method configurations used in the reported comparisons. Benchmark conditions are matched across methods, but architectures, auxiliary computation, and native federated strategies are not forced to be identical.

C.1 Training Protocol

Within a benchmark condition, methods share the same task definitions, ownership, local graphs, prediction examples, task order, participation schedule, and communication-round budget. Construction and model randomness are separated but paired by seed index wherever architectures permit. NC and LC use ten clients with five participants per round. S_7 uses three clients with two participants per round to retain sufficient within-client positive support while preserving multi-client aggregation. Each continual stage contains 50 communication rounds.

One local epoch corresponds to one full-batch optimizer update for the standard training rule, FedPUB uses two updates per local epoch. Method-specific auxiliary optimization is additional to these local updates. Evaluation is performed after each stage on all tasks encountered by each client using the model prescribed by the corresponding learning method. NC and LC use three paired seeds, S_7 uses five as it provides a more reliable estimate of performance variability across runs. The same scenario-level seed set is reused across methods and benchmark conditions.

Table 11: Default experimental configurations. Each stage uses 50 communication rounds and one local epoch per selected client. Unless stated otherwise, the default model is a 3-layer GCN (Kipf & Welling, 2016) with hidden dimension 256, trained with Adam (Kingma & Ba, 2014) at learning rate 0.01 and zero weight decay.

Scenario	Clients	Tasks	Clients / round	Seeds	Metric
S_1/ S_2	10	8	5 / 10	3	Accuracy
S_4/ S_5	10	3	5 / 10	3	Accuracy
S_3	10	8	5 / 10	3	ROC-AUC
S_6	10	4	5 / 10	3	Accuracy
S_7	3	2	2 / 3	5	Hits@50

C.2 Metrics and Statistical Aggregation

For method m and seed r, let $M^{m,r}_{k,s,t}$ denote the primary metric for client k on global task t after stage s. Client k first encounters task t at

$$\tau_{k,t} = \pi_k^{-1}(t). \tag{15}$$

Accuracy is used for NC Task-/Class-IL and all LC scenarios, ROC-AUC for S_3, and Hits@50 for S_7.

Final performance and forgetting. NC and LC average valid client-task pairs uniformly. LP final performance is weighted by the number of positive evaluation examples, while forgetting remains uniform across client-task pairs. Let $\mathcal{I}$ denote the valid pairs and $n^+_{k,t}$ the LP positive count:

$$w^P_{k,t} = \begin{cases} \dfrac{1}{|\mathcal{I}|}, & \text{NC or LC,} \\ \dfrac{n^+_{k,t}}{\sum_{(i,j)\in\mathcal{I}} n^+_{i,j}}, & \text{LP,} \end{cases} \qquad w^F_{k,t} = \frac{1}{|\mathcal{I}|}. \tag{16}$$

C.3 Method Coverage and Fixed Configurations

Methods without a defining federated optimization rule use FedAvg in the main comparison. Methods with a prescribed server-side optimization or aggregation rule retain that rule. The comparison therefore evaluates complete methods under matched benchmark conditions rather than forcing all methods into a common federated optimizer. Table 12 summarizes the reported scenario coverage and execution. Core Class-IL and S_7 results use the full allocation-order grid. Matched Task-IL and S_6 comparisons use the four endpoint anchors. S_3 provides scenario and construction coverage rather than a learner-level comparison.

For each method and evaluation setting, we use one fixed set of hyperparameters across allocation levels, task orders, and random seeds. These settings are chosen based on the original papers,

Table 12: Method coverage and designated execution in the reported comparisons. Coverage denotes the evaluated adaptation, not every scenario a method could support in principle.

Method	Reported scenarios	Execution
Bare	S_2, S_5, S_7	FedAvg
CaT	S_2, S_5	FedAvg
DSLR	S_2	FedAvg
FedDC	S_1/ S_2	FedDC
FedFST	S_1/ S_2	FedFST
FedGTA	S_1/ S_2	FedGTA
FedPUB	S_2	FedPUB
GEM	S_1/ S_2, S_4/ S_5/ S_6	FedAvg
MOTION	S_1/ S_2	MOTION
POWER	S_1/ S_2	POWER
SSM	S_2, S_5	FedAvg
TWP	S_1/ S_2, S_4/ S_5/ S_6	FedAvg
EWC	S_7	FedAvg
LwF	S_7	FedAvg
GraphKeeper	S_7	FedAvg

adaptations required for our benchmark, or available validation results. We do not apply a unified hyperparameter search to all methods. The reported results should therefore be interpreted as comparisons under these fixed settings, rather than as the best performance obtainable by separately tuning each method for every evaluation condition.

Table 13: Method-specific configurations. Unless specified otherwise, methods use the default 3-layer GCN and Adam settings in Table 11. Auxiliary steps are additional to ordinary federated rounds.

Method	Model / optimization	Method-specific settings
Bare	Default	No continual-learning component.
CaT	Condensation LR 10^{-3}, 32 steps	2 synthetic items/class.
DSLR	99 structure-learning epochs	$\beta = 0.1$, radius 0.2, replay 5%.
FedDC	Default	$\alpha_{\mathrm{DC}} = 0.1$, client drift enabled.
FedFST	2-layer 1-head GAT (Veličković et al., 2017), hidden 64, dropout 0.5	Client/gen./distill. LR $= 0.01/0.005/0.002$, $\lambda_{\mathrm{KL}} = 100$, $\lambda_{\mathrm{low}} = 1$, 2-hop smoothing.
FedGTA	5 propagation steps	$\alpha_{\mathrm{GTA}} = 0.5$, temperature 20, moment order 6, similarity threshold 0.5.
FedPUB	2 optimizer steps/local epoch	$\tau = 3$, $\lambda_1 = \lambda_2 = 10^{-3}$, proxy seed 0.
GEM	Default	Memory 100 queries/client, margin 0.5, projection $\epsilon = 10^{-3}$.
MOTION	3 expert selections	Buffer 200, replay weight 2.
POWER	Server LR 10^{-3}, 300 reconstruction steps, 3 server epochs	$\alpha_{\mathrm{POWER}} = 0.5$, $\beta = 0.01$, 10 samples/class.
SSM	Sampling budgets $[10, 25]$	Replay weight 1, uniform sampling (NC), degree-based sampling (LC).
TWP	Default	$\lambda_l = \lambda_t = 10^4$, $\beta = 0.01$.
EWC	Default	Regularization weight 1.0.
LwF	Default	Regularization weight 1.0, temperature 2.0.
GraphKeeper	LoRA rank 16	Temperature 0.2, feature/edge drop $0.3/0.2$.

Task-IL and Class-IL use the same task groups and prediction examples but not identical training semantics. Task-IL restricts its loss to the current task's classes, whereas Class-IL includes classes encountered so far. Their matched comparison therefore measures the combined incremental-learning semantics and should not be interpreted as an inference-only task-identity intervention. Native methods also retain their defining model or server-side components. Accordingly, the main comparison matches benchmark conditions but does not equalize architecture capacity, auxiliary computation, or communicated statistics across methods.

C.4 Resource Accounting

The common 50-round budget equalizes communication rounds rather than total computation, as method-specific operations such as condensation, structure learning, propagation, and server-side optimization incur additional cost; their prescribed budgets are reported in Table 13. CaT, DSLR, SSM, MOTION, and POWER use a 16 MiB replay or auxiliary-memory cap, while GEM stores 100 queries per client. We also record elapsed time, peak allocated GPU memory, and logical communication volume where available, but do not use them for hardware-controlled efficiency comparisons because runs span different experimental campaigns and hardware settings.

D Complete Results

This section reports the absolute numerical results underlying the main comparisons. Final performance and average forgetting (AF) are reported separately for the core allocation-order grids. NC and LC use three paired seeds, and S_7 uses five.

D.1 S_2 Full-Grid Results

Tables 14 and 15 report the complete available S_2 grid.

Table 14: Complete S_2 final accuracy on OGBN-Arxiv (%). Values are mean±sample SD over three paired seeds. FedFST is reported only under synchronized task order.

Order	Method	$\alpha = 0.1 \uparrow$	$\alpha = 1 \uparrow$	$\alpha = 10 \uparrow$	$\alpha = 100 \uparrow$
Sync. task order	Bare	6.81 ± 2.60	10.15 ± 0.13	10.10 ± 0.30	10.49 ± 0.14
	CaT	4.40 ± 2.11	17.62 ± 1.55	17.91 ± 1.67	18.98 ± 0.93
	DSLR	5.68 ± 0.81	17.55 ± 1.88	26.26 ± 3.12	25.05 ± 2.00
	FedDC	3.61 ± 1.25	4.31 ± 0.91	5.71 ± 0.56	4.70 ± 1.08
	FedFST	7.05 ± 2.09	13.03 ± 1.43	17.12 ± 2.27	15.29 ± 1.46
	FedGTA	5.88 ± 0.79	9.93 ± 0.16	10.04 ± 0.42	10.40 ± 0.22
	FedPUB	3.15 ± 2.66	1.41 ± 0.91	1.33 ± 0.59	1.57 ± 0.52
	GEM	11.76 ± 5.11	25.65 ± 0.29	28.13 ± 5.21	33.54 ± 4.04
	MOTION	$\mathbf{25.30 \pm 2.58}$	$\mathbf{45.13 \pm 2.65}$	$\mathbf{48.01 \pm 1.67}$	$\mathbf{49.71 \pm 1.65}$
	POWER	7.56 ± 0.47	14.27 ± 2.43	15.18 ± 1.02	15.17 ± 2.91
	SSM	7.53 ± 5.20	9.29 ± 1.38	12.50 ± 1.95	10.17 ± 2.01
	TWP	4.38 ± 0.33	3.08 ± 2.14	3.90 ± 2.58	4.23 ± 2.99
Unsync. task order	Bare	5.58 ± 2.33	13.45 ± 3.79	14.90 ± 5.09	17.28 ± 4.47
	CaT	7.62 ± 0.95	17.37 ± 2.15	19.69 ± 2.05	20.11 ± 3.79
	DSLR	15.00 ± 6.58	34.32 ± 5.13	36.35 ± 1.65	38.18 ± 1.72
	FedDC	2.76 ± 0.47	7.47 ± 1.63	9.09 ± 2.33	9.65 ± 3.75
	FedGTA	6.28 ± 0.83	15.22 ± 1.51	15.49 ± 2.42	16.75 ± 0.90
	FedPUB	2.63 ± 2.22	1.66 ± 0.43	2.22 ± 0.77	1.52 ± 0.42
	GEM	14.71 ± 4.00	36.81 ± 1.98	39.21 ± 0.87	37.24 ± 1.57
	MOTION	$\mathbf{28.77 \pm 2.90}$	$\mathbf{46.43 \pm 2.55}$	$\mathbf{50.81 \pm 1.59}$	$\mathbf{50.40 \pm 0.66}$
	POWER	10.48 ± 0.45	11.55 ± 6.25	9.57 ± 1.96	12.66 ± 6.05
	SSM	6.04 ± 0.54	19.16 ± 2.28	20.02 ± 2.79	19.32 ± 4.22
	TWP	2.75 ± 1.70	4.16 ± 5.36	4.86 ± 3.78	4.54 ± 3.83

D.2 S_5 Full-Grid Results

Tables 16 and 17 report the complete S_5 grid.

Table 15: Complete S_2 average forgetting on OGBN-Arxiv (percentage points). Values are mean±sample SD over three paired seeds. FedFST is reported only under synchronized task order.

Order	Method	$\alpha = 0.1 \downarrow$	$\alpha = 1 \downarrow$	$\alpha = 10 \downarrow$	$\alpha = 100 \downarrow$
Sync. task order	Bare	63.43 ± 1.03	72.52 ± 1.69	74.26 ± 0.83	75.55 ± 0.61
	CaT	$\mathbf{23.64 \pm 5.80}$	43.13 ± 3.05	43.69 ± 1.40	42.13 ± 0.31
	DSLR	51.45 ± 7.28	59.59 ± 1.59	53.17 ± 3.24	53.96 ± 2.90
	FedDC	47.51 ± 2.03	52.98 ± 2.65	55.98 ± 2.01	55.34 ± 1.16
	FedFST	36.58 ± 1.79	32.33 ± 4.80	28.71 ± 5.05	36.92 ± 3.17
	FedGTA	60.26 ± 4.34	74.37 ± 0.83	74.06 ± 1.57	75.27 ± 1.13
	FedPUB	27.84 ± 5.92	27.69 ± 4.91	31.75 ± 6.32	36.01 ± 3.94
	GEM	51.05 ± 9.23	50.43 ± 2.50	47.21 ± 2.69	43.23 ± 4.58
	MOTION	28.00 ± 7.90	$\mathbf{10.73 \pm 2.73}$	$\mathbf{11.76 \pm 1.12}$	$\mathbf{11.01 \pm 0.84}$
	POWER	28.15 ± 2.83	32.81 ± 1.19	29.53 ± 2.16	33.56 ± 1.60
	SSM	49.14 ± 6.81	48.12 ± 3.33	50.25 ± 3.29	54.93 ± 6.02
	TWP	45.98 ± 6.23	38.46 ± 2.81	32.49 ± 1.78	33.56 ± 1.77
Unsync. task order	Bare	37.48 ± 2.54	47.05 ± 7.76	49.67 ± 8.22	48.64 ± 7.43
	CaT	19.53 ± 1.66	28.52 ± 3.97	27.07 ± 1.17	25.46 ± 1.27
	DSLR	31.88 ± 3.93	32.68 ± 4.40	33.16 ± 2.96	32.61 ± 1.73
	FedDC	31.18 ± 0.98	38.58 ± 1.36	40.09 ± 3.44	40.95 ± 1.38
	FedGTA	40.95 ± 2.89	54.83 ± 3.44	57.26 ± 0.84	57.10 ± 2.82
	FedPUB	21.10 ± 4.16	25.48 ± 0.36	25.50 ± 0.91	24.63 ± 5.45
	GEM	35.56 ± 1.80	29.96 ± 2.11	30.90 ± 2.51	31.39 ± 1.48
	MOTION	$\mathbf{19.52 \pm 1.59}$	$\mathbf{11.91 \pm 2.96}$	$\mathbf{10.39 \pm 0.92}$	$\mathbf{10.83 \pm 1.85}$
	POWER	23.76 ± 0.95	29.48 ± 4.58	29.54 ± 3.70	32.46 ± 4.15
	SSM	35.39 ± 2.39	38.32 ± 0.92	41.72 ± 0.85	42.83 ± 2.14
	TWP	19.54 ± 2.85	20.06 ± 6.18	20.80 ± 6.14	20.81 ± 9.36

Table 16: Complete S_5 final accuracy on Bitcoin-OTC (%). Values are mean±sample SD over three paired seeds.

Order	Method	$\alpha = 0.1 \uparrow$	$\alpha = 1 \uparrow$	$\alpha = 10 \uparrow$	$\alpha = 100 \uparrow$
Sync. task order	Bare	18.50 ± 3.57	17.03 ± 1.83	13.93 ± 0.61	15.46 ± 2.54
	CaT	$\mathbf{21.83 \pm 1.57}$	18.24 ± 0.52	18.25 ± 7.65	17.28 ± 1.77
	SSM	17.50 ± 1.07	21.03 ± 5.28	$\mathbf{19.31 \pm 0.69}$	15.80 ± 7.25
	TWP	18.90 ± 4.65	19.97 ± 0.07	18.09 ± 4.24	18.53 ± 2.98
	GEM	16.76 ± 2.25	$\mathbf{22.63 \pm 3.46}$	17.03 ± 1.61	$\mathbf{21.25 \pm 1.23}$
Unsync. task order	Bare	20.45 ± 0.49	21.37 ± 4.12	19.71 ± 2.37	19.95 ± 1.74
	CaT	20.54 ± 6.02	19.67 ± 5.15	17.48 ± 4.46	18.01 ± 8.11
	SSM	12.45 ± 5.38	16.98 ± 1.83	18.83 ± 2.77	18.18 ± 4.36
	TWP	$\mathbf{22.17 \pm 2.69}$	20.71 ± 1.91	24.07 ± 2.77	$\mathbf{24.48 \pm 2.73}$
	GEM	20.82 ± 3.98	$\mathbf{23.60 \pm 5.93}$	$\mathbf{24.19 \pm 4.07}$	20.55 ± 4.13

D.3 S_7 Full-Grid Results

Tables 18 and 19 report the complete S_7 grid.

D.4 Matched Task- and Class-IL Results

The matched Task-/Class-IL comparison uses the four endpoint anchors with the same task groups, ownership, prediction examples, order, participation, and training budget. The two incremental settings retain their respective training and output-space rules, so this comparison is not an inference-only task-identity intervention. Tables 20 and 21 report the matched Task-/Class-IL results for NC and LC, respectively.

Table 17: Complete S_5 average forgetting on Bitcoin-OTC (percentage points). Values are mean±sample SD over three paired seeds.

Order	Method	$\alpha = 0.1 \downarrow$	$\alpha = 1 \downarrow$	$\alpha = 10 \downarrow$	$\alpha = 100 \downarrow$
Sync. task order	Bare	41.54 ± 5.42	41.59 ± 3.19	38.38 ± 9.58	38.72 ± 1.51
	CaT	$\mathbf{12.27 \pm 8.43}$	25.03 ± 4.03	$\mathbf{17.19 \pm 6.13}$	22.21 ± 4.07
	SSM	19.41 ± 4.63	$\mathbf{13.88 \pm 7.46}$	23.47 ± 4.12	$\mathbf{21.63 \pm 2.85}$
	TWP	38.06 ± 11.12	38.92 ± 11.50	47.35 ± 2.56	38.80 ± 11.28
	GEM	37.18 ± 5.29	27.18 ± 1.69	27.75 ± 13.37	29.63 ± 8.42
Unsync. task order	Bare	24.61 ± 5.75	21.37 ± 4.32	18.21 ± 1.48	21.06 ± 5.07
	CaT	17.18 ± 0.94	17.40 ± 5.07	18.80 ± 2.35	20.31 ± 3.30
	SSM	23.71 ± 1.99	24.34 ± 5.71	21.45 ± 0.72	20.62 ± 3.18
	TWP	$\mathbf{17.03 \pm 1.64}$	17.90 ± 2.16	$\mathbf{15.94 \pm 1.86}$	$\mathbf{15.46 \pm 3.22}$
	GEM	20.01 ± 6.51	$\mathbf{14.87 \pm 3.66}$	17.29 ± 2.36	21.39 ± 7.85

Table 18: Complete S_7 Hits@50 on GEMSEC Facebook (%). Values are mean±sample SD over five paired seeds.

Order	Method	$\alpha = 0.1 \uparrow$	$\alpha = 1 \uparrow$	$\alpha = 10 \uparrow$	$\alpha = 100 \uparrow$
Sync. task order	Bare	28.49 ± 3.18	27.07 ± 11.88	43.24 ± 9.89	28.41 ± 15.24
	EWC	39.68 ± 15.48	32.71 ± 14.08	33.05 ± 14.37	33.91 ± 18.03
	LwF	32.54 ± 11.29	30.72 ± 8.26	33.98 ± 4.74	34.77 ± 15.57
	GraphKeeper	$\mathbf{40.57 \pm 13.29}$	$\mathbf{44.46 \pm 6.06}$	$\mathbf{45.90 \pm 6.66}$	$\mathbf{51.10 \pm 3.80}$
Unsync. task order	Bare	$\mathbf{44.66 \pm 12.22}$	40.01 ± 6.70	43.49 ± 14.07	39.34 ± 11.17
	EWC	37.94 ± 8.56	$\mathbf{43.74 \pm 7.69}$	$\mathbf{46.64 \pm 5.84}$	$\mathbf{45.62 \pm 11.32}$
	LwF	40.29 ± 11.58	39.25 ± 10.74	38.83 ± 13.21	39.45 ± 9.57
	GraphKeeper	38.05 ± 9.95	30.27 ± 12.06	39.52 ± 10.00	39.87 ± 10.69

Table 19: Complete S_7 average forgetting on GEMSEC Facebook (percentage points). Values are mean±sample SD over five paired seeds.

Order	Method	$\alpha = 0.1 \downarrow$	$\alpha = 1 \downarrow$	$\alpha = 10 \downarrow$	$\alpha = 100 \downarrow$
Sync. task order	Bare	9.71 ± 8.73	9.45 ± 7.83	$\mathbf{4.47 \pm 1.46}$	12.73 ± 14.37
	EWC	$\mathbf{4.99 \pm 4.33}$	8.31 ± 8.83	9.70 ± 12.98	11.61 ± 14.76
	LwF	7.27 ± 5.56	11.60 ± 5.87	8.50 ± 8.64	11.97 ± 14.40
	GraphKeeper	6.77 ± 6.36	$\mathbf{5.80 \pm 3.87}$	10.39 ± 9.52	$\mathbf{5.00 \pm 5.15}$
Unsync. task order	Bare	$\mathbf{1.61 \pm 2.22}$	2.66 ± 1.44	2.91 ± 2.02	2.41 ± 2.21
	EWC	2.17 ± 2.35	$\mathbf{1.87 \pm 1.87}$	$\mathbf{1.99 \pm 1.89}$	$\mathbf{1.40 \pm 1.19}$
	LwF	1.92 ± 2.10	6.04 ± 7.36	9.88 ± 11.83	6.08 ± 8.06
	GraphKeeper	13.29 ± 7.28	9.07 ± 3.49	4.87 ± 4.18	7.53 ± 7.05

Table 20: Matched NC Task-/Class-IL results on OGBN-Arxiv at the four anchors. Final accuracy is reported in percent and AF in percentage points. Values are mean±sample SD over three paired seeds.

Method	Base	Alloc.	Order	Joint
Final accuracy: Task-IL ↑				
GEM	82.02 ± 1.78	61.25 ± 4.42	82.72 ± 1.18	64.46 ± 2.71
TWP	56.49 ± 1.38	52.52 ± 6.25	56.77 ± 0.84	45.47 ± 0.89
FedGTA	75.83 ± 1.44	54.63 ± 5.75	79.76 ± 1.73	55.59 ± 2.90
FedDC	32.55 ± 12.66	18.16 ± 2.14	61.39 ± 3.52	20.62 ± 2.38
POWER	57.43 ± 4.14	45.93 ± 4.12	53.70 ± 1.42	44.48 ± 8.38
MOTION	$\mathbf{86.27 \pm 1.28}$	$\mathbf{71.82 \pm 3.82}$	$\mathbf{86.01 \pm 1.63}$	$\mathbf{73.62 \pm 2.03}$
Final accuracy: Class-IL ↑				
GEM	33.54 ± 4.04	11.76 ± 5.11	37.24 ± 1.57	14.71 ± 4.00
TWP	4.23 ± 2.99	4.38 ± 0.33	4.54 ± 3.83	2.75 ± 1.70
FedGTA	10.40 ± 0.22	5.88 ± 0.79	16.75 ± 0.90	6.28 ± 0.83
FedDC	4.70 ± 1.08	3.61 ± 1.25	9.65 ± 3.75	2.76 ± 0.47
POWER	15.17 ± 2.91	7.56 ± 0.47	12.66 ± 6.05	10.48 ± 0.45
MOTION	$\mathbf{49.71 \pm 1.65}$	$\mathbf{25.30 \pm 2.58}$	$\mathbf{50.40 \pm 0.66}$	$\mathbf{28.77 \pm 2.90}$
Average forgetting: Task-IL ↓				
GEM	5.15 ± 1.32	12.79 ± 3.01	4.91 ± 0.76	8.69 ± 0.50
TWP	$\mathbf{0.01 \pm 0.01}$	$\mathbf{2.38 \pm 1.79}$	$\mathbf{1.34 \pm 0.37}$	11.37 ± 2.53
FedGTA	11.53 ± 0.75	15.18 ± 4.34	6.85 ± 0.86	12.13 ± 1.41
FedDC	42.65 ± 10.78	38.80 ± 4.94	10.49 ± 3.80	37.40 ± 2.70
POWER	14.59 ± 4.03	15.05 ± 2.98	16.59 ± 1.81	15.94 ± 7.33
MOTION	1.76 ± 0.67	3.03 ± 1.35	1.80 ± 0.60	$\mathbf{4.31 \pm 0.71}$
Average forgetting: Class-IL ↓				
GEM	43.23 ± 4.58	51.05 ± 9.23	31.39 ± 1.48	35.56 ± 1.80
TWP	33.56 ± 1.77	45.98 ± 6.23	20.81 ± 9.36	19.54 ± 2.85
FedGTA	75.27 ± 1.13	60.26 ± 4.34	57.10 ± 2.82	40.95 ± 2.89
FedDC	55.34 ± 1.16	47.51 ± 2.03	40.95 ± 1.38	31.18 ± 0.98
POWER	33.56 ± 1.60	28.15 ± 2.83	32.46 ± 4.15	23.76 ± 0.95
MOTION	$\mathbf{11.01 \pm 0.84}$	$\mathbf{28.00 \pm 7.90}$	$\mathbf{10.83 \pm 1.85}$	$\mathbf{19.52 \pm 1.59}$

Table 21: Matched S_4/ S_5 results on Bitcoin-OTC at the four anchors. Final accuracy is reported in percent and AF in percentage points. Values are mean±sample SD over three paired seeds.

Method	Base	Alloc.	Order	Joint
Final accuracy: Task-IL ↑				
GEM	57.89 ± 11.01	54.61 ± 10.43	54.17 ± 8.40	52.57 ± 11.18
TWP	$\mathbf{65.34 \pm 1.62}$	$\mathbf{63.94 \pm 6.03}$	$\mathbf{65.34 \pm 1.62}$	$\mathbf{63.94 \pm 6.03}$
Final accuracy: Class-IL ↑				
GEM	$\mathbf{21.25 \pm 1.23}$	16.76 ± 2.25	20.55 ± 4.13	20.82 ± 3.98
TWP	18.53 ± 2.98	$\mathbf{18.90 \pm 4.65}$	$\mathbf{24.48 \pm 2.73}$	$\mathbf{22.17 \pm 2.69}$
Average forgetting: Task-IL ↓				
GEM	6.38 ± 3.96	10.01 ± 8.16	9.70 ± 4.25	12.38 ± 7.34
TWP	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{2.22 \pm 2.45}$
Average forgetting: Class-IL ↓				
GEM	$\mathbf{29.63 \pm 8.42}$	$\mathbf{37.18 \pm 5.29}$	21.39 ± 7.85	20.01 ± 6.51
TWP	38.80 ± 11.28	38.06 ± 11.12	$\mathbf{15.46 \pm 3.22}$	$\mathbf{17.03 \pm 1.64}$

D.5 Extended S_6 Results

S_6 is reported at the four endpoint anchors for GEM and TWP. S_3 provides scenario and construction coverage but has no learner-level result table in this release. The large uncertainty for GEM under Alloc. should be read directly from the reported seed variation. TWP's zero AF is an observed forgetting value and, by itself, does not establish successful acquisition or retention. Table 22 reports the four-anchor S_6 results for GEM and TWP.

Table 22: S_6 results on Bitcoin-OTC at the four anchors. Values are mean±sample SD over three paired seeds. Final accuracy is in percent and AF in percentage points.

Method	**Base**	**Alloc.**	**Order**	**Joint**
Final accuracy ↑				
GEM	66.28 ± 2.07	49.93 ± 29.18	66.14 ± 1.09	63.44 ± 0.85
TWP	$\mathbf{66.54 \pm 1.71}$	$\mathbf{67.32 \pm 2.16}$	$\mathbf{66.54 \pm 1.71}$	$\mathbf{67.32 \pm 2.16}$
Average forgetting ↓				
GEM	0.31 ± 0.50	13.19 ± 22.53	0.26 ± 0.46	3.27 ± 2.61
TWP	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$

E Sensitivity and Alternative Specifications

This section distinguishes descriptive method-order changes from comparisons that change the training rule or the construction. Existing numerical records are reused only under their stated protocol. Unavailable alternative-rule results remain pending, they do not support a conclusion about federation.

E.1 Descriptive Pairwise Method Comparisons

Tables 23–25 summarize each method pair over the eight allocation-order conditions using the range of final-performance gaps and the number of positive mean differences. At the level of condition means, most evaluated NC pairs retain the same ordering, whereas every evaluated LC and LP pair changes ordering somewhere in the grid. These summaries do not establish consistency across seeds.

Several ordering changes involve small gaps, so the reported mean differences need to be read with seed variation. These descriptive comparisons show how the relative performance of two methods can depend on the chosen condition.

E.2 Training-Rule Sensitivity

The main benchmark results use 50 communication rounds per stage and one local epoch (50×1). For the training-rule comparison, we instead use a common 10×5 protocol for Local-only, FedAvg, and FedProx, with 10 communication rounds per stage and five local updates per selected client. This gives FedProx multiple local optimization steps, allowing its proximal regularization to affect later updates. We fix the FedProx coefficient to $\mu = 0.01$ throughout.

LC results are reported at the four anchors spanning both allocation endpoints and task orders. For NC and LP, we evaluate $\alpha \in \{0.1, 100\}$ under synchronized order. This compares training rules under concentrated and more even allocations while holding task order fixed. NC and LC include CaT, GEM, SSM, and TWP with three paired seeds, LP includes EWC, LwF, and GraphKeeper with five paired seeds. Within each condition, the three training rules use the same client data, task order, participation schedule, initialization, model, and training budget. Tables 26–28 report these 10×5 results, no 50×1 scores are reused.

Effects of the training rule. Training-rule effects depend on the learner and allocation. In NC, FedProx improves mean final accuracy more consistently at Base than at Alloc., LC and LP show mixed responses. Accuracy gains can accompany higher AF, and some mean differences are small

Table 23: NC descriptive pairwise final-performance differences from the printed cell means (percentage points). Each difference is first method minus second method. Min/max are taken over eight condition means, not seeds. The final column counts positive printed-mean differences.

First	Second	Min. gap	Max. gap	Positive / 8
CaT	DSLR	−18.07	+0.07	1 / 8
	FedDC	+0.79	+14.28	8 / 8
	FedGTA	−1.48	+8.58	7 / 8
	FedPUB	+1.25	+18.59	8 / 8
	GEM	−19.52	−7.09	0 / 8
	MOTION	−31.12	−20.90	0 / 8
	POWER	−3.16	+10.12	6 / 8
	SSM	−3.13	+8.81	5 / 8
	TWP	+0.02	+15.57	8 / 8
DSLR	FedDC	+2.07	+28.53	8 / 8
	FedGTA	−0.20	+21.43	7 / 8
	FedPUB	+2.53	+36.66	8 / 8
	GEM	−8.49	+0.94	2 / 8
	MOTION	−27.58	−12.11	0 / 8
	POWER	−1.88	+26.78	7 / 8
	SSM	−1.85	+18.86	7 / 8
	TWP	+1.30	+33.64	8 / 8
FedDC	FedGTA	−7.75	−2.27	0 / 8
	FedPUB	+0.13	+8.13	8 / 8
	GEM	−30.12	−8.15	0 / 8
	MOTION	−45.01	−21.69	0 / 8
	POWER	−10.47	−0.48	0 / 8
	SSM	−11.69	−3.28	0 / 8
	TWP	−0.77	+5.11	7 / 8
FedGTA	FedPUB	+2.73	+15.23	8 / 8
	GEM	−23.72	−5.88	0 / 8
	MOTION	−39.31	−19.42	0 / 8
	POWER	−5.14	+5.92	3 / 8
	SSM	−4.53	+0.64	3 / 8
	TWP	+1.50	+12.21	8 / 8
FedPUB	GEM	−36.99	−8.61	0 / 8
	MOTION	−48.88	−22.15	0 / 8
	POWER	−13.85	−4.41	0 / 8
	SSM	−17.80	−3.41	0 / 8
	TWP	−3.02	−0.12	0 / 8
GEM	MOTION	−19.88	−9.62	0 / 8
	POWER	+4.20	+29.64	8 / 8
	SSM	+4.23	+23.37	8 / 8
	TWP	+7.38	+34.35	8 / 8
MOTION	POWER	+17.74	+41.24	8 / 8
	SSM	+17.77	+39.54	8 / 8
	TWP	+20.92	+45.95	8 / 8
POWER	SSM	−10.45	+5.00	5 / 8
	TWP	+3.18	+11.28	8 / 8
SSM	TWP	+3.15	+15.16	8 / 8

Table 24: LC descriptive pairwise final-performance differences from the printed cell means (percentage points). Each difference is first method minus second method. Min/max are taken over eight condition means, not seeds. The final column counts positive printed-mean differences.

First	Second	Min. gap	Max. gap	Positive / 8
CaT	GEM	−6.71	+5.07	2 / 8
	SSM	−2.79	+8.09	4 / 8
	TWP	−6.59	+2.93	2 / 8
GEM	SSM	−2.28	+8.37	6 / 8
	TWP	−3.93	+2.89	4 / 8
SSM	TWP	−9.72	+1.22	2 / 8

Table 25: LP descriptive pairwise final-performance differences from the printed cell means (percentage points). Each difference is first method minus second method. Min/max are taken over eight condition means, not seeds. The final column counts positive printed-mean differences.

First	Second	Min. gap	Max. gap	Positive / 8
Bare	EWC	−11.19	+10.19	2 / 8
	LwF	−6.36	+9.26	4 / 8
	GraphKeeper	−22.69	+9.74	3 / 8
EWC	LwF	−2.35	+7.81	5 / 8
	GraphKeeper	−17.19	+13.47	3 / 8
LwF	GraphKeeper	−16.33	+8.98	2 / 8

Table 26: Training-rule sensitivity on S_2 under 10×5. Values are mean±SD over three seeds, shading marks the best mean within each method and anchor.

Metric	Method	Training	Base	Alloc.
$P \uparrow$	CaT	Local-only	9.06 ± 0.51	4.88 ± 0.33
		FedAvg	17.28 ± 2.13	4.71 ± 1.73
		FedProx	$\mathbf{20.01 \pm 0.34}$	$\mathbf{9.80 \pm 1.51}$
	GEM	Local-only	30.01 ± 1.75	$\mathbf{20.02 \pm 1.13}$
		FedAvg	29.70 ± 1.29	10.02 ± 2.54
		FedProx	$\mathbf{30.34 \pm 0.40}$	10.72 ± 5.17
	SSM	Local-only	10.13 ± 0.29	7.14 ± 1.58
		FedAvg	8.88 ± 1.81	$\mathbf{8.43 \pm 4.30}$
		FedProx	$\mathbf{15.91 \pm 3.25}$	5.76 ± 1.32
	TWP	Local-only	$\mathbf{4.86 \pm 1.30}$	4.49 ± 1.97
		FedAvg	1.97 ± 0.14	$\mathbf{5.38 \pm 3.50}$
		FedProx	2.86 ± 1.16	3.33 ± 0.47
AF $\downarrow$	CaT	Local-only	$\mathbf{25.16 \pm 1.96}$	$\mathbf{16.82 \pm 1.44}$
		FedAvg	35.34 ± 2.30	20.33 ± 4.41
		FedProx	36.30 ± 0.73	24.93 ± 7.14
	GEM	Local-only	45.47 ± 1.72	$\mathbf{19.06 \pm 0.67}$
		FedAvg	44.48 ± 2.74	42.27 ± 2.79
		FedProx	$\mathbf{43.08 \pm 1.64}$	45.87 ± 3.51
	SSM	Local-only	$\mathbf{51.91 \pm 1.42}$	$\mathbf{25.68 \pm 1.45}$
		FedAvg	53.41 ± 2.62	38.12 ± 6.07
		FedProx	52.89 ± 2.87	40.71 ± 4.05
	TWP	Local-only	$\mathbf{37.81 \pm 1.09}$	$\mathbf{22.21 \pm 1.43}$
		FedAvg	50.80 ± 4.28	34.59 ± 4.12
		FedProx	45.78 ± 6.46	39.78 ± 1.08

Table 27: Training-rule sensitivity on S_5 at the four anchors under 10×5. Values are mean±SD over three seeds, shading marks the best mean within each method and anchor.

Metric	Method	Training	Base	Alloc.	Order	Joint
$P \uparrow$	CaT	Local-only	19.61 ± 1.87	17.72 ± 2.49	$\mathbf{18.59 \pm 1.47}$	19.25 ± 2.03
		FedAvg	19.73 ± 2.78	16.50 ± 4.33	18.19 ± 6.04	$\mathbf{20.19 \pm 3.57}$
		FedProx	$\mathbf{20.35 \pm 5.71}$	$\mathbf{18.86 \pm 7.48}$	16.73 ± 5.59	19.33 ± 4.54
	GEM	Local-only	$\mathbf{20.78 \pm 3.89}$	20.23 ± 2.98	$\mathbf{24.38 \pm 1.12}$	$\mathbf{23.83 \pm 2.99}$
		FedAvg	20.32 ± 2.35	19.69 ± 2.89	21.27 ± 3.44	21.81 ± 2.11
		FedProx	18.19 ± 2.64	$\mathbf{22.40 \pm 2.41}$	19.88 ± 1.29	18.40 ± 5.22
	SSM	Local-only	$\mathbf{15.07 \pm 1.59}$	$\mathbf{18.43 \pm 1.86}$	18.87 ± 0.93	17.57 ± 3.29
		FedAvg	13.29 ± 1.56	14.76 ± 5.37	$\mathbf{20.77 \pm 1.89}$	17.45 ± 5.79
		FedProx	14.52 ± 0.84	14.69 ± 6.72	18.98 ± 1.02	$\mathbf{17.61 \pm 5.86}$
	TWP	Local-only	18.24 ± 1.02	17.23 ± 1.10	18.00 ± 3.67	17.35 ± 1.00
		FedAvg	$\mathbf{18.53 \pm 2.98}$	$\mathbf{18.81 \pm 4.80}$	$\mathbf{24.48 \pm 2.73}$	$\mathbf{24.78 \pm 0.98}$
		FedProx	$\mathbf{18.53 \pm 2.98}$	18.57 ± 4.61	$\mathbf{24.48 \pm 2.73}$	23.29 ± 2.17
AF $\downarrow$	CaT	Local-only	19.09 ± 3.72	$\mathbf{16.96 \pm 1.90}$	$\mathbf{21.09 \pm 1.74}$	16.77 ± 2.90
		FedAvg	$\mathbf{13.03 \pm 8.69}$	19.77 ± 5.68	21.28 ± 5.94	$\mathbf{16.65 \pm 4.23}$
		FedProx	18.19 ± 9.54	18.80 ± 9.96	22.53 ± 3.88	17.70 ± 2.78
	GEM	Local-only	$\mathbf{22.62 \pm 3.62}$	22.73 ± 6.94	19.59 ± 0.57	$\mathbf{19.17 \pm 0.60}$
		FedAvg	28.46 ± 5.95	28.27 ± 15.16	$\mathbf{19.08 \pm 2.69}$	19.18 ± 0.50
		FedProx	30.92 ± 6.88	$\mathbf{22.12 \pm 17.02}$	21.36 ± 3.71	20.00 ± 3.60
	SSM	Local-only	$\mathbf{15.14 \pm 0.30}$	14.21 ± 1.58	$\mathbf{11.89 \pm 1.36}$	$\mathbf{16.74 \pm 1.05}$
		FedAvg	16.96 ± 11.63	15.98 ± 4.09	19.86 ± 2.79	24.65 ± 4.78
		FedProx	21.89 ± 9.37	$\mathbf{11.16 \pm 4.12}$	16.42 ± 3.07	21.95 ± 3.94
	TWP	Local-only	$\mathbf{31.04 \pm 2.55}$	$\mathbf{31.74 \pm 5.65}$	28.49 ± 3.66	23.61 ± 2.60
		FedAvg	38.71 ± 11.19	38.30 ± 11.40	15.62 ± 3.33	$\mathbf{12.72 \pm 1.74}$
		FedProx	38.80 ± 11.28	37.80 ± 12.37	$\mathbf{15.46 \pm 3.22}$	17.02 ± 4.60

Table 28: Training-rule sensitivity on S_7 (GEMSEC Facebook) under 10×5. Values are mean±SD over five seeds, shading marks the best mean within each method and anchor.

Metric	Method	Training	Base	Alloc.
$P \uparrow$	EWC	Local-only	$\mathbf{47.72 \pm 8.28}$	41.31 ± 13.59
		FedAvg	40.41 ± 14.94	$\mathbf{41.45 \pm 8.69}$
		FedProx	42.23 ± 5.67	39.52 ± 14.20
	LwF	Local-only	$\mathbf{48.59 \pm 8.45}$	42.63 ± 6.97
		FedAvg	43.18 ± 11.19	$\mathbf{42.75 \pm 6.23}$
		FedProx	38.81 ± 8.19	38.48 ± 12.24
	GraphKeeper	Local-only	42.18 ± 6.74	$\mathbf{42.95 \pm 12.86}$
		FedAvg	41.95 ± 4.25	38.60 ± 11.90
		FedProx	$\mathbf{44.99 \pm 5.79}$	38.22 ± 8.16
AF $\downarrow$	EWC	Local-only	$\mathbf{8.15 \pm 6.55}$	10.04 ± 12.98
		FedAvg	9.09 ± 15.40	8.88 ± 14.82
		FedProx	8.77 ± 13.40	$\mathbf{5.04 \pm 6.71}$
	LwF	Local-only	$\mathbf{5.91 \pm 5.27}$	7.61 ± 8.93
		FedAvg	7.85 ± 12.74	6.77 ± 14.00
		FedProx	8.03 ± 12.44	$\mathbf{6.49 \pm 7.92}$
	GraphKeeper	Local-only	14.96 ± 7.55	11.06 ± 5.33
		FedAvg	9.99 ± 11.84	5.78 ± 4.96
		FedProx	$\mathbf{7.59 \pm 7.06}$	$\mathbf{4.63 \pm 4.10}$

relative to seed variation. These comparisons therefore need to be read with both final performance and forgetting.

Allocation effects under synchronized order. To compare allocation responses over the same task-order setting, we use synchronized results for all three scenarios. For each method and seed,

$$E^{P}_{\text{alloc|sync}} = P_{100,\text{S}} - P_{0.1,\text{S}}, \tag{17}$$

$$E^{\text{AF}}_{\text{alloc|sync}} = \text{AF}_{0.1,\text{S}} - \text{AF}_{100,\text{S}}. \tag{18}$$

Positive values indicate higher final performance or lower AF at $\alpha = 100$. Table 29 reports the median of the method-level paired-seed mean effects and the number of strictly positive method means. These summaries are descriptive and are not averaged over task orders.

More even allocation gives positive median performance effects in NC and LP across the tested training rules. Forgetting responds differently: median AF rises in NC, while the LP response depends on the rule. LC shows smaller, mixed effects. These findings apply to the two allocation endpoints under synchronized order.

Table 29: Allocation effects under synchronized order (10×5). Entries are medians of method-level effects, counts in parentheses show positive method means. Positive values favor $\alpha = 100$.

Scenario	**Training**	$E^{P}_{\text{alloc\|sync}}$	$E^{\text{AF}}_{\text{alloc\|sync}}$
NC	Local-only	$+3.58$ (4/4)	-20.92 (0/4)
NC	FedAvg	$+6.50$ (3/4)	-15.15 (0/4)
NC	FedProx	$+10.18$ (3/4)	-8.68 (1/4)
LC	Local-only	$+0.78$ (3/4)	-0.41 (2/4)
LC	FedAvg	$+0.18$ (2/4)	-0.31 (1/4)
LC	FedProx	-0.10 (1/4)	-4.90 (1/4)
LP	Local-only	$+5.96$ (2/3)	$+1.69$ (2/3)
LP	FedAvg	$+0.43$ (2/3)	-1.08 (0/3)
LP	FedProx	$+2.71$ (3/3)	-2.96 (0/3)

Four-anchor factor effects for LC. LC additionally supports allocation contrasts averaged over both task orders and order contrasts averaged over $\alpha \in \{0.1, 100\}$. Table 30 reports the corresponding method-level summaries. Its order columns measure the change from unsynchronized to synchronized order:

$$E^{P}_{\text{U}\to\text{S}} = \frac{1}{2} \sum_{\alpha\in\{0.1,100\}} \left(P_{\alpha,\text{S}} - P_{\alpha,\text{U}}\right), \tag{19}$$

$$E^{\text{AF}}_{\text{U}\to\text{S}} = \frac{1}{2} \sum_{\alpha\in\{0.1,100\}} \left(\text{AF}_{\alpha,\text{U}} - \text{AF}_{\alpha,\text{S}}\right). \tag{20}$$

Positive order effects favor synchronized order. Under FedAvg, unsynchronized order improves mean final accuracy across the evaluated LC methods, the AF response varies. Changing the training rule can change these responses, which also occur under Local-only. This comparison therefore leaves the contribution of cross-client aggregation unresolved. Its task-order findings apply to LC only.

E.3 Constructor Sensitivity of Learning Results

We compare node-wise and topology-aware construction on synchronized S_2 at $\alpha \in \{0.1, 100\}$. Both constructors target the same client-by-class training allocation. Task definitions, data splits, task order, participation, model initialization, backbone, training budget, and federated rule are matched. Ownership and local neighborhoods may differ, so this comparison measures the effect of the constructor as a whole rather than edge retention alone.

Table 30: Four-anchor factor effects for S_5 under 10×5. Entries are medians of method-level effects, counts in parentheses show positive method means. Positive allocation effects favor $\alpha = 100$, and positive order effects favor synchronized order.

Training	E^{P}_{alloc}	$E^{\text{AF}}_{\text{alloc}}$	$E^{P}_{\text{U}\to\text{S}}$	$E^{\text{AF}}_{\text{U}\to\text{S}}$
Local-only	$+0.58$ (3/4)	-1.12 (1/4)	-0.86 (1/4)	-1.83 (1/4)
FedAvg	$+0.33$ (3/4)	$+0.50$ (2/4)	-3.31 (0/4)	-3.33 (2/4)
FedProx	$+0.01$ (2/4)	-2.35 (1/4)	-1.27 (2/4)	-2.11 (2/4)

We evaluate CaT, DSLR, GEM, SSM, and TWP over three paired seeds. For each seed, the constructor effects are

$$E^{P}_{\text{partition}} = P_{\text{topology-aware}} - P_{\text{node-wise}}, \tag{21}$$

$$E^{\text{AF}}_{\text{partition}} = \text{AF}_{\text{node-wise}} - \text{AF}_{\text{topology-aware}}. \tag{22}$$

Positive values favor topology-aware construction. Table 31 reports the corresponding final-performance and forgetting results.

Table 31: Constructor sensitivity on synchronized S_2 at $\alpha \in \{0.1, 100\}$. Values are mean±sample SD over three paired seeds. Positive paired effects favor topology-aware construction.

Metric	**Method**	α	**Node-wise**	**Topology-aware**	**Paired effect**
$P\uparrow$	CaT	0.1	6.10 ± 1.71	4.40 ± 2.11	-1.69 ± 0.88
		100	20.18 ± 2.06	18.98 ± 0.93	-1.20 ± 2.73
	DSLR	0.1	5.94 ± 1.11	5.68 ± 0.81	-0.26 ± 1.73
		100	12.52 ± 2.55	25.05 ± 2.00	$+12.53 \pm 2.13$
	GEM	0.1	11.88 ± 3.18	11.76 ± 5.11	-0.13 ± 1.93
		100	23.37 ± 2.18	33.54 ± 4.04	$+10.17 \pm 4.63$
	SSM	0.1	8.72 ± 4.66	7.53 ± 5.20	-1.18 ± 0.55
		100	9.27 ± 0.51	10.17 ± 2.01	$+0.90 \pm 2.16$
	TWP	0.1	4.56 ± 2.06	4.38 ± 0.33	-0.19 ± 1.74
		100	4.05 ± 2.09	4.23 ± 2.99	$+0.18 \pm 0.90$
AF $\downarrow$	CaT	0.1	27.40 ± 1.11	23.64 ± 5.80	$+3.77 \pm 6.15$
		100	33.26 ± 2.51	42.13 ± 0.31	-8.87 ± 2.73
	DSLR	0.1	63.45 ± 0.95	51.45 ± 7.28	$+12.00 \pm 6.91$
		100	67.27 ± 1.72	53.96 ± 2.90	$+13.31 \pm 2.65$
	GEM	0.1	59.46 ± 4.34	51.05 ± 9.23	$+8.41 \pm 6.80$
		100	58.24 ± 3.53	43.23 ± 4.58	$+15.00 \pm 5.73$
	SSM	0.1	59.77 ± 3.75	49.14 ± 6.81	$+10.63 \pm 3.28$
		100	62.14 ± 2.36	54.93 ± 6.02	$+7.20 \pm 4.62$
	TWP	0.1	46.03 ± 4.81	45.98 ± 6.23	$+0.05 \pm 1.53$
		100	37.89 ± 0.95	33.56 ± 1.77	$+4.33 \pm 2.19$

Constructor effects depend on both allocation and learner. Under concentrated allocation, lower AF accompanies slightly lower mean final performance. At the more even allocation, most methods have higher mean final performance and lower mean AF. These paired comparisons measure the combined changes in ownership and local neighborhoods, the contribution of retained-edge count remains unresolved.

F Acquisition and Retention Analysis

Table 4 distinguishes whether low final performance follows poor initial learning, loss of previously acquired performance, or both. This appendix defines the reference settings and the additional quantities used to interpret that comparison. The reference protocols used in this analysis are summarized in Table 32.

Table 32: Reference protocols used in the learnability analysis of Table 4. All-task Training removes sequential task availability and is used as a learnability reference rather than a compute-matched upper bound.

Reference	**Training information**	**Local memory**	**Role**
Training-label prior	Class frequencies from training labels encountered by the client	None	Non-learning reference
Bare	Current local task only	None	Continual-learning reference
Simple Replay	Current local task together with stored examples from earlier tasks	At most 100 examples per client	Replay reference
All-task Training	All local training tasks available from the start	None	Learnability reference

F.1 Reference Protocols

We use four references on S_2 and S_5 at the Base and Joint conditions. All references use the same client-local data and evaluation protocol as the corresponding benchmark condition. They differ only in how training supervision is made available.

Bare and Simple Replay use the common GCN with FedAvg. All-task Training uses the same number of client updates as the corresponding continual runs, but each update can access supervision from all local tasks and therefore processes more training examples. It is consequently not a compute-matched upper bound. Acquisition is the test performance on a client-task pair at the end of the first stage in which that client encounters the task. Final training fit evaluates the final model on all training examples of that client-task pair using the same output mask as evaluation. Both are averaged uniformly over valid client-task pairs within each seed. Acquisition and AF are not defined for all-task Training because its tasks are available jointly from the beginning.

F.2 Acquisition, Retention, and Final Fit

Table 4 shows strong initial learning followed by performance loss in NC at Base, weaker acquisition also contributes under Joint. LC shows weak training fit even when all tasks are available together, alongside performance loss after acquisition. The source of this fitting difficulty remains unresolved under the tested setup. Acquisition and final performance are therefore needed to interpret low AF.

F.3 Cross-client task support

A task still being learned by one client can support another client's earlier predictions. We relate this support to receiver loss and examine how acquisition enters the forgetting score.

For the reported accuracies $M_{k,s,t}$, acquisition is $a_{k,t} = M_{k,\tau_{k,t},t}$ and final performance is $p_{k,t} = M_{k,T,t}$. The best score from acquisition onward is $h_{k,t} = \max_{\tau_{k,t} \le s \le T} M_{k,s,t}$. Averaging $h - p = (a - p) + (h - a)$ uniformly over the same valid client–task pairs $\mathcal{I}$ gives

$$\mathrm{AF} = \mathrm{Acq} - P + \Gamma, \qquad \Gamma = \frac{1}{|\mathcal{I}|} \sum_{(k,t)\in\mathcal{I}} (h_{k,t} - a_{k,t}) \ge 0, \tag{23}$$

$$\mathrm{AF_S} - \mathrm{AF_U} = (\mathrm{Acq_S} - \mathrm{Acq_U}) + (P_\mathrm{U} - P_\mathrm{S}) + (\Gamma_\mathrm{S} - \Gamma_\mathrm{U}). \tag{24}$$

These identities require the same weights for acquisition, final performance, and AF, LP uses different weights. Table 33 covers all 80 client-task pairs in the separate balanced-participation study. Unsynchronized order lowers acquisition and AF at both allocation levels, while its final-performance response varies. Lower initial learning can therefore contribute to a lower forgetting score.

Table 33: **Learning history and AF.** (a) Mean±SD over three seeds. (b) Seed-level differences, the last column gives their sum before rounding. Shading marks the better mean within each α.

(a) Learning-history scores

α	Order	$P \uparrow$	AF $\downarrow$	Acq $\uparrow$
0.1	Sync.	$\mathbf{7.332 \pm 1.051}$	60.181 ± 1.691	$\mathbf{67.513 \pm 1.014}$
0.1	Unsync.	6.258 ± 2.101	$\mathbf{35.135 \pm 1.115}$	35.785 ± 3.394
100	Sync.	10.559 ± 0.291	74.531 ± 0.679	$\mathbf{85.090 \pm 0.969}$
100	Unsync.	$\mathbf{15.199 \pm 4.829}$	$\mathbf{50.283 \pm 3.942}$	63.273 ± 0.729

(b) Paired seed differences

α	Seed	$\mathrm{Acq_S} - \mathrm{Acq_U}$	$P_\mathrm{U} - P_\mathrm{S}$	$\Gamma_\mathrm{S} - \Gamma_\mathrm{U}$	$\mathrm{AF_S} - \mathrm{AF_U}$
0.1	0	$+26.748$	$+1.302$	-3.986	$+24.064$
0.1	1	$+34.471$	-0.494	-5.857	$+28.120$
0.1	2	$+33.964$	-4.029	-6.982	$+22.953$
100	0	$+20.961$	$+10.163$	-3.164	$+27.960$
100	1	$+22.960$	$+0.088$	-1.273	$+21.776$
100	2	$+21.530$	$+3.669$	-2.189	$+23.010$

G Reproducibility and Release

Benchmark Records Each GECKO condition records its task definitions, client-local graph and prediction data, task order, participation schedule, and protocol version independently of the learning method. The same condition can therefore be reused across methods, while per-seed result records preserve the identifiers needed to regenerate the reported summaries and paired comparisons. The final artifact will include the benchmark construction and evaluation code, experiment configurations, environment specifications, and scripts used to produce the reported results. Versioned records distinguish changes to the benchmark protocol from changes to a learning method.

Data and Code Availability GECKO uses OGBN-Arxiv and OGBN-Proteins (Hu et al., 2020), Bitcoin-OTC (Kumar et al., 2016), and GEMSEC Facebook (Rozemberczki et al., 2019). Dataset use and redistribution follow the terms of the original sources. When redistribution is not permitted, the benchmark retrieves the original data and constructs the required benchmark instances locally. The final artifact will provide the public repository, release version, and instructions for constructing benchmark conditions and reproducing the reported results.

H Benchmark Scope

Scope. GECKO studies FCGL on a fixed graph that is divided across clients. Each client's message passing and prediction queries are restricted to its local graph, training is synchronized across clients, and every client eventually encounters all tasks. The benchmark varies how graph data are distributed and the order in which tasks are seen, allowing methods to be compared under matched conditions. These controlled settings are designed for analysis and may not reflect naturally occurring client partitions.

Interpretation. NC, LC, and LP use different prediction problems and evaluation rules, so their scores should be interpreted separately rather than combined into one overall ranking. Likewise, when changing client allocation also changes the local graph or the examples available for learning, the resulting performance change should not be attributed to a single cause. Future versions can extend GECKO to changing graphs, broader task orders, different task availability across clients, and cross-client graph access.