ONE GRAPH, EVERY NODE, HIDDEN TASK: HIDE, A PROTOCOL FOR GRAPH CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph class-incremental learning (GCIL) methods are compared by their accuracy under the CGLB benchmark protocol. CGLB supplies the task identity the class-incremental setting is meant to withhold, in two ways: test nodes are scored one task at a time, and each node is scored on the subgraph of its own task. We measure what each of them, and each other evaluation choice, contributes to reported accuracy. We introduce HIDE (Hidden-task IDentity Evaluation), which scores every test node in one batch, on the full graph, over every class seen so far, with accuracy averaged over nodes, and we vary each choice independently. We evaluate six methods on eight datasets, using one shared graph per dataset, alongside two reference classifiers with no continual-learning mechanism, one of them trained jointly on all tasks. Every method that does not already collapse under CGLB loses accuracy under HIDE; the one ranked first under CGLB on seven of eight datasets ranks last under HIDE on seven of eight, with its per-node accuracy falling from 97% to 32% on ogbn-arxiv and from 87% to 12% at the widest. On ogbn-arxiv, no method beats either reference classifier. We release the shared graphs, the splits, the reference classifiers, and every per-seed result. We propose assessing novel GCIL methods on HIDE, where they are evaluated under realistic, more challenging hidden-task node-by-node class-incremental learning.

1 INTRODUCTION

In class-incremental learning, a model learns from a sequence of tasks, each introducing classes not seen before, and must then classify inputs among all classes learned so far, without being told which task an input belongs to (van de Ven et al., 2022). Graph class-incremental learning (GCIL) applies this setting to graph data ç. In its node-level form, which this paper focuses on, each task adds nodes of unseen classes to one growing graph. Progress in GCIL is measured by accuracy under the protocol of the CGLB benchmark (Zhang et al., 2022a), and several methods have reported improvements under it (Zhang et al., 2022b; 2024; Niu et al., 2024a;b). Accuracy, however, also depends on choices made by the evaluation protocol.

The GCIL evaluation protocol comprises six choices: (Section 3). Four are the subject of this paper. CGLB evaluates the test nodes of each task together (i, *batch composition*), so a method that summarizes its input batch can infer the task, the information the class-incremental setting withholds; it scores each task on the subgraph induced by its own nodes (ii, *scoring graph*), so a test node never has a neighbor from another task; it averages accuracy over classes and then over tasks (iii, *aggregation*); and released implementations build the graph through different libraries (iv, *data loading*), so two methods evaluated on one dataset may read different graphs. The other two are held fixed: (v, *class scope*), which classes are evaluated and with how many nodes each, stays at each dataset's native distribution and the (vi, *training graph*), whether inter-task edges are visible while a method trains.

Batch composition and aggregation were recently addressed by Cheng et al. (2025), whose benchmark evaluates all tasks in one batch and averages over nodes; Ko et al. (2025) score class-incremental node classification on the full graph through one loader and average over tasks; neither separates what each choice costs, and neither compares loaders.

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.

We define HIDE (Hidden-task IDentity Evaluation), an evaluation setting that removes every source of task information from the protocol: all test nodes are scored in a single batch, on the full graph with its inter-task edges, over all classes seen so far, with accuracy averaged over nodes. It is the class-incremental scenario these methods are designed for, with the evaluation choices made explicit, revealing their individual contribution to the reported accuracy.

Between CGLB's protocol and HIDE, we ablate each of the six choices, one choice at a time, to see how they impact the methods performance (Section 5.3). We evaluate six published GCIL methods (PDGNNs, DeLoMe, SSM, TPP, TAAM, and DyMoE) over ten seeds on eight datasets (CoraFull, ogbn-arxiv, Cora, CiteSeer, Amazon Photos, Amazon Computers, WikiCS, and Roman-Empire), alongside two reference classifiers with no continual-learning mechanism: feature propagation over the graph and a model trained jointly on all tasks, that represents the performance upper bound (Section 4), and a cascade of frozen per-task experts that cannot forget by construction. We compute error on task-identification part and a within-tasks (Section 5.2).

Our paper has four contributions: **1.** We show that most GCIL methods performance degrades under HIDE on every dataset evaluated, altering even the methods ranking, for instance method PDGNNs was the best performing one under CGLB, but it loses accuracy and becomes worse than the references and the baseline. Among the methods with released code, TPP ranks first under CGLB's protocol on seven of eight datasets and last under HIDE on seven of eight; on ogbn-arxiv its per-node accuracy falls from 97% to 32%. **2.** The performance degradation is due to incorrect task identification. Given the correct task, the CGIL methods classify 85% to 98% of test nodes correctly on CoraFull and ogbn-arxiv, yet they assign only 33% to 75% of nodes to the correct task. Therefore, the biggest gap on current CGIL methods appears to be task identification **3.** The protocol choices *Batch Composition*, *Scoring Graph*, *Aggregation* and *Data Loading* have different impacts on the performance of different methods. *Batch Composition* of Mixed batches cost TPP and TAAM between 7.5 and 49.5 points of per-node accuracy; inter-task edges lower the performance of all the evaluated methods; and the loader alone separates DeLoMe and SSM by 21.5 points on ogbn-arxiv, a gap that shrinks to 7.1 once one hyperparameter of DeLoMe is reselected on calibration data. **4.** On ogbn-arxiv no method exceeds either reference classifier, and the propagation reference outperforms the jointly trained model in its standard configuration on five of the eight datasets, so that model is not the ceiling it is treated as.

2 Background

2.1 Graph class-incremental learning

Graph class-incremental learning (GCIL) is node classification on a graph whose label space expands over time (Zhang et al., 2022a). In GCIL, training proceeds over a sequence of tasks, each bringing nodes of new classes together with their labels and edges. Replay-based methods may only save nodes and labels of a task during training.

There are two scenarios in GCIL that differ in task availability during inference time (van de Ven et al., 2022): in the *task-incremental* scenario, the task identity is provided during inference, so the model only discriminates among the classes of that task; whereas in the *class-incremental* scenario the task is not provided during inference, so the model has to identify both the task and the class. This paper addresses the *class-incremental* scenario.

Kim et al. (2022) show that a class-incremental classifier is accurate only if both task identification and classification are accurate. The authors also frame task identification step as out-of-distribution detection on an input considered in isolation. In the context of GNN however (Kipf & Welling, 2017; Hamilton et al., 2017), prediction of the node label depends both on its own features and on the features of its neighbors (Su et al., 2023). Which neighbors, possibly from many distinct tasks, the model sees at test time is therefore decided by the evaluation protocol (Section 3); Section 5.2 measures the two steps separately for each method.

Notation. Let $G = (V, E)$ be a graph in which each node $v \in V$ carries a feature vector x_v, a ground-truth label y_v, and its edges E. The T tasks have class sets $C_1, \ldots, C_T$, which are non-empty and pairwise disjoint; $V_t \subseteq V$ denotes the task nodes whose label lies in C_t. Every node belongs to exactly one of three splits (training, calibration, and test), so the test nodes of task t are

Table 1: The three settings this paper measures, on the four choices it varies.

	batch S	inter-task edges	aggregation	data loading
Isolated (CGLB, Zhang et al., 2022a)	task-pure	absent	per task, per class	shared
Mixed	mixed	absent	per node	shared
HIDE (this work)	mixed	present	per node	shared
LLM4GCL (Cheng et al., 2025)	mixed	absent	per node	own
BeGin (Ko et al., 2025)	mixed	present	per task	own

$V_t \cap V_{\text{test}}$. An edge whose endpoints belong to different tasks is an *inter-task edge*; the *task subgraph* $G[V_t]$, induced by the nodes of task t, therefore contains none. The *edge homophily* of a graph is the fraction of its edges that join two nodes with the same label.

2.2 Related work

CGLB (Zhang et al., 2022a) is the de facto benchmark for GCIL: SSM, PDGNNs, DeLoMe and TPP report their results under it, while older methods are evaluated through their own implementations. CGLB standardizes the datasets, the train/validation/test partition (its validation set serves as our calibration set), the mapping of classes to tasks, and the evaluation pipeline. We focus on the node-level track, which comprises four datasets. The more recent LLM4GCL (Cheng et al., 2025) departs from CGLB on three choices analyzed in Section 3: it evaluates all tasks jointly, averages accuracy over nodes, and rebalances class sizes. Ko et al. (2025) reorganizes CGIL on graphs, however it fails to identify how task occlusion during inference impacts each CGIL method.

Vulnerability of rankings to the evaluation protocol. Evaluation protocol impacting GCIL leaderboards has been observed before. Masana et al. (2023) found that the best image class-incremental method changes with the memory budget and the task order, Shchur et al. (2018) found that the ranking of GNN architectures on the citation datasets changes with the split, and Errica et al. (2020) found that, under consistent splits and tuning, several graph-classification models do not beat a structure-agnostic baseline, and Platonov et al. (2023) found that, on heterophilous benchmarks, standard GNNs match heterophily-specific designs once duplicated nodes are removed. Continual learning compares against joint training, a single model trained on all tasks at once, as an upper bound (van de Ven et al., 2022). Section 4 proposes new GCIL baselines that follow this reasoning.

3 What the Evaluation Protocol Decides

3.1 Choices and settings

Of the six protocol choices, this paper varies four. Two are fixed as controls: the *class scope*, which determines which classes are evaluated and with how many nodes each, held at each dataset's native distribution, and the *training graph*, which determines whether inter-task edges are visible during training, which only one released implementation lets us vary (Section 5.3). We call a *setting* a specific combination of the four varied choices, as shown in Table 1.

Two choices impact the prediction directly. The *evaluation batch* $S \subseteq V_{\text{test}}$ is the set of test nodes evaluated together with a node, and the *scoring graph* G' is the graph the model reads while predicting. In this paper, S is either $V_t \cap V_{\text{test}}$ or all of V_{test}, and G' is either the task subgraph $G[V_t]$ or the full graph G. Predictions always range over every class seen so far.

The *cost* of a choice, for a given method, is the difference in accuracy between two settings that differ in that choice alone, computed per seed and averaged over seeds. A negative cost is a loss. The published settings change batch composition and aggregation together. We separate them: obtaining the aggregation cost by evaluating one set of predictions under both averages, and the batch cost by using per-node accuracy on both sides, so no cost compares an AA value with a per-node value.

Batch composition. The batch is *task-pure* if it holds the test nodes of a single task and *mixed* if it holds those of all tasks. Isolated scores each task in its own task-pure batch, Mixed and HIDE score all test nodes in one mixed batch. A task-pure batch leaks task identity to any method that computes

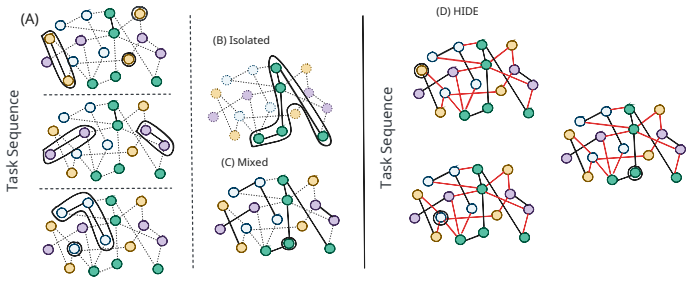


Figure 1: The task sequence (A) and the three evaluation settings compared in this paper (B–D), shown on one toy graph with two classes per task. Node color indicates the task; a double outline marks a test node being evaluated; solid edges are visible to the model, dotted edges are withheld, and red edges connect nodes of different tasks.

statistics of its input batch before classifying, since a summary of the batch is then a summary of one task. TPP and TAAM do this. With a mixed batch however, they must estimate the task of each node individually, which is how we evaluate them, using for TPP a per-node task estimate we added to its released code, which estimates the task only from the mean of the batch. The batch cost is the change in per-node accuracy from Isolated to Mixed, it is zero for every method whose prediction for a node does not depend on the rest of the batch.

Scoring graph. Inter-task edges are either *absent*, $G' = G[V_t]$, so that every neighbor of a test node belongs to its own task, or *present*, $G' = G$. Isolated and Mixed score each node on the subgraph of its own task, and HIDE scores it on G, the step from Mixed to HIDE changes only this choice, so its cost is the cost of the inter-task edges.

Aggregation. CGLB reports *average accuracy* (AA): the per-class accuracy of the final model, averaged over the classes of each task and then over tasks. With $V_c \subseteq V_{\text{test}}$ the test nodes of class c,

$$\text{AA} = \frac{1}{T}\sum_{j=1}^{T}\frac{1}{|C_j|}\sum_{c\in C_j}\frac{1}{|V_c|}\sum_{v\in V_c}\mathbf{1}[\hat{y}_v = y_v]. \tag{1}$$

AA gives every class the same weight within its task and every task the same weight overall, so a class with thirty test nodes counts as much as one with three thousand; per-node accuracy, $\text{Acc} = |S|^{-1}\sum_{v\in S}\mathbf{1}[\hat{y}_v = y_v]$, weights each test node equally. The two agree only when all classes have equally many test nodes and all tasks equally many classes, which holds for none of our datasets. Isolated reports AA, as CGLB does, and Mixed and HIDE report per-node accuracy, Isolated and HIDE are read under both averages (Appendix N), so the aggregation cost is the difference between the two averages of one set of predictions.

Data loading. The graph a method reads is either its *own graph*, the one its released loader builds, or the *shared graph*, one graph per dataset that every method reads (Section 3.2). The three settings are read on the shared graph the difference between the two is the cost of the graph choice.

The three settings. Figure 1 illustrates the three settings on a toy graph whose task sequence is shown in panel A, one row per arriving task. *Isolated* (panel B), the edge-free setting of CGLB, evaluates the test nodes of one task together, on that task's subgraph, and reports AA; the nodes of other tasks are faded because the model does not see them. *Mixed* (panel C) evaluates the test nodes of all tasks in one batch and reports per-node accuracy, each node still on its own task subgraph. *HIDE* (panel D) additionally evaluates every node on the full graph with its inter-task edges, the three copies show the test nodes of successive tasks arriving for evaluation.

Mixed is the intermediate setting that separates the two differences between Isolated and HIDE. The step from Isolated to Mixed changes batch composition and aggregation. The step from Mixed to HIDE changes only the scoring graph. Mixed matches LLM4GCL's setting on batch composition, scoring graph, and aggregation, differing only in class scope, which this paper holds native since accuracies under a rebalanced scope are not comparable with published results.

3.2 What is held fixed

Isolating the effect of one choice requires that everything else stay the same. All methods therefore read the same graph, use the same partition of nodes into training, calibration, and test sets for a given seed, follow the same assignment of classes to tasks, and are evaluated on the same test nodes. The partition is drawn per class in proportions of 60%, 20%, and 20% and replaces the split distributed by each dataset's publisher. It is redrawn for each of the ten seeds, so every paired difference spans ten partitions.

Training labels are available while their task is active. Calibration labels are available for the selection of whatever training does not determine and, for the reference classifiers of Section 4, as training labels and as labels in their smoothing step. Test labels are never revealed. Every method runs at the hyperparameters its authors released, with one exception whose effect we measure (Section 5.3).

Agreeing on a split does not fix the graph: each released implementation loads its datasets through its own library, and the resulting graphs differ, in edge direction and multiplicity on ogbn-arxiv (Hu et al., 2020) and in the feature encoding on CoraFull (Bojchevski & Günnemann, 2018) (Appendix E). Every method therefore reads the *shared graph*, the undirected graph with the features distributed by the dataset's publisher, in place of the *own graph* its loader would build. Section 5.3 measures the cost of this *graph choice*.

Throughout, $\mathrm{Acc}(\cdot)$ denotes per-node accuracy under HIDE, and any other reading is named where it is used. Differences between accuracies are paired per seed and given in percentage points, with the number of seeds agreeing on the sign. Appendices U and V give bootstrap intervals.

4 Reference Classifiers and the Cascade Baseline

Interpreting GCIL accuracy requires comparison with classifiers that do not use a continual-learning mechanism (Farquhar & Gal, 2018). We use two reference families for this purpose: a propagation classifier, R_{prop}, and jointly trained models, collectively denoted R_{joint}. Tables 2 and 3 report three reference rows: R_{prop} and two joint-training variants, R_1 and R_4. Separately, we introduce a cascade of frozen task-specific experts as a diagnostic baseline: it tests whether hidden-task evaluation remains difficult when subsequent training cannot overwrite earlier experts. Together, these reference comparisons examine the performance attainable without a continual-learning mechanism and the protocol's effects when expert parameters are preserved.

4.1 Propagation reference

The propagation reference combines graph diffusion, linear classification, and prediction refinement. It first diffuses node features using personalized PageRank (Gasteiger et al., 2019), then fits a linear classifier over all classes, following the use of propagated features with linear models (Wu et al., 2019). Finally, it refines the class probabilities using Correct and Smooth (Huang et al., 2021). R_{prop} retains training and calibration labels, including calibration supervision in its prediction-refinement step. It does not train incrementally, maintain task-specific parameters, or infer task identity. It uses the same scoring graph available to the evaluated methods: the full graph G under HIDE and the task subgraph under Isolated and Mixed.

4.2 Jointly trained references

R_1, is the conventional joint baseline (R_{joint}) from (Zhang et al., 2022a), trained on the union of the task subgraphs. R_1 differs from R_{prop} in three ways: (i) in its classifier, a GCN in place of a linear head, (ii) in the information it uses, since it trains without the inter-task edges and the calibration labels, and (iii) in its prediction refinement, since it applies no correct-and-smooth step. R_2 instead trains on the full graph G. R_3 additionally includes calibration labels in fitting, and R_4 further applies Correct and Smooth over G. Thus, R_4 has the same information access and prediction-refinement step as R_{prop}, although the classifiers differ. R_2 and R_3 are intermediate ablations. The main tables (2 and 3) report R_1 and R_4.

Appendix K describes the configuration and hyperparameter search of the reference models. The reference-to-reference comparisons of Section 5.4 use the runs reported in Appendix L, scored un-

der HIDE on G, with differences paired by seed and expressed in percentage points. Since R_1 trains without inter-task edges but is scored with them, $\mathrm{Acc}(R_2) - \mathrm{Acc}(R_1)$ reflects both additional training connectivity and removal of that training–evaluation mismatch.

4.3 CASCADE AS A DIAGNOSTIC BASELINE

We introduce also a simple *cascade* baseline in our evaluations. The cascade baseline trains one network per task and freezes it when that task ends. At inference, the cascade baseline assigns each node to an expert, which predicts a class within its task. This construction prevents later training from overwriting earlier experts while retaining the need to identify the task of each test node. Errors can arise from either incorrect routing or classification within the correct task. The cascade therefore provides a control for the difference between parameter forgetting and difficulty introduced by hidden-task evaluation.

Routing compares a node's diffused features with the subspace spanned by each task's calibration features. The label-memory tier evaluated in Section 5 supplements this score with task probabilities obtained by summing the class probabilities of R_{prop} within each task. It assigns nodes independently of batch composition, so its batch cost is zero. This tier carries R_{prop} in full and retains the same labels, while additionally storing task-specific experts and calibration feature vectors. Its comparison with the reference therefore matches label retention but not total memory. Appendix M describes the routing procedure, the streaming and transductive alternatives, and calibration-based hyperparameter selection.

5 EXPERIMENTS

Setup. We evaluate six GCIL methods: PDGNNs, DeLoMe, SSM, TPP, TAAM and DyMoE (See Appendix H for details on methods we had to change or reimplement in our study). Under the conditions of Section 3.2, every method runs with two classes per task in class-index order, over ten seeds. Tables provide means and standard deviations over seeds, a dagger marks a difference on whose sign the seeds do not all agree, and TAAM and DyMoE rows are reimplementations. Hardware and software environments are given in Appendix C. The datasets are CoraFull (Bojchevski & Günnemann, 2018) and ogbn-arxiv (Hu et al., 2020), the two datasets of the CGLB node-level track supported by all four released implementations, and six others: Cora and CiteSeer (Sen et al., 2008), Amazon Photo and Amazon Computers (Shchur et al., 2018), WikiCS (Mernyei & Cangea, 2020) and Roman-empire (Platonov et al., 2023). The eight span cross-task edge fractions range from 0.129 to 0.892, and edge homophily ranges from 0.827 to 0.047 (Table 6).

5.1 FROM ISOLATED TO HIDE

Inter-task edges. Every method loses accuracy when inter-task edges are added to the scoring graph. The step from Mixed to HIDE lowers the mean accuracy of all six methods on every dataset by 0.09 points to 58.73, the only exception being SSM on Amazon Computers, whose training collapsed to the classes of the first task on every seed. The references drop as well: R_{prop} falls from 86.74% to 70.64% on CoraFull and from 89.95% to 74.46% on ogbn-arxiv (Table 2).

Batch composition and aggregation. For PDGNNs, DeLoMe, SSM, DyMoE, and the cascade, whose prediction for a node does not depend on the rest of the batch (Section 3.1), the step from Isolated to Mixed is purely a change of aggregation: their per-node accuracy is identical under task-pure and mixed batches on every dataset and every seed. This aggregation change moves them by -13.44 to $+21.45$ points, with no consistent sign across datasets, as shown in Table 3. For TPP and TAAM, the same step also changes the batch, and the batch alone costs them 7.48 to 49.45 points on every dataset and every seed, more than the at most 7.18 points the change of aggregation returns, so the combined step is a net loss for both everywhere (Table 3).

Spread and ranking. Between Isolated and HIDE, TPP and TAAM lose 40.45 to 64.89 points per node on CoraFull and ogbn-arxiv, more than the entire spread among the six methods under HIDE (19.32 and 30.95 points). The other four lose 0.41 to 18.06, and the relative order of the three replay methods is unchanged on CoraFull, while on ogbn-arxiv PDGNNs and SSM, 0.43 points apart under Isolated, swap. Under AA, CGLB's own metric, the losses are larger still, up to 75.28 points for

Table 2: Accuracy under CGLB's protocol (Isolated) and under HIDE, shared graph. Mixed and HIDE-AA columns in Appendix N. Every method is at its released configuration. Table 10 on Appendix F provides the three replay methods on ogbn-arxiv with one hyperparameter reselected on calibration.

	CoraFull			ogbn-arxiv		
	Isolated		HIDE	Isolated		HIDE
Method	AA	per node	per node	AA	per node	per node
PDGNNs	81.39 ± 0.55	86.27 ± 0.43	74.44 ± 0.64	65.61 ± 0.56	69.48 ± 0.43	63.08 ± 0.24
DeLoMe	80.34 ± 0.58	84.15 ± 0.59	71.37 ± 1.08	52.82 ± 0.94	39.38 ± 2.84	34.59 ± 2.22
SSM	76.50 ± 0.91	81.22 ± 0.85	63.16 ± 0.87	56.35 ± 1.11	69.91 ± 0.92	56.08 ± 0.88
TPP	93.94 ± 0.70	96.77 ± 0.32	55.12 ± 0.62	89.84 ± 0.45	97.02 ± 0.06	32.13 ± 0.15
TAAM	94.22 ± 1.09	97.00 ± 0.29	56.55 ± 0.88	93.54 ± 0.36	94.60 ± 0.11	43.28 ± 0.32
DyMoE	50.82 ± 1.31	61.33 ± 1.15	60.92 ± 1.13	23.88 ± 1.21	45.33 ± 1.97	42.92 ± 2.01
cascade	77.64 ± 0.65	85.15 ± 0.60	71.85 ± 0.85	71.03 ± 0.94	88.65 ± 0.39	73.55 ± 0.28
R_1	80.77 ± 0.61	86.89 ± 0.41	73.23 ± 0.60	47.67 ± 0.41	76.80 ± 0.21	67.50 ± 0.25
R_{prop}	80.75 ± 0.72	86.74 ± 0.49	70.64 ± 0.60	74.33 ± 0.24	89.95 ± 0.12	74.46 ± 0.23
R_4	81.16 ± 0.56	86.84 ± 0.47	71.29 ± 0.59	72.45 ± 0.34	89.63 ± 0.11	73.65 ± 0.18

Table 3: Cost of each protocol choice, in accuracy points; negative is a loss. Each block compares two settings under one aggregation. The batch block shows only TPP and TAAM, the two methods that compute statistics of their input batch before classifying (Section 3.1). For the other five and the three references, it is 0 on every dataset and seed. R_1 and R_4 keep the model they trained on the task subgraphs and on G, respectively, and are rescored under each setting. R_{prop} and R_4 run on the scoring graph (Section 4). Table 20 in Appendix N shows every block, including the inter-task edges from Mixed to HIDE.

Method	CoraFull	ogbn-arxiv	Cora	CiteSeer	Photo	Comp.	WikiCS	Roman
aggregation: AA → *per node, under Isolated*								
PDGNNs	$+4.88$	$+3.88$	$+0.05^{\dagger}$	$+2.89$	$-0.14^{\dagger}$	-2.11	-1.69	$-0.66^{\dagger}$
DeLoMe	$+3.81$	-13.44	$-0.50^{\dagger}$	$+2.15$	$+0.15^{\dagger}$	-1.88	$-0.14^{\dagger}$	-3.21
SSM	$+4.71$	$+13.55$	$-0.64^{\dagger}$	$+2.83$	$-1.38^{\dagger}$	$-5.57^{\dagger}$	$+0.88^{\dagger}$	-2.59
TPP	$+2.83$	$+7.18$	$+0.41^{\dagger}$	$+3.35$	$+1.68$	$+0.53^{\dagger}$	$+0.10^{\dagger}$	$-0.19^{\dagger}$
TAAM	$+2.78$	$+1.06$	$+0.37^{\dagger}$	$+2.19$	$+0.81$	$+0.55$	-0.60	-1.18
DyMoE	$+10.50$	$+21.45$	$+0.36^{\dagger}$	$+3.96$	$+1.27^{\dagger}$	$+13.53$	$+3.15$	$+6.11$
cascade	$+7.51$	$+17.62$	$+0.60$	$+2.78$	$+1.55$	$+0.15^{\dagger}$	$+1.04$	$+8.99$
R_1	$+6.12$	$+29.13$	$+0.42^{\dagger}$	$+2.88$	$+1.39^{\dagger}$	$+0.09^{\dagger}$	$+1.12$	$+9.06$
R_{prop}	$+6.00$	$+15.63$	$+0.32^{\dagger}$	$+2.70$	$+2.06$	$+0.37^{\dagger}$	$+0.98$	$+4.92$
R_4	$+5.68$	$+17.18$	$+0.39^{\dagger}$	$+2.83$	$+2.38$	$+0.42^{\dagger}$	$+1.16$	-1.47
batch: Isolated → Mixed								
TPP	-16.89	-49.00	-9.00	-7.48	-24.42	-26.53	-29.47	-16.26
TAAM	-36.46	-49.45	-18.22	-13.23	-11.28	-20.89	-23.60	-41.39
batch and edges: Isolated → HIDE, AA *throughout*								
PDGNNs	-13.14	-8.64	-4.35	-4.14	-4.15	-8.18	-9.69	-36.02
DeLoMe	-12.72	-4.76	-4.82	-4.32	-5.35	-11.96	-11.87	-31.27
SSM	-20.27	-14.01	-6.62	-4.82	-15.97	0	-30.47	-28.24
TPP	-50.27	-75.28	-14.75	-13.84	-43.67	-64.12	-63.59	-74.87
TAAM	-46.95	-64.11	-18.39	-14.71	-16.96	-38.98	-33.19	-59.30
DyMoE	-0.44	-1.74	$-0.87^{\dagger}$	$-0.31^{\dagger}$	$-0.34^{\dagger}$	-1.25	-7.46	-7.29
cascade	-16.31	-19.71	-7.06	-6.92	-2.39	-8.74	-11.99	-34.84
R_1	-15.96	-9.20	-4.99	-4.47	-5.28	-13.75	-14.56	-38.20
R_{prop}	-16.44	-20.88	-6.92	-7.78	-2.92	-9.03	-10.22	-17.02
R_4	-17.57	-21.22	-5.74	-6.79	-4.08	-11.56	-13.15	-39.86

TPP on ogbn-arxiv (last block of Table 20). What the settings change for the replay methods is less their mutual order than their task hit (Section 5.2) and their distance to the references (Section 5.4).

Dependence on graph structure. The cost of the inter-task edges grows with the fraction of a dataset's edges that connect different tasks, with Spearman rank correlations between -0.50 (SSM) and -0.83 (PDGNNs, TPP) across the eight datasets. Since this fraction correlates with edge homophily at -0.98, the two properties cannot be separated on these datasets.

The cascade. The cascade behaves similarly to the replay methods, unlike TPP: its batch cost is zero by definition, and the inter-task edges cost it 13.30 and 15.09 points on CoraFull and ogbn-arxiv, respectively, all in task-identification accuracy (Appendix M).

5.2 Where the accuracy is lost

Under HIDE, methods mostly fail at identifying the task. We measure the two steps of Section 2.1 separately. The *task hit* is task-identification accuracy: the fraction of test nodes whose predicted class belongs to their own task, regardless of being the correct class. The number of possible tasks vary per dataset: 35 on CoraFull, 20 on ogbn-arxiv, and 3 to 9 elsewhere. The *class given the true task* is the accuracy obtained when, for every test node, the argmax is restricted to the classes of the node's own task. Unlike an accuracy computed on correctly routed nodes, it does not depend on which nodes the model routes correctly. Table 22 in Appendix P reports both for CoraFull and ogbn-arxiv, and Table 23 reports the other six datasets.

Task identification is the bottleneck. The task hit under HIDE, by contrast, ranges from 56.3% to 75.4% on CoraFull and from 32.8% to 66.4% on ogbn-arxiv, and per-node accuracy under HIDE lies within 1.2 points of the task hit on CoraFull and within 4.1 points on ogbn-arxiv. Given the true task, every method classifies 88.8% to 97.7% of test nodes correctly on CoraFull and 85.5% to 97.1% on ogbn-arxiv. The loss from Isolated to HIDE is therefore incurred in the first step: TPP, the clearest case, routes only 56.3% and 32.8% of nodes to the correct task under HIDE yet classifies 96.9% and 97.1% correctly given the true task. The picture is the same on the own graph (Appendix E) and on the six other datasets (Table 23), where the class given the true task stays at or above 79.7% except for SSM on Amazon Computers (70.3%) and on Roman-empire (71.3% to 82.8%), where the task hit also falls furthest, to 17.6% for TPP.

Position, task size and forgetting. The loss is not concentrated on early tasks. The cost of the inter-task edges for the replay methods is higher on the first third of the sequence than on the last in 12 of 23 dataset–method cells and smaller in the other 11.

As tasks grow, the two steps move in opposite directions: retrained at up to fourteen classes per task on CoraFull and ten on ogbn-arxiv (Appendix Q), every method routes more nodes correctly and classifies fewer correctly within the task, which leaves accuracy nearly flat on CoraFull for the methods that do not use the batch but reorders the methods on ogbn-arxiv, where SSM falls from second to fifth and DyMoE rises from fourth to second between two and ten classes per task. A ranking in this setting is therefore specific to the task size at which it was obtained. Finally, forgetting (Zhang et al., 2022a) does not order the methods the way the loss from Isolated to HIDE does; TPP and the cascade forget nothing by construction, and what changes for them over the sequence is the task hit (Appendix S).

5.3 The graph choice

The loaders disagree about the graph (Section 3.2), and that disagreement alone shifts the results (Table 7 in Appendix E). On ogbn-arxiv, DeLoMe and SSM reach 44.58% and 44.56% on their own graphs, 0.02 points apart, but 34.59% and 56.08% on the shared graph, 21.49 points apart. Two controls attribute the shift to the graph itself: a directed variant of the shared graph, which keeps OGB's directed edges as the CGLB loader does, returns DeLoMe and SSM to 44.51% and 44.41%, and on CoraFull, substituting the TF-IDF features of the DGL loader into the shared graph returns PDGNNs, DeLoMe and SSM to within 0.05 points of their own-graph values.

No hyperparameter is returned between a method's two graphs, which keeps the comparison faithful to released work but confounds the shift with configurations tuned on the other graph. Retraining the three replay methods on the shared graph of ogbn-arxiv under three further configurations of the two hyperparameters their command lines expose, selected per seed on calibration nodes (Table 10 in Appendix F), changes PDGNNs by 0.13 points, leaves SSM at its released configuration, and lifts DeLoMe from 34.59% to 49.02% by setting its weight decay to zero, shrinking its gap to SSM from

21.49 to 7.06. Most of that gap therefore reflects DeLoMe's released weight decay interacting with the symmetrized graph, and the costs in Table 7 are costs at released configurations.

Inter-task edges during training. PDGNNs is the only method whose released code supports training with inter-task edges. Trained with them and evaluated under HIDE on its own graph, it only loses 2.76 points on CoraFull and 1.56 on ogbn-arxiv relative to its released configuration (Table 7). For the other three released methods, the switch yields no usable measurement (Appendix G).

5.4 Comparison with the reference classifiers

Only two of the six methods exceed R_{prop} anywhere, and only on three of the eight datasets (Table 2; every difference in Appendix T). On CoraFull, PDGNNs is above all three references on every seed, by 1.23 over R_1, 3.79 over R_{prop} and 3.16 over R_4; DeLoMe is level with R_{prop} and R_4 and below R_1; the other four are below all three. On ogbn-arxiv, all six methods are below all three references, by 4.39 to 35.34 points against R_1 and by 11.38 to 42.33 against R_{prop}. Under Isolated, where the references also read the task subgraph and every label R_{prop} propagates from belongs to the node's own task, TPP and TAAM exceed R_{prop} by 13 to 19 points on both datasets. Under Mixed, no method exceeds it, and under HIDE only PDGNNs does on every seed, on CoraFull.

The cascade against the references. The cascade is 1.20 points above R_{prop} on CoraFull and 0.91 below it on ogbn-arxiv, level with R_4 on both, while using 125 times as many parameters.

The two references. R_1, the reconstructed joint baseline, falls below R_{prop} on five of the eight datasets, by 1.62 to 16.28 points, and is ahead by 2.34 to 3.15 on the other three, every seed agreeing on the sign (Appendix L). Given the same information and refinement, R_4 lies within 1.51 points of R_{prop} on every dataset but Roman-empire, where $\delta = \text{Acc}(R_{\text{prop}}) - \text{Acc}(R_4) = +29.39$. A method is therefore read against both.

The six other datasets. On the six other datasets (Appendix N) PDGNNs, the best method overall, exceeds R_{prop} only on Cora and CiteSeer, by 2.52 and 3.43 points, and falls 2.62 to 10.85 points below it on the other four; DeLoMe is above it on CiteSeer by 2.55 and on Cora by 0.47, the latter with the seeds not all agreeing On Roman-empire, where the two references are 29.39 points apart (Section 5.4), five of the six methods lie above R_4 and below R_{prop}, and TPP is below both. Appendix T also compares each replay method with R_{prop} fitted on only as many labels as its buffer holds.

6 Conclusion

We introduced HIDE, a harder benchmarking protocol for GCIL that withholds the task identity during inference and that allows the evaluation of both task identification and classification separately. Our experiments show that previous methods scored under CGLB lose performance when evaluated under HIDE protocol, PDGNN for instance was the SOTA under CGLB protocol, however its accuracy drops more than 15 percentage points under HIDE benchmark, PDGNN loses both to reference classifiers and to a simpler cascade baseline.

These results suggest that a GCIL accuracy should be reported together with the evaluation setting that produced it, the graph it was measured on, and a reference classifier with no incremental mechanism. To support further research, we release the shared graphs, splits and task assignments, the reference classifiers. Our work shows that the current bottleneck in GCIL is the capacity to accurately identify a node´s task from a neighborhood that potentially spans every known task.

AI use statement

We used generative AI tools to implement methods, including the reimplementations of TAAM and DyMoE, to edit text, to write code and to format references; we did not use them to generate data or to formulate or prove claims. Every number and table is read from a log produced by the released code, the reimplementations are checked against their published numbers (Appendix H), and every citation was checked by the authors.

Ethics statement

This work re-evaluates published methods on public benchmark graph datasets that are distributed for research and contain no personally identifying information, and it collects no new data. Its contribution is an evaluation protocol and a measurement of how much reported accuracy depends on protocol choices. Every method is cited and run from its released code or, for TAAM and DyMoE, reimplemented from its paper, and every departure from a released configuration is declared in Appendix H and Appendix F. The authors declare no conflicts of interest.

Reproducibility statement

Section 3 specifies the protocol in full: the shared graph, the per-class $60/20/20$ split, the assignment of classes to tasks, and the three settings. Appendix B reports the run-to-run variation of the pipeline, Appendix E the differences between the graphs each released loader builds, Appendix C the hardware and software environments, and Appendix K the configuration of the reference classifiers and the search over it. Every method with released code runs at its released hyperparameters, and TAAM and DyMoE at those of their papers, over ten seeds except where Appendix B lists fewer, with each departure declared in Appendix H, where the reimplementations of TAAM and DyMoE are also checked against their published numbers.

References

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

Ziyang Cheng, Zhixun Li, Yuhan Li, Yixin Song, Kangyi Zhao, Dawei Cheng, Jia Li, Hong Cheng, and Jeffrey Xu Yu. Can LLMs alleviate catastrophic forgetting in graph continual learning? A systematic study, 2025. URL https://arxiv.org/abs/2505.18697.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales 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, 2022. doi: 10.1109/TPAMI.2021.3057446.

Giovanni Donghi, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicolò Navarin. Online continual graph learning. *arXiv preprint arXiv:2508.03283*, 2025.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HygDF6NFPB.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning, 2018. URL https://arxiv.org/abs/1805.09733.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1903.02428.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8E1-f3VhX1o.

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5065–5079, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

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. doi: 10.1145/3702648.

Lecheng Kong, Theodore Vasiloudis, Seongjun Yun, Han Xie, and Xiang Song. Dynamic mixture-of-experts for incremental graph learning, 2025. URL https://arxiv.org/abs/2508.09974.

Jingtao Liu and Xinming Zhang. TAAM: Inductive graph-class incremental learning with task-aware adaptive modulation. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, 2026. URL https://arxiv.org/abs/2602.08036.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023. doi: 10.1109/TPAMI.2022.3213473.

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-based benchmark for graph neural networks. In *International Conference on Machine Learning Workshop on Graph Representation Learning and Beyond*, 2020. URL https://arxiv.org/abs/2007.02901.

Chaoxi Niu, Guansong Pang, and Ling Chen. Graph continual learning with debiased lossless memory replay. In *ECAI: European Conference on Artificial Intelligence*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 1808–1815. IOS Press, 2024a. doi: 10.3233/FAIA240692.

Chaoxi Niu, Guansong Pang, Ling Chen, and Bing Liu. Replay-and-forget-free graph class-incremental learning: A task profiling and prompting approach. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/a07e87ecfa8a651d62257571669b0150-Abstract-Conference.html.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pp. 524–540. Springer, 2020. doi: 10.1007/978-3-030-58536-5_31.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008. doi: 10.1609/aimag.v29i3.2157.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS Workshop on Relational Representation Learning*, 2018. URL https://arxiv.org/abs/1811.05868.

Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. Towards robust graph incremental learning on evolving graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32728–32748, 2023. URL https://proceedings.mlr.press/v202/su23a.html.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

Felix Wu, Amauri H. Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 2019.

Xikun Zhang, Dongjin Song, and Dacheng Tao. CGLB: Benchmark tasks for continual graph learning. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*, volume 35, 2022a. URL https://openreview.net/forum?id=5wNiiIDynDF.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Sparsified subgraph memory for continual graph representation learning. In *IEEE International Conference on Data Mining (ICDM)*, pp. 1335–1340, 2022b. doi: 10.1109/ICDM54844.2022.00177.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4): 4622–4636, 2023. doi: 10.1109/TPAMI.2022.3186909.

Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. Topology-aware embedding memory for continual learning on expanding networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4326–4337, 2024. doi: 10.1145/3637528.3671732.

Tianqi Zhao, Alan Hanjalic, and Megha Khosla. AGALE: A graph-aware continual learning evaluation framework. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=xDTKRLyaNN.

A LIMITATIONS

Coverage. We evaluate six methods on eight datasets in the node-level class-incremental scenario only, with two classes per task in class-index order; task size is varied over four further sizes on CoraFull and ogbn-arxiv (Appendix Q), and the class-to-task assignment is never changed. Reddit and ogbn-products, the two remaining CGLB datasets, enter none of the ranges and rankings the body states over the six methods, although the reference analyses of Appendices L and K include both: on Reddit the methods are measured on seeds 0 to 4 rather than ten (Table 21), and on ogbn-products TPP cannot run at its released configuration (Appendix H) and no other method has ten seeds. Both fall inside the range of edge homophily and cross-task edge fraction covered by the eight (Table 6), so their absence removes only the two largest graphs.

Reconstructed methods. TAAM and DyMoE are reconstructions written from their papers: TAAM's repository lacks the training entry point and hyperparameters, while DyMoE has no released code. Each is checked against its published numbers on the graph encoding its paper opens and under its paper's metric, but on the partition, seeds and class-index task order of this work in place of the published splits and task sequence, so the checks hold the original encoding and metric and not the published protocol (Appendix H). Under that check TAAM exceeds its published number on both datasets where the check is possible, while DyMoE falls 5.83 to 8.75 points short on all three at the memory fraction its paper declares per dataset, 0.05 and 0.01 on ogbn-arxiv, and crosses the published CoraFull number only at two to four times that fraction. Their numbers describe our implementations and are read as reconstructions; ranges over the six methods include them, statements restricted to released work do not, and no ranking stated in the abstract includes them.

Run-to-run variation of DyMoE. DyMoE is the one method whose training does not repeat at a fixed seed: two runs at the same seed on the same machine differ by up to 1.2 points on Cora and 1.6 on ogbn-products when the CPU reductions are multi-threaded, and agree bit for bit on a single thread at a different accuracy, while TAAM agrees bit for bit under both. Its reported standard deviation therefore includes execution noise on top of the variation across seeds, and no paired per-seed claim in this paper is made against DyMoE.

Unmeasured protocol choices. Every method runs at its released configuration, and the cost of that choice is measured only for the three replay methods on ogbn-arxiv (Section 5.3), although De Lange et al. (2022) argue that hyperparameter selection is itself part of the protocol. The cascade, by contrast, selects its two hyperparameters on calibration nodes, so its comparison with the methods also carries that difference. Every result also rests on partitions drawn at one proportion, $60/20/20$ per class and one per seed (Section 3.2), which replace the publisher's and CGLB's splits, even though Shchur et al. (2018) showed that rankings on these datasets change with the split. The partition varies across seeds, but its proportions do not. The cost of the inter-task edges is measured for models trained without them. Only PDGNNs is also trained with them (Section 5.3), so for the other methods the cost combines the difficulty the edges add with the change from the training graph to the scoring graph.

References and resolution. The references are trained at one fixed configuration. The finding about joint training remains a statement about the grid searched (Appendix K), and R_1 is our own reconstruction of the joint baseline. A reference retrained on the GPU does not repeat at a fixed seed, so its accuracy drifts between trainings, by less than 0.7 points in a ten-seed mean once the two collapsed seeds of R_1 are set aside (Appendix N), and each table names the training it reads. Every difference rests on at most ten paired seeds, which support a direction and an interval rather than a significance claim.

B REPRODUCIBILITY

Every split, initialization and subsample is drawn from the seed its run is named by, the integers 0 to 9 throughout, with these exceptions, each stated where its numbers appear: the six CGLB baselines and plain fine-tuning of Appendix J, TAAM and DyMoE under their own graph and in their parity checks, every method on Reddit, the homophily sweep of Appendix R, and the task sizes of Appendix Q other than two classes per task exist on seeds 0 to 4 or on a subset of them, TWP on ogbn-arxiv on four. The checks of TAAM and DyMoE against their published numbers

(Appendix H) restore the graph encoding each paper opens and its metric, and keep the partition and the class-index task order of Section 3.2 at seeds 0 to 4, so they are checks under the original encoding and metric on our splits and not reproductions under the published splits. Repeating a run at a fixed seed does not repeat its accuracy: retraining the three replay methods on CoraFull at seed 4 moved their accuracy by between 0.08 and 0.16 points, two trainings of a reference on the GPU differ by less than 0.7 points in a ten-seed mean, two collapsed seeds of R_1 set aside (Appendix N), and two runs of DyMoE differ by up to 1.6 points on ogbn-products (Appendix A), so every difference is reported with its seeds and its interval. Section 3.2 gives the node partition, Appendix K the configuration of the references and the cascade, Appendix C the hardware, the three software environments and the licenses, and Appendix V the seeds agreeing and the bootstrap interval of every difference between the references. A reported mean can therefore be recomputed. The protocol logs each number was read from, the frozen per-seed tables and the code that produces every table in this paper are part of the release accompanying it, together with a datasheet recording provenance, composition and intended use, and a description of what adding a method, a dataset or a metric requires. Every table here is produced by the per-experiment scripts. The configurable entrypoint the release also carries does not yet reproduce any number in this paper, and the release says so.

C Compute, software and artifacts

Every run reported here was carried out on a single NVIDIA RTX A6000 with 48 GB of memory, one of six on a host carrying an Intel Xeon Silver 4314 and 436 GB of system memory. The released implementations keep no record of their run time, so the total compute is not reported.

Three software environments are involved, since the released implementations evaluated here require library versions that cannot coexist in one environment. Table 4 gives them.

Table 4: The three environments, and what each one produces. The first is the one this work releases; the other two are the ones the released implementations require, and are the environments in which their numbers were obtained. TAAM and DyMoE are written inside the TPP code base and run in its environment.

used for	Python	PyTorch	graph library
the protocol, the references, the cascade	3.12.13	2.12.0, CUDA 13.0	PyTorch Geometric 2.7.0
PDGNNs, DeLoMe, SSM	3.8.19	1.13.1, CUDA 11.7	DGL 0.9.1
TPP, TAAM, DyMoE	3.8.19	1.13.1, CUDA 11.7	DGL 1.0.1

The datasets are the public ones distributed by PyTorch Geometric (Fey & Lenssen, 2019) and by the Open Graph Benchmark (Hu et al., 2020), obtained through those loaders and cited individually in the caption of Table 6. What this work supplies to a method is the shared graph of Section 3.2 built from them. The data is not redistributed.

The code released with this work is under the MIT license, as is the released implementation of TPP. The CGLB code base, from which the implementations of PDGNNs, DeLoMe and SSM are run, is under Creative Commons Attribution-NonCommercial 4.0. The repositories of those three methods carry no license file of their own, so each is run in place from its own source, and none of them is redistributed here. The release carries only the patches applied to them.

Because no repository is redistributed, a repository that moves takes the numbers measured from it with it. Each one is therefore cloned at a pinned commit, and Table 5 gives the five.

Table 5: The five third-party repositories, each at the commit the numbers reported here were produced from. Commits are abbreviated to their first ten characters; the full hashes, and the patches applied on top of them, are in the setup script released with this work.

repository	clone URL	pinned commit
CGLB	`https://github.com/QueuQ/CGLB.git`	`0c72c93c0a`
SSM	`https://github.com/QueuQ/SSM.git`	`4ea59ea036`
DeLoMe	`https://github.com/mala-lab/DeLoMe.git`	`4f6402d702`
pdgnns	`https://github.com/imZHANGxikun/pdgnns.git`	`04624540af`
TPP	`https://github.com/mala-lab/TPP.git`	`1048d6d634`

D THE DATASETS

Table 6 gives the ten datasets measured in this work, the eight on which the body compares the methods and the two, Reddit and ogbn-products, that enter no range or ranking of the methods (Appendix A). The partition of Section 3.2 replaces the one each dataset's publisher distributes, so no number here is comparable to a published leaderboard entry on the same graph. Classes are taken two per task in class-index order, and a trailing odd class is dropped where a dataset has an odd number of them: ogbn-products (Hu et al., 2020) provides 47 classes and 46 are used, Reddit (Hamilton et al., 2017) provides 41 and 40 are used, and Cora (Sen et al., 2008) provides 7 and 6 are used. The nodes of a dropped class keep their features and their edges and carry no label, so they belong to no task: every training graph, with or without the inter-task edges, and every task subgraph of the Isolated and Mixed readings is induced on the labelled nodes of the classes it holds, which leaves them out, and every scored set is drawn per labelled class, which never scores them; on the full graph of HIDE they are present as unlabelled context, where they propagate and are never scored.

Table 6: The ten datasets as the protocol of Section 3.2 builds them: Cora and CiteSeer (Sen et al., 2008), Amazon Photo and Amazon Computers (Shchur et al., 2018), WikiCS (Mernyei & Cangea, 2020), CoraFull (Bojchevski & Günnemann, 2018), Roman-empire (Platonov et al., 2023), ogbn-arxiv and ogbn-products (Hu et al., 2020) and Reddit (Hamilton et al., 2017). *Classes* counts those in scope once the trailing odd class is dropped and *tasks* the two-class tasks they make; *homophily* is the share of edges joining two nodes of the same class, and *crossing* the share whose endpoints belong to different tasks, which is what a scoring graph built from the task subgraphs removes. Edges are counted in both directions. The last two rows enter no range or ranking of the methods, for the reasons Appendix A gives.

dataset	nodes	edges	classes	tasks	homophily	crossing
CoraFull	19,793	126,842	70	35	0.567	0.423
ogbn-arxiv	169,343	2,315,598	40	20	0.654	0.334
Cora	2,708	10,556	6	3	0.810	0.153
CiteSeer	3,327	9,104	6	3	0.736	0.182
Amazon Photo	7,650	238,162	8	4	0.827	0.129
Amazon Computers	13,752	491,722	10	5	0.777	0.216
WikiCS	11,701	431,726	10	5	0.655	0.249
Roman-empire	22,662	65,854	18	9	0.047	0.892
Reddit	232,965	114,615,892	40	20	0.756	0.235
ogbn-products	2,449,029	123,718,152	46	23	0.808	0.187

E THE TWO GRAPHS

Table 7 gives the cost of the graph choice under HIDE and of the inter-task edges during training. Table 8 gives the accuracy of the six methods under Isolated, Mixed, and HIDE on each method's own graph, and the steps of Table 20 on CoraFull and ogbn-arxiv under that graph, beside the shared graph of Section 3.2. The references are measured under the shared graph alone, so the interval they span is not tested under the graphs the rest of this literature reports on.

Table 7: Graph choice under HIDE (shared minus own graph, first four rows) and inter-task edges during training (PDGNNs with minus without, own graph), at released configurations, as paired means over seeds 0 to 9 in percentage points; a dagger marks a cell whose seeds do not all agree with its sign. TAAM and DyMoE, whose own-graph runs cover seeds 0 to 4, are given in Table 32.

difference	CoraFull	ogbn-arxiv
PDGNNs	$+1.59$	$+15.89$
DeLoMe	-1.53	-9.99
SSM	$-0.42^{\dagger}$	$+11.52$
TPP	-4.59	-1.06
PDGNNs, edges at training	-2.76	-1.56

The own graphs differ from the shared one in two respects: on ogbn-arxiv the CGLB loader keeps the directed edges OGB distributes, where the shared graph is undirected, and on CoraFull the DGL loader supplies TF-IDF features in place of the publisher's binary ones. Undoing either difference on the shared graph returns PDGNNs, DeLoMe and SSM to within 0.15 points of their own-graph accuracy under HIDE (Section 5.3).

Table 8: The six methods on their own graph, the graph each released loader builds (TPP's for TAAM and DyMoE), under Isolated, Mixed and HIDE, in the form of Table 21; mean and standard deviation over seeds 0 to 9, and a count in parentheses marks a cell read on fewer seeds (TAAM and DyMoE, our reconstructions, were run on the own graph for seeds 0 to 4). A dash marks a reading the module-per-task pipeline of TPP, TAAM and DyMoE does not produce on the own graph: the task-pure batch read per node. The last two columns are the task hit and the class given the true task under HIDE, per node; for TPP they come from a second scoring pass of the same checkpoints, whose HIDE accuracy differs from the column beside it by 0.01 on CoraFull and 0.19 on ogbn-arxiv. TPP's Mixed and HIDE readings apply its task estimate to each node's own features, our change to its released code. * marks TAAM and DyMoE, our reconstructions (Appendix H).

	Isolated		Mixed	HIDE		
method	AA	per node	per node	per node	task hit	class | true task
CoraFull						
PDGNNs	81.56 ± 0.67	85.04 ± 0.54	85.04 ± 0.54	72.85 ± 0.81	73.75 ± 0.74	97.90 ± 0.21
DeLoMe	81.76 ± 0.69	85.75 ± 0.54	85.75 ± 0.54	72.89 ± 0.84	73.85 ± 0.77	97.68 ± 0.23
SSM	77.78 ± 1.32	82.37 ± 0.81	82.37 ± 0.81	63.58 ± 0.74	64.53 ± 0.63	96.69 ± 0.29
TPP	93.10 ± 0.70	—	83.98 ± 0.64	59.71 ± 0.70	60.83 ± 0.70	96.58 ± 0.38
TAAM*	$95.45 \pm 0.62^{(5)}$	—	$62.97 \pm 0.36^{(5)}$	$59.13 \pm 0.44^{(5)}$	$60.20 \pm 0.37^{(5)}$	$97.28 \pm 0.24^{(5)}$
DyMoE*	$30.76 \pm 0.71^{(5)}$	—	$35.78 \pm 1.16^{(5)}$	$32.12 \pm 1.23^{(5)}$	$32.60 \pm 1.28^{(5)}$	$90.87 \pm 0.76^{(5)}$
ogbn-arxiv						
PDGNNs	52.20 ± 0.58	49.12 ± 0.32	49.12 ± 0.32	47.19 ± 0.31	51.16 ± 0.33	87.99 ± 0.28
DeLoMe	48.72 ± 0.67	47.26 ± 0.53	47.26 ± 0.53	44.58 ± 0.47	48.80 ± 0.51	86.60 ± 0.48
SSM	44.62 ± 0.65	47.44 ± 0.91	47.44 ± 0.91	44.56 ± 0.73	48.11 ± 0.65	88.44 ± 0.59
TPP	86.40 ± 0.40	—	73.75 ± 0.22	33.19 ± 0.32	34.24 ± 0.27	95.60 ± 0.09
TAAM*	$90.37 \pm 0.21^{(5)}$	—	$33.70 \pm 0.14^{(5)}$	$32.84 \pm 0.21^{(5)}$	$34.71 \pm 0.23^{(5)}$	$92.24 \pm 0.39^{(5)}$
DyMoE*	$26.39 \pm 0.75^{(5)}$	—	$43.04 \pm 0.79^{(5)}$	$42.13 \pm 0.70^{(5)}$	$43.59 \pm 0.73^{(5)}$	$89.10 \pm 0.43^{(5)}$

Which protocol step moves a method further on ogbn-arxiv reverses with the graph choice for two of the three methods whose own-graph runs carry both readings. Under its own graph, PDGNNs moves 3.07 points with the aggregation and 1.94 with the edges, and under the shared graph 3.88 and 6.40, so the larger of the two changes hands; DeLoMe reverses the other way, 1.46 against 2.68 under its own graph and 13.44 against 4.78 under the shared one; SSM does not reverse, the edges carrying more under both. The inter-task edges cost TPP 40.57 points under its own graph and 15.88 under the shared one, but its own-graph runs do not carry the reading that separates the aggregation from the batch, so the pair cannot be compared for it. On CoraFull the edges carry more than the aggregation for every method under each graph on which both readings exist, except DyMoE under the shared graph, which the aggregation moves 10.50 points and the edges 0.41, so the reversal is a result about ogbn-arxiv alone. The task hit and the class given the task under the own graph

follow the shared graph: given its own task, the class is right in 90.9% to 97.9% of the test nodes on CoraFull and 86.6% to 95.6% on ogbn-arxiv, and the task hit under HIDE runs from 32.6% to 73.9% and from 34.2% to 51.2%, the two reconstructions over five seeds and the other four over ten.

Table 9: The change of aggregation and the cost of the inter-task edges, under the own graph and under the shared graph, as paired differences over seeds 0 to 9; no cell subtracts one average from the other. A count in parentheses marks a cell measured on fewer seeds; a dash marks a reading the own-graph runs do not carry, so the aggregation cannot be separated from the batch there for TPP, TAAM and DyMoE.

	own graph		shared graph	
protocol step	CoraFull	ogbn-arxiv	CoraFull	ogbn-arxiv
aggregation, PDGNNs	$+3.47$	-3.07	$+4.88$	$+3.88$
aggregation, DeLoMe	$+3.99$	-1.46	$+3.81$	-13.44
aggregation, SSM	$+4.59$	$+2.82$	$+4.71$	$+13.55$
aggregation, TPP	—	—	$+2.83$	$+7.18$
aggregation, TAAM*	—	—	$+2.78$	$+1.06$
aggregation, DyMoE*	—	—	$+10.50$	$+21.45$
inter-task edges, PDGNNs	-12.19	-1.94	-11.84	-6.40
inter-task edges, DeLoMe	-12.86	-2.68	-12.78	-4.78
inter-task edges, SSM	-18.79	-2.88	-18.06	-13.83
inter-task edges, TPP	-24.28	-40.57	-24.76	-15.88
inter-task edges, TAAM*	$-3.84^{(5)}$	$-0.86^{(5)}$	-3.99	-1.87
inter-task edges, DyMoE*	$-3.66^{(5)}$	$-0.91^{(5)}$	-0.41	-2.41

Table 2 in Section 5 gives the accuracies under the shared graph those differences are taken between.

F Replay methods reselected on calibration

Table 10 gives the three replay methods on ogbn-arxiv at their released configuration and at the one selected per seed on calibration, the retuning Section 5.3 reads.

Table 10: Replay methods on ogbn-arxiv, shared graph, per-node accuracy under HIDE at the released configuration and at the one selected per seed on calibration, among the released pair (lr 0.005, wd 5×10^{-4}), lr 0.001, lr 0.02 and wd 0. *Own graph* gives the released configuration under the method's own graph, as in Table 8.

method	own graph	released	chosen on calibration	chosen − released	pairs chosen
PDGNNs	47.19 ± 0.31	63.08 ± 0.24	63.21 ± 0.28	$+0.13$	released 6 seeds, lr 0.02 3, wd 0 1
DeLoMe	44.58 ± 0.47	34.59 ± 2.22	49.02 ± 2.11	$+14.43$	wd 0 on every seed
SSM	44.56 ± 0.73	56.08 ± 0.88	56.08 ± 0.88	$+0.00$	released on every seed

G Inter-task edges during training, method by method

The code of CGLB (Zhang et al., 2022a), which the four methods fork, exposes a flag for training with the inter-task edges. The flag selects the directory the data preparation writes the task subgraphs to; which directory the training pipeline reads is fixed in the pipeline itself. Of the four forks, only PDGNNs kept the pipelines that read the directory with the edges, so it is the one method the flag reaches as released. The forks of SSM and DeLoMe kept only pipelines that read the edge-free directory, and TPP selects the edge-free pipeline regardless of the flag. We restored the pipeline of the CGLB trunk in SSM and DeLoMe and re-ran both: SSM trains to the end, and DeLoMe fails in its condensation step, which assumes the class structure of the isolated subgraph. Neither is

reported, under the criterion that admits the reconstructions of TAAM and DyMoE (Appendix H): a result of code we wrote is reported once it is checked against a number its authors published under a stated check and is labelled as a reconstruction, and the SSM run trained with the edges was checked against none. The two patches are in the released code so that a reader can reproduce both outcomes.

H WHAT EACH RELEASED IMPLEMENTATION PERMITS

This work measures released implementations, changed only where scoring the scenario, fitting the hardware, running on the shared graph, or the task sizes and training edges of Appendices Q and G require it, with every patch released (Table 5): a method whose released code does not score the class-incremental scenario is not run as released, and Table 11 says why, per repository and commit, so that the reason can be checked. The two hyperparameters retuned in Section 5.3 are the two the released command lines expose, the learning rate and the weight decay; no line of code changes between a released run and a retuned one.

Two of the architecture-based methods of Table 11 have no release that scores the scenario and are written here from their papers: TAAM, whose repository holds the model classes and a pipeline but not the training entry point its own script calls and not the hyperparameters, and DyMoE, for which no code is released. Both are reported as reconstructions, and a number either produces describes the implementation written here.

A reconstruction is held to the numbers its paper publishes before it is read as a baseline, and the check restores the graph encoding each paper opens and its metric while keeping the partition, the seeds and the class-index task order of Section 3.2, so it is a check under the original encoding and metric on our splits and not a reproduction under the published protocol. TAAM publishes average accuracy under Isolated with two classes per task, and the reconstruction reaches $95.45 \pm 0.62\%$ on CoraFull against 92.5% published and $90.37 \pm 0.21\%$ on ogbn-arxiv against 88.6%, over seeds 0 to 4; the four other datasets TAAM publishes, CiteSeer, Photo, WikiCS and Computers, have no path through the released loader for their original encoding and are measured here only on the graph the benchmark supplies, so on them no check is made. DyMoE publishes the mean of the diagonal of the accuracy matrix on the growing graph, with five classes per block, on three datasets, and at the memory fraction its paper declares per dataset, in its Table 5, $p = 0.05$ and $p = 0.01$ on ogbn-arxiv and ogbn-products, the reconstruction stands below all three over five seeds each: $70.76 \pm 0.71\%$ against 76.59% on CoraFull, $59.58 \pm 1.75\%$ against 68.33% on ogbn-arxiv and $84.28 \pm 1.35\%$ against 92.84% on Reddit, a gap of 5.83 to 8.75 points in the same direction.

The memory fraction is the knob that closes the gap on CoraFull. The reconstruction crosses the published number between $p = 0.1$, at $75.31 \pm 0.68\%$, and $p = 0.2$, at $78.96 \pm 0.89\%$, and reaches $80.60 \pm 0.69\%$ at $p = 0.3$ and $80.84 \pm 0.97\%$ at $p = 0.4$, over five seeds each, while no other knob the paper declares moves the number by more than 0.4 points in the favourable direction, within one seed standard deviation: at five classes per block over five seeds, against the default $70.76 \pm 0.71\%$, $\gamma = 0.1$ gives $69.09 \pm 0.46\%$ and $\gamma = 5$ gives $70.22 \pm 0.45\%$, a hidden width of 256 gives $71.13 \pm 0.65\%$, a learning rate of 10^{-3} gives $71.04 \pm 0.78\%$, 30 balancing epochs give $68.55 \pm 0.55\%$, a weight decay of 10^{-2} gives $61.88 \pm 1.42\%$, and the paper's learning rate of 10^{-4} under its epoch budget gives $39.62 \pm 0.96\%$, whereas the memory fraction moves it by 4.6 points at $p = 0.1$ and 8.2 at $p = 0.2$. The tables of this paper hold DyMoE at the declared fractions. Where the paper leaves a choice open or does not train full-batch, the reconstruction departs from it: it uses one layer, which the paper does not state and which was chosen because it lies closer to the published CoraFull number than two layers, 70.76% against $64.36 \pm 1.05\%$, a choice made on the reproduction target and never on a number of this protocol; and a learning rate of 0.005 for 200 full-batch epochs in place of 40 epochs of minibatches of 128 at 10^{-4}. The DyMoE rows therefore carry a reconstruction that stands six to nine points below its published headlines, and the memory fraction that closes the gap on CoraFull lies between two and four times the declared one.

ogbn-products, at 2.4 million nodes the largest dataset of Table 6, is beyond the released pipeline of TPP and within reach of TAAM and DyMoE, built here on the same code base. The released scripts of TPP cover CoraFull and ogbn-arxiv and select the full-batch pipeline, on which the contrastive pretraining of the first task builds a similarity matrix over every node of that task's subgraph; on ogbn-products that subgraph holds about 224 thousand nodes and the matrix requests 187 GiB in

a single allocation, which is what its own objective costs at that size. TAAM and DyMoE, on the same 48 GB card, run through the dataset at one layer and full batch on the four seeds that carry a partition; no method of the six is nevertheless read on ogbn-products in a range or a ranking, for the reason Appendix A gives.

Table 11: The architecture-based methods of Section 2.2, which add a component per task and freeze it, and what their released code permits. TPP is scored from its released code, with the per-node task estimate we added for Mixed and HIDE; TAAM and DyMoE are scored as reconstructions written from their papers, and HPN is not scored.

method	what the released code does	what scoring it under HIDE would need
TPP (Niu et al., 2024b)	repository `mala-lab/TPP`: the class-incremental pipeline of CGLB, the task estimated from the batch by a prototype profile	a task estimated per node instead of per batch (Section 3.1), which we added to its released code
HPN (Zhang et al., 2023)	repository `QueuQ/HPNs` at `21c431f`: one classifier per task, sized to that task's classes, and labels restricted to those classes in the loader, so a prediction ranges over the classes of the node's own task, the task-incremental scenario	a classifier over every class seen, which is not the released method
TAAM (Liu & Zhang, 2026)	repository `1iuJT/TAAM_AAMAS2026`: model classes and a pipeline, without the training entry point its script calls and without hyperparameters, and its optimizer never receives the classifier	a runner and a configuration, written here from the paper; scored as a reconstruction and not as released work
DyMoE (Kong et al., 2025)	no code released: the repository the paper cites does not resolve	the method itself, written here from the paper; scored as a reconstruction and not as released work

I WHAT EACH METHOD RETAINS

Unlike class-incremental learning on images, where rehearsal methods are compared under a common memory budget (Rebuffi et al., 2017; Prabhu et al., 2020; Masana et al., 2023), the GCIL methods evaluated here each set their own storage; TAAM and DyMoE, reconstructed here, are not counted. Table 12 counts what survives the last task, so that the differences of Section 5 can be read with the memory budget each method was allowed beside them.

The cascade is counted on the same terms, and the zero in the label column of its streaming tier counts stored class ids and nothing more. What that tier stores is one subspace basis per task, the right singular vectors of the diffused features of that task's calibration nodes (Appendix M); the 3,932 vectors are the rows of those bases, and their grouping by task records the task each calibration node belonged to. The tier therefore retains supervision in the form of task membership, and a zero in that column is read as the absence of class ids, which is what the column counts.

The label-memory tier holds the labels R_{prop} holds, the same 3,932 vectors against none, and 125 times the parameters, the last of these coming from one frozen network per task. That parameter count is the streaming tier's plus R_{prop}'s, since the tier carries the reference whole. What the tier reaches at a parameter count matched to R_{prop} is not measured here, so its 1.20 points above R_{prop} on CoraFull (Appendix M) are reported at the parameter count of its own row.

Equal counts in Table 12 are not equal bytes. The length of a vector is the dimension of what the method stores, a node feature, a basis row, a prototype or an embedding, which the methods do not share, and the three columns leave out state that is none of the three, such as the standardization statistics of a linear head or the per-task calibration counts of the transductive tier. The table orders the methods by what they keep, and a byte count would have to be read from the dimensions beside it.

Table 12: What survives the last task on CoraFull, counted. Vectors are stored feature or embedding vectors, labels are retained ground-truth labels, and parameters are the weights carried at inference. Each third-party memory budget is the one its own authors set. The cascade holds one frozen network per task, which is where its parameter count comes from, and its label-memory and transductive tiers retain the labels R_{prop} retains while exceeding it in the other two columns. R_3 and R_4 are fitted on the training and calibration labels together, which R_1 and R_2 are not. TAAM and DyMoE, reconstructed here, are not counted.

method	vectors	labels	parameters
TPP	35	0	2,900,980
DeLoMe	3,949	3,949	2,248,006
SSM	5,910	5,910	2,248,006
PDGNNs	11,511	11,511	2,248,006
cascade, streaming	3,932	0	78,068,550
cascade, label memory	3,932	15,781	78,695,740
cascade, transductive	3,932	15,781	78,695,775
R_{prop}	0	15,781	627,190
R_1, R_2	11,849	11,849	2,248,006
R_3, R_4	11,849	15,781	2,248,006

J THE CGLB BASELINES AND PLAIN FINE-TUNING

Seven further entries are reported under their own graph, six of them methods and the seventh plain fine-tuning (Table 13); their runs under the shared graph, over seeds 0 to 4, are released with the logs and enter no table. Read per node, all but ER-GNN on ogbn-arxiv and TWP on CoraFull have collapsed to the last task before any protocol choice of this paper is applied. On ogbn-arxiv six of the seven already score between 2.03% and 2.05% under Isolated, the share of the test nodes the two classes of the last task hold, and the inter-task edges move them by at most 0.02 points; ER-GNN is the exception, at 34.37% and 31.94%. On CoraFull all but TWP begin between 0.23% and 4.40%, against a uniform chance of 1.4%, and the edges take them to between 0.22% and 2.17%; TWP begins at 17.80% and ends at 10.85%. These rows show a floor: the task-averaged AA of the first column reads 4.84% to 4.89% on ogbn-arxiv for models that are answering with one task's classes, because a task of two classes answered perfectly counts as much as the nineteen answered not at all.

Table 13: The six baselines CGLB ships and plain fine-tuning under Isolated and HIDE on the own graph, Isolated read under both averages so that the pair of per-node columns is one metric from end to end; mean and standard deviation over seeds 0 to 4 (TWP on ogbn-arxiv over four). None of the seven reads its batch, so their Mixed accuracy is the Isolated per-node column.

	Isolated		HIDE
method	AA	per node	per node
CoraFull			
TWP	15.40 ± 3.78	17.80 ± 5.13	10.85 ± 3.91
GEM	8.92 ± 1.68	4.40 ± 1.66	2.17 ± 0.98
EWC	5.47 ± 2.74	2.87 ± 2.60	1.31 ± 1.13
MAS	2.95 ± 0.39	0.42 ± 0.15	0.33 ± 0.06
LwF	2.83 ± 0.13	0.65 ± 0.83	0.46 ± 0.45
ER-GNN	2.51 ± 0.25	0.28 ± 0.06	0.24 ± 0.02
fine-tuning	2.44 ± 0.36	0.23 ± 0.02	0.22 ± 0.02
ogbn-arxiv			
TWP	4.84 ± 0.02	2.03 ± 0.01	2.04 ± 0.01
GEM	4.85 ± 0.07	2.03 ± 0.03	2.03 ± 0.03
EWC	4.88 ± 0.04	2.04 ± 0.02	2.05 ± 0.01
MAS	4.85 ± 0.05	2.03 ± 0.02	2.04 ± 0.03
LwF	4.89 ± 0.02	2.05 ± 0.01	2.06 ± 0.01
ER-GNN	32.38 ± 2.90	34.37 ± 3.04	31.94 ± 2.90
fine-tuning	4.87 ± 0.04	2.04 ± 0.02	2.05 ± 0.02

K CONFIGURATION OF THE REFERENCES AND THE CASCADE

Table 14 gives the configuration of the classifiers trained here. It is fixed across datasets, and every row of the references and the cascade is trained at it, except those of Table 15. The four terms of R_{joint} share it with one another by construction: a difference between consecutive terms is meant to carry one capability, and would not if it also carried a second choice of width or epoch budget.

Table 14: The configuration of the references and of the cascade's per-task networks, shared across all datasets. The graph convolutional network is that of Kipf & Welling (2017); the diffusion is the propagation of Gasteiger et al. (2019) and the linear classifier over it that of Wu et al. (2019). Every optimizer is that of Kingma & Ba (2015). The cascade holds one such network per task.

	R_{prop}	$R_1, \ldots, R_4$	cascade, per task
classifier	linear	2-layer GCN	2-layer GCN
hidden width	—	256	256
dropout	—	0.5	0.5
epochs	300	200	200
optimizer	Adam	Adam	Adam
learning rate	10^{-2}	10^{-2}	10^{-2}
weight decay	10^{-4}	5×10^{-4}	5×10^{-4}
label smoothing	—	—	0.1
early stopping	none	none	none
feature diffusion	personalized PageRank, $\alpha = 0.15$, 10 iterations		

The correct-and-smooth step is the same operation wherever it appears, in R_4, in R_{prop} and in the cascade's label-memory tier: fifty residual-propagation iterations retaining 0.9 of the propagated residual, the labeled rows then reset to their one-hot labels, and fifty smoothing iterations retaining 0.8, all over the symmetrically normalized adjacency of the scoring graph G' with self-loops added. That graph is G under HIDE and the union of the task subgraphs under Isolated and Mixed, for the references and for the cascade alike.

The cascade selects its two per-dataset quantities, k and ε, on calibration (Appendix M). The candidates are $k \in \{16, 32, 48, 64, 96, 128\}$, kept only where k lies below the rank of the smallest task's calibration half and below the feature dimension, together with the untruncated case, and $\varepsilon \in \{0.05, 0.1, 0.2, 0.3, 0.5\}$. Each tier selects its k by the routing accuracy on the held-out calibration half, and ε is selected at the k of the label-memory tier; a tie goes to the narrower k and to the smaller ε.

The choice of the fixed configuration was measured. The four terms of R_{joint} were retrained at a configuration selected on calibration from a grid of eight settings, the hidden widths $\{64, 256\}$, the learning rates $\{10^{-2}, 10^{-3}\}$ and the weight decays $\{5 \times 10^{-4}, 5 \times 10^{-5}\}$, each read every twenty epochs from 20 to 400, which makes 160 candidates per dataset. One candidate is selected per dataset, by the mean calibration accuracy of R_1 over seeds 0 to 4; R_1 is the term that holds the calibration nodes out of its training set, and it is scored on them per node on G, as the reported metric scores it. The winner is then applied to all four terms on seeds 0 to 9, which is why Table 15 lists one configuration per dataset.

Table 15 gives what was chosen and what it moved. For $R_{\text{prop}} - R_1$ no sign changes. For δ the sign changes on Reddit, from $+0.46$ to -0.17, and on CoraFull and WikiCS the seeds stop agreeing on it. On Reddit the search returns the fixed configuration's accuracy to within 0.01 points.

Three datasets, Amazon Computers, Reddit and ogbn-products, select the largest epoch count on the grid, 400, and Amazon Photo and WikiCS select 380. A selection at the boundary of the grid records that the mean calibration accuracy at the last reading was at least that at every earlier one; what lies past 400 epochs was not measured, and the comparison Table 15 supports is with this grid.

Table 15: The reference ladder at a configuration chosen on calibration, over ten seeds. Width, rate, decay and epoch count are selected on the calibration accuracy of R_1 and applied to all four terms. The R_1 gain is the searched first term minus the fixed one. The two pairs of columns beside it are $\mathrm{Acc}(R_{\mathrm{prop}}) - \mathrm{Acc}(R_1)$ and $\delta = \mathrm{Acc}(R_{\mathrm{prop}}) - \mathrm{Acc}(R_4)$, each at the fixed and at the searched configuration. R_{prop} is not searched, so both pairs move on the R_{joint} side alone. A dagger marks a difference whose seeds do not all agree on its sign.

	chosen on calibration					$R_{\mathrm{prop}} - R_1$		$\delta = R_{\mathrm{prop}} - R_4$	
dataset	width	rate	decay	epochs	R_1 gain	fixed	searched	fixed	searched
Cora	64	10^{-3}	5×10^{-4}	240	$+0.43^{\dagger}$	-2.34	-2.77	$-1.00^{\dagger}$	-1.58
CiteSeer	64	10^{-3}	5×10^{-4}	40	$+0.42^{\dagger}$	-3.15	-3.56	$-0.83^{\dagger}$	-2.46
Amazon Photo	64	10^{-2}	5×10^{-4}	380	$+0.33^{\dagger}$	$+1.62$	$+1.29$	$+1.19$	$+1.08$
Amazon Computers	64	10^{-2}	5×10^{-4}	400	$+0.27^{\dagger}$	$+3.18$	$+2.91$	$+0.64^{\dagger}$	$+0.96^{\dagger}$
WikiCS	256	10^{-2}	5×10^{-5}	380	$+0.85$	$+3.39$	$+2.54$	$+1.51$	$+0.16^{\dagger}$
CoraFull	256	10^{-2}	5×10^{-5}	140	$+0.29^{\dagger}$	-2.56	-2.85	-0.63	$-0.25^{\dagger}$
ogbn-arxiv	256	10^{-2}	5×10^{-5}	340	$+2.08$	$+6.99$	$+4.91$	$+0.83$	$+0.18$
Reddit	256	10^{-2}	5×10^{-5}	400	$-0.01^{\dagger}$	$+2.67$	$+2.68$	$+0.46$	-0.17
ogbn-products	256	10^{-2}	5×10^{-5}	400	$+1.61$	$+3.03$	$+1.43$	-0.98	-1.38
Roman-empire	64	10^{-3}	5×10^{-5}	340	$+1.97$	$+16.28$	$+14.31$	$+29.39$	$+34.04$

L THE REFERENCE LADDER

Table 16 gives R_{prop} and the four terms of R_{joint} on every dataset they were measured on. The two references are measured on ten datasets; the body of the paper scores its methods on eight of them, and the other two, Reddit and ogbn-products, are kept here so that the interval the references span can be checked outside those eight. Consecutive terms differ in one operation, the correct-and-smooth step counted as one.

Table 16: Per-node accuracy on G with inter-task edges present, mean and standard deviation over ten seeds, one node partition per seed. $R_1, \ldots, R_4$ restore one operation of the jointly trained model per column, left to right; R_1 is a reconstruction of the joint baseline the CGLB code trains, matched to R_2 to R_4 in architecture, optimizer and epoch budget and scored under HIDE rather than under Isolated, and R_4 restores everything R_{prop} uses. Appendix K gives the configuration of all five, and Table 6 the datasets.

		R_{joint}, one capability restored per column			
dataset	R_{prop}	R_1 subgraphs	R_2 on G	R_3 + calib.	R_4 + C&S
Cora	87.56 ± 1.65	89.90 ± 0.93	87.17 ± 0.90	87.66 ± 1.17	88.55 ± 1.11
CiteSeer	73.10 ± 0.95	76.25 ± 1.10	72.01 ± 1.20	73.19 ± 1.10	73.93 ± 1.19
Amazon Photo	94.44 ± 0.31	92.82 ± 0.55	93.77 ± 0.77	93.72 ± 0.67	93.25 ± 0.65
Amazon Computers	90.18 ± 0.25	87.00 ± 0.98	89.92 ± 0.42	90.10 ± 0.60	89.54 ± 0.34
WikiCS	84.51 ± 0.60	81.13 ± 0.48	82.18 ± 0.69	82.37 ± 0.73	83.00 ± 0.69
CoraFull	70.64 ± 0.60	73.20 ± 0.58	70.98 ± 0.64	71.95 ± 0.75	71.27 ± 0.56
ogbn-arxiv	74.46 ± 0.23	67.47 ± 0.22	68.00 ± 0.23	68.01 ± 0.15	73.63 ± 0.18
Reddit	94.85 ± 0.10	92.17 ± 0.22	94.26 ± 0.09	94.27 ± 0.09	94.39 ± 0.07
ogbn-products	87.72 ± 0.04	84.69 ± 0.05	84.95 ± 0.05	84.95 ± 0.05	88.70 ± 0.03
Roman-empire	43.36 ± 0.58	27.07 ± 1.02	48.40 ± 0.82	48.75 ± 0.82	13.97 ± 0.48

$\mathrm{Acc}(R_2) - \mathrm{Acc}(R_1)$ carries the training-time removal of the edges that cross tasks. It does not isolate it: R_1 trains on the union of the per-task subgraphs and is scored on G, so the difference compounds the information withheld during training with the mismatch between the graph a model was fitted on and the graph it is scored on. Separating the two would need the remaining cells of the two-by-two over training and scoring graphs, which are not measured here.

That difference separates the panel into two groups of datasets. On the nine datasets whose edge homophily (Table 6) lies between 0.567 and 0.827 the difference lies between -2.73 and $+2.92$

Table 17: Every difference between the references, as paired means in percentage points. Each runs over the ten seeds its two rows share, and a dagger marks a cell whose seeds do not all agree with its sign.

difference	CoraFull	ogbn-arxiv	Cora	CiteSeer	Photo	Comp.	WikiCS	Reddit	Roman	prod.
$R_{\text{prop}} - R_1$	-2.56	$+6.99$	-2.34	-3.15	$+1.62$	$+3.18$	$+3.39$	$+2.67$	$+16.28$	$+3.03$
$R_2 - R_1$, training graph	-2.22	$+0.53$	-2.73	-4.23	$+0.94^{\dagger}$	$+2.92$	$+1.05$	$+2.08$	$+21.32$	$+0.27$
$R_3 - R_2$, calibration labels	$+0.97$	$+0.01^{\dagger}$	$+0.49^{\dagger}$	$+1.18^{\dagger}$	$-0.04^{\dagger}$	$+0.18^{\dagger}$	$+0.19^{\dagger}$	$+0.01^{\dagger}$	$+0.36^{\dagger}$	$-0.00^{\dagger}$
$R_4 - R_3$, correct and smooth	-0.68	$+5.62$	$+0.90^{\dagger}$	$+0.74^{\dagger}$	-0.47	$-0.56^{\dagger}$	$+0.63^{\dagger}$	$+0.12$	-34.78	$+3.75$
$\delta = R_{\text{prop}} - R_4$	-0.63	$+0.83$	$-1.00^{\dagger}$	$-0.83^{\dagger}$	$+1.19$	$+0.64^{\dagger}$	$+1.51$	$+0.46$	$+29.39$	-0.98
R_{prop} without its labels	-1.04	-0.57	$-0.41^{\dagger}$	-1.58	$-0.21^{\dagger}$	-0.31	-0.61	-0.24	$+0.87$	-0.75
R_{prop} without smoothing	-0.49	-4.16	-1.27	$-0.75^{\dagger}$	$+1.18$	$+0.41^{\dagger}$	-0.88	-2.07	$+18.81$	-10.03

and takes either sign, with one exception at -4.23 on CiteSeer; on Roman-empire, whose edge homophily is 0.047, it is $+21.32$, with every seed above and a bootstrap interval of $[+20.47, +21.98]$.

Which property of the graph separates the two groups cannot be settled on this panel. Because tasks are class-disjoint, an edge that crosses a task boundary is an edge whose endpoints carry different labels, and the two quantities are near-substitutes here: the fraction of deleted edges and the edge homophily rank at Spearman -0.93 across the ten datasets. Roman-empire is also the only dataset in the heterophilous range, so the second group is a single graph, and on that graph the training removal takes 89% of the edges, which is where the confound above is largest.

The remaining two capabilities are cheaper to read. $\mathrm{Acc}(R_3) - \mathrm{Acc}(R_2)$, the calibration labels, lies between -0.04 and $+1.18$. $\mathrm{Acc}(R_4) - \mathrm{Acc}(R_3)$ isolates Roman-empire again and adds ogbn-arxiv and ogbn-products: on ogbn-arxiv it is $+5.62$, which is 80% of the 6.99 points between R_1 and R_{prop} and still leaves R_4 0.83 below R_{prop}, and on ogbn-products it is $+3.75$, which carries R_4 past R_{prop} by 0.98. On the remaining homophilous datasets it lies between -0.68 and $+0.90$. On Roman-empire it is -34.78: smoothing a posterior over neighbors that mostly carry other labels destroys the posterior.

The two things R_{prop} uses, the retained labels and the smoothing over G, separate by the same kind of ablation. Withholding the calibration labels never lowers $\mathrm{Acc}(R_{\text{prop}})$ by more than 1.58 across the ten datasets it is measured on, and on one of them, Roman-empire, it helps by $+0.87$. Withholding the smoothing changes it by -10.03 on ogbn-products, -4.16 on ogbn-arxiv and -2.07 on Reddit, by less than 1.3 on the four remaining homophilous datasets, and by $+1.18$, $+0.41$ and $+18.81$ on Amazon Photo, Amazon Computers and Roman-empire. The smoothing is the term whose sign depends on the graph.

M THE CASCADE OF FROZEN PER-TASK NETWORKS

The cascade is a baseline in which no per-task network is updated after its task. It trains one network per task, at the configuration of the last column of Table 14, freezes it when the task ends, and at inference assigns each node to one task and classifies it with that task's network. No parameter of a frozen network is overwritten, so its errors are the errors of the assignment and those of the frozen networks within their own tasks. Its accuracy can still change over the sequence, because the graph it scores on grows, the diffusion computed on that graph changes with it, and the label-memory router is refit on it, so a node's assignment can move among the tasks already seen as well as toward a new one (Appendix S). The assignment is built in three tiers, which differ in what they retain and what they observe at inference.

The streaming tier assigns a node to the task whose calibration subspace leaves the least of the node's diffused features unexplained. Let $\hat{A} = D^{-1/2}(A + I)D^{-1/2}$ be the normalized adjacency of the scoring graph G' and X the feature matrix; the diffused features are $H = H^{(10)}$ with $H^{(0)} = X$

and $H^{(i)} = (1-\alpha)\hat{A}H^{(i-1)} + \alpha X$ at $\alpha = 0.15$ (Gasteiger et al., 2019), and h_v is the row of H at node v. For each task t the same diffusion is run on the features of V_t over the task subgraph $G[V_t]$, the rows at the calibration nodes of task t are centred by their mean, and $B_t \in \mathbb{R}^{d\times k_t}$ holds the first k_t right singular vectors of that centred block, with $k_t = \min(k, r_t)$ and $r_t = \min(|V_{\text{cal}} \cap V_t|, d-1)$ the number of directions the block has. The score of task t for node v is the fraction of energy outside the subspace,

$$s_t(v) = 1 - \frac{\|B_t^\top h_v\|^2}{\|h_v\|^2}, \tag{2}$$

with h_v taken as diffused on G' and left uncentred, and the tier assigns $\hat{t}(v) = \arg\min_t s_t(v)$, the lowest index on a tie. It retains the bases $B_1, \ldots, B_T$ and their task grouping and no label, and it scores each node independently of the rest of the batch S.

The label-memory tier adds the class posterior of R_{prop}, read as a task posterior, to the subspace score. The posterior p_v over the classes seen so far is the one R_{prop} produces on H and G': one linear layer on H standardized by the mean and standard deviation of its labelled rows, fit by 300 steps of Adam at learning rate 0.01 and weight decay 10^{-4} on the training and calibration labels of those classes, then corrected and smoothed over $\hat{A}$ (Huang et al., 2021) with the same labelled rows, the residual propagated for 50 steps at 0.9, the labelled rows reset to their labels, and the result propagated for 50 steps at 0.8. The task posterior is $q_t(v) = \sum_{c\in C_t} p_v(c)$, its cost $\ell_t(v) = -\log\max(q_t(v), 10^{-12})$, and the combined cost is

$$c_t(v) = \mathrm{z}(s)_{v,t} + \mathrm{z}(\ell)_{v,t}, \tag{3}$$

where z subtracts from the $|S| \times T$ matrix its mean and divides by its standard deviation, both taken over every entry, so that the two families of scores enter on one scale. Both terms are lower for the task that fits better, and the tier assigns $\hat{t}(v) = \arg\min_t c_t(v)$, the lowest index on a tie. It retains the labels R_{prop} retains and carries R_{prop} whole, and the batch enters it only through the ratio of the two standard deviations in Equation 3.

The transductive tier replaces the per-node argmin by a balanced assignment over the whole of S (Cuturi, 2013). The kernel is $K_{v,t} = \exp(-\mathrm{z}(c)_{v,t}/\varepsilon)$ at temperature ε; the row marginal is $1/|S|$ for every node and the column marginal is $m_t = |V_{\text{cal}} \cap V_t|/|V_{\text{cal}}|$, the task proportions counted on the calibration nodes; with $r = \mathbf{1}/|S|$, 200 Sinkhorn iterations from $b = \mathbf{1}$, $a \leftarrow r/(Kb)$ and $b \leftarrow m/(K^\top a)$, give the plan $P = \mathrm{diag}(a)\, K\, \mathrm{diag}(b)$, and the tier assigns $\hat{t}(v) = \arg\max_t P_{v,t}$. It reads the composition of S, which no reference does. A method may read the batch, and under Mixed and HIDE the batch holds the test nodes of every task, so the transductive tier is permitted under both.

Each tier's parameters are chosen on calibration, with the test set held out of the choice. The streaming and label-memory tiers each choose a truncation k; the transductive tier inherits the label-memory tier's k and chooses ε alone. The calibration nodes of each task are halved, deterministically per seed, into a fitting half A and a selection half B. During selection every component that uses labels sees the training labels and the labels of half A only: the subspace bases and the counts m_t are computed from half A, and the linear head is fit and its correction and smoothing are run with the training nodes and half A as the labelled set. The labels of half B are used for one thing, the task hit of each candidate on half B, the fraction of its nodes assigned to their own task, and each tier takes the candidate with the highest task hit, the narrower k or the smaller ε on a tie. The candidates for k are the members of $\{16, 32, 48, 64, 96, 128\}$ below the smallest task's number of directions on half A, together with the untruncated case, which keeps every direction of each task's block; a member at or beyond that number would coincide with the untruncated case during selection and denote a different truncation after the refit, so it is excluded. The candidates for ε are $\{0.05, 0.1, 0.2, 0.3, 0.5\}$. After the choice, the bases, the counts, the head, and its correction and smoothing are refit on the training nodes and the whole calibration set, halves A and B together, at the chosen configuration, and the test nodes are routed once. Unlabelled access to the graph, the features, the edges and the diffusion over them, is not restricted at any stage. Under Isolated and Mixed the cascade reads the graph with the inter-task edges removed and makes the choice under that graph. The methods, by contrast, run at the configuration their authors released, so a comparison between the cascade and a method also carries this difference.

The re-selection of k under Isolated and Mixed moves the cascade's Mixed accuracy by at most 0.25 points on every dataset but Roman-empire. Table 18 gives, per node over ten seeds, the Mixed

accuracy with k re-selected on the graph without inter-task edges, the number every table of this paper reports, beside a control that holds k at the value selected under HIDE, and their paired difference. On CoraFull and ogbn-arxiv both selections pick the untruncated subspace on every seed, so the two numbers coincide and no control was run. On Roman-empire the re-selection picks the untruncated subspace on the task subgraphs on every seed, where under HIDE it picks 32 or 48, and the re-selected number is 3.27 points higher on all ten seeds; with k held, the cascade's cost from Mixed to HIDE there is 32.18 points in place of 35.45, and on the other datasets it is unchanged to the first decimal. The contrast that isolates the graph alone is therefore the k-held one, and the tables keep the re-selected number because each method is configured on its own setting.

Table 18: The cascade under Mixed, per node, with k re-selected on the graph without inter-task edges, the number the tables report, and with k held at the value selected under HIDE; mean and standard deviation over seeds 0 to 9. Δ is the paired difference, re-selected minus held, with the count of seeds on which it is positive, negative and zero. CoraFull and ogbn-arxiv select the untruncated subspace under both graphs on every seed and are omitted.

dataset	k re-selected	k held	Δ	$+/-/0$
Cora	94.45 ± 1.00	94.67 ± 0.79	-0.22 ± 0.64	3/4/3
CiteSeer	80.09 ± 1.35	80.11 ± 1.17	-0.02 ± 0.40	4/5/1
Amazon Photo	97.48 ± 0.21	97.51 ± 0.28	-0.03 ± 0.13	3/4/3
Amazon Computers	97.32 ± 0.30	97.30 ± 0.34	$+0.03 \pm 0.17$	4/2/4
WikiCS	91.65 ± 0.32	91.71 ± 0.40	-0.06 ± 0.21	2/5/3
Roman-empire	62.77 ± 0.61	59.50 ± 0.80	$+3.27 \pm 0.52$	10/0/0

The cascade carries far more parameters than any reference: on CoraFull the label-memory tier carries 125 times those of R_{prop} and 35 times those of R_1 (Appendix I).

The label-memory tier is the one Tables 21, 31 and 32 report. It reads the batch only through the two standard deviations of Equation 3, and it is scored once on the task subgraphs and read under both batches, so the batch cost in Table 32 is zero by construction. The inter-task edges cost it 13.30 points per node on CoraFull and 15.09 on ogbn-arxiv, while from Mixed to HIDE its task hit falls from 86.78% to 73.06% and from 90.66% to 75.14% (Table 19). Given the true task, its experts classify 96.84% and 96.93% of the test nodes on CoraFull and 97.12% and 97.02% on ogbn-arxiv under the two graphs, so the loss of that step is in the assignment of nodes to tasks, measured rather than inferred from the frozen parameters. Under HIDE it is 1.20 points above R_{prop} on CoraFull and 0.91 below it on ogbn-arxiv, on every seed. Against R_4 its mean difference is $+0.57$ on CoraFull, with a bootstrap interval of $[+0.19, +0.97]$ that excludes zero and seeds that do not all agree on the sign, and -0.08 on ogbn-arxiv, with an interval of $[-0.23, +0.05]$ that contains zero; it is 1.36 below R_1 on CoraFull and 6.08 above it on ogbn-arxiv.

The other two tiers fall on either side of the reference each is compared with (Table 33). The streaming tier, which retains no label, falls 2.36 points short of R_{prop} on CoraFull and 11.11 on ogbn-arxiv, on every seed. The transductive tier exceeds R_4 by 2.17 on CoraFull, on all ten seeds, and by 0.30 on ogbn-arxiv, with an interval of $[+0.17, +0.43]$ and nine of ten seeds agreeing on the sign.

Table 19: The cascade's label-memory tier under Mixed (the task subgraphs) and under HIDE (the whole graph), per node, with its task hit, the share of test nodes assigned to their own task, and the class accuracy given the true task, every node classified by the expert of its own task; shared graph, mean and standard deviation over seeds 0 to 9. The accuracies are those of Table 21; the given-true-task column was measured on CoraFull and ogbn-arxiv only.

	Mixed			HIDE		
dataset	per node	task hit	class | true task	per node	task hit	class | true task
CoraFull	85.15 ± 0.60	86.78 ± 0.58	96.84 ± 0.24	71.85 ± 0.85	73.06 ± 0.85	96.93 ± 0.18
ogbn-arxiv	88.65 ± 0.39	90.66 ± 0.40	97.12 ± 0.05	73.55 ± 0.28	75.14 ± 0.29	97.02 ± 0.07
Cora	94.45 ± 1.00	97.03 ± 0.80	—	87.62 ± 1.59	89.84 ± 1.34	—
CiteSeer	80.09 ± 1.35	91.58 ± 0.79	—	74.04 ± 1.00	84.23 ± 1.08	—
Amazon Photo	97.48 ± 0.21	98.97 ± 0.16	—	94.34 ± 0.45	96.09 ± 0.36	—
Amazon Computers	97.32 ± 0.30	97.81 ± 0.28	—	90.49 ± 0.28	90.91 ± 0.31	—
WikiCS	91.65 ± 0.32	96.83 ± 0.26	—	82.93 ± 0.71	87.98 ± 0.75	—
Roman-empire	62.77 ± 0.61	73.43 ± 0.35	—	27.32 ± 1.01	43.74 ± 0.68	—

N Accuracy and cost on the eight datasets and Reddit

Table 21 gives the accuracy of every method on the eight datasets, and on Reddit, in five readings of the three settings of Table 2, with the cascade and the three references beside them, so that the two datasets the body argues over can be read against the other six in one place; Table 20 gives the cost of each protocol choice on the eight.

The reference rows of Table 21 read two runs of one recipe. The HIDE per-node cell of R_1 and R_4 is the training of Table 16, and every other cell of a reference comes from the run that re-scores the references under the three settings, the only run scored on the union of the task subgraphs. The two runs share the labels, the edges, the epoch budget and the seed, but a GCN retrained on the GPU does not repeat bit for bit, and R_1, trained on the union of the task subgraphs and scored on G, is the reference this variation moves: in the re-scoring run one of its ten seeds on Amazon Computers landed at 74.75 against 87.75 in the training shown, and one on Amazon Photo at 86.92 against 92.45, which is where the earlier disagreement of 1.44 and 1.13 points between the two tables came from. In the ten-seed mean the two runs differ by at most 0.62 points for R_4, and for R_1 by at most 0.3 outside the two Amazon datasets and by less than 0.7 on them once the diverged seed is set aside, the run-to-run variation Appendix A records.

Table 20: Cost of each protocol choice, in accuracy points; negative is a loss. Each block compares two settings under one aggregation; the second and third blocks sum to the per-node loss from Isolated to HIDE, and the last block gives that loss under AA. Shared graph, paired means over seeds 0 to 9; a dagger marks a cell whose seeds do not all agree with its sign. R_1 and R_4 keep the model they trained, on the task subgraphs and on G respectively, and are rescored under each setting; R_{prop} and the correct-and-smooth step of R_4 run on the scoring graph (Section 4). Table 3 in the body keeps the first, second and last blocks.

method	CoraFull	ogbn-arxiv	Cora	CiteSeer	Photo	Comp.	WikiCS	Roman
aggregation: AA → *per node, under Isolated*								
PDGNNs	$+4.88$	$+3.88$	$+0.05^{\dagger}$	$+2.89$	$-0.14^{\dagger}$	-2.11	-1.69	$-0.66^{\dagger}$
DeLoMe	$+3.81$	-13.44	$-0.50^{\dagger}$	$+2.15$	$+0.15^{\dagger}$	-1.88	$-0.14^{\dagger}$	-3.21
SSM	$+4.71$	$+13.55$	$-0.64^{\dagger}$	$+2.83$	$-1.38^{\dagger}$	$-5.57^{\dagger}$	$+0.88^{\dagger}$	-2.59
TPP	$+2.83$	$+7.18$	$+0.41^{\dagger}$	$+3.35$	$+1.68$	$+0.53^{\dagger}$	$+0.10^{\dagger}$	$-0.19^{\dagger}$
TAAM	$+2.78$	$+1.06$	$+0.37^{\dagger}$	$+2.19$	$+0.81$	$+0.55$	-0.60	-1.18
DyMoE	$+10.50$	$+21.45$	$+0.36^{\dagger}$	$+3.96$	$+1.27^{\dagger}$	$+13.53$	$+3.15$	$+6.11$
cascade	$+7.51$	$+17.62$	$+0.60$	$+2.78$	$+1.55$	$+0.15^{\dagger}$	$+1.04$	$+8.99$
R_1	$+6.12$	$+29.13$	$+0.42^{\dagger}$	$+2.88$	$+1.39^{\dagger}$	$+0.09^{\dagger}$	$+1.12$	$+9.06$
R_{prop}	$+6.00$	$+15.63$	$+0.32^{\dagger}$	$+2.70$	$+2.06$	$+0.37^{\dagger}$	$+0.98$	$+4.92$
R_4	$+5.68$	$+17.18$	$+0.39^{\dagger}$	$+2.83$	$+2.38$	$+0.42^{\dagger}$	$+1.16$	-1.47
batch: Isolated → Mixed								
PDGNNs	0	0	0	0	0	0	0	0
DeLoMe	0	0	0	0	0	0	0	0
SSM	0	0	0	0	0	0	0	0
TPP	-16.89	-49.00	-9.00	-7.48	-24.42	-26.53	-29.47	-16.26
TAAM	-36.46	-49.45	-18.22	-13.23	-11.28	-20.89	-23.60	-41.39
DyMoE	0	0	0	0	0	0	0	0
cascade	0	0	0	0	0	0	0	0
R_1	0	0	0	0	0	0	0	0
R_{prop}	0	0	0	0	0	0	0	0
R_4	0	0	0	0	0	0	0	0
inter-task edges: Mixed → HIDE								
PDGNNs	-11.84	-6.40	-4.34	-3.70	-4.78	-7.30	-8.12	-27.70
DeLoMe	-12.78	-4.78	-4.86	-3.80	-6.66	-9.64	-9.75	-24.91
SSM	-18.06	-13.83	-6.38	-4.16	-17.14	0	-23.99	-23.77
TPP	-24.76	-15.88	-5.86	-5.36	-10.96	-16.93	-22.09	-58.73
TAAM	-3.99	-1.87	-1.75	$-0.70^{\dagger}$	-2.86	-3.26	-4.02	-12.42
DyMoE	-0.41	-2.41	$-0.98^{\dagger}$	$-0.09^{\dagger}$	$-0.29^{\dagger}$	-0.86	-6.47	$-4.68^{\dagger}$
cascade	-13.30	-15.09	-6.84	-6.05	-3.14	-6.83	-8.72	-35.45
R_1	-13.66	-9.30	-4.77	-3.96	-5.52	-10.26	-10.14	-36.37
R_{prop}	-16.10	-15.49	-7.01	-7.57	-2.73	-7.12	-7.92	-16.42
R_4	-15.55	-15.99	-5.66	-6.33	-3.95	-8.64	-9.51	-35.89
batch and edges: Isolated → HIDE, AA *throughout*								
PDGNNs	-13.14	-8.64	-4.35	-4.14	-4.15	-8.18	-9.69	-36.02
DeLoMe	-12.72	-4.76	-4.82	-4.32	-5.35	-11.96	-11.87	-31.27
SSM	-20.27	-14.01	-6.62	-4.82	-15.97	0	-30.47	-28.24
TPP	-50.27	-75.28	-14.75	-13.84	-43.67	-64.12	-63.59	-74.87
TAAM	-46.95	-64.11	-18.39	-14.71	-16.96	-38.98	-33.19	-59.30
DyMoE	-0.44	-1.74	$-0.87^{\dagger}$	$-0.31^{\dagger}$	$-0.34^{\dagger}$	-1.25	-7.46	-7.29
cascade	-16.31	-19.71	-7.06	-6.92	-2.39	-8.74	-11.99	-34.84
R_1	-15.96	-9.20	-4.99	-4.47	-5.28	-13.75	-14.56	-38.20
R_{prop}	-16.44	-20.88	-6.92	-7.78	-2.92	-9.03	-10.22	-17.02
R_4	-17.57	-21.22	-5.74	-6.79	-4.08	-11.56	-13.15	-39.86

Table 21: Every method on the eight datasets under the shared graph, in five readings of the three settings of Table 2, mean and standard deviation over seeds 0 to 9, and on Reddit over seeds 0 to 4, which the body excludes from every range and ranking. On Reddit, the cascade is not shown, having fewer than five seeds of the five readings there (Table 33 gives its HIDE accuracy against the references on Reddit, over ten seeds), and the references are not re-scored; Table 16 gives their HIDE accuracy. TAAM and DyMoE are the reconstructions of Appendix H. The HIDE per-node cell of R_1 and R_4 is the training of Table 16, so the two tables read one training; their other four cells, and every cell of R_{prop}, come from the re-scoring run, the only run scored on the union of the task subgraphs. R_1 retrained on the GPU is not deterministic, and in that run one of its seeds per Amazon dataset diverged, to 74.75 on Amazon Computers and 86.92 on Amazon Photo against 87.75 and 92.45 in the training shown, which is why the two runs disagreed by more than a point there; in the ten-seed mean R_4 differs between the runs by at most 0.62 points.

	Isolated		Mixed	HIDE	
method	AA	per node	per node	per node	AA
CoraFull					
PDGNNs	81.39 ± 0.55	86.27 ± 0.43	86.27 ± 0.43	74.44 ± 0.64	68.24 ± 0.74
DeLoMe	80.34 ± 0.58	84.15 ± 0.59	84.15 ± 0.59	71.37 ± 1.08	67.62 ± 1.06
SSM	76.50 ± 0.91	81.22 ± 0.85	81.22 ± 0.85	63.16 ± 0.87	56.24 ± 1.14
TPP	93.94 ± 0.70	96.77 ± 0.32	79.88 ± 0.69	55.12 ± 0.62	43.67 ± 0.95
TAAM*	94.22 ± 1.09	97.00 ± 0.29	60.54 ± 0.90	56.55 ± 0.88	47.27 ± 0.93
DyMoE*	50.82 ± 1.31	61.33 ± 1.15	61.33 ± 1.15	60.92 ± 1.13	50.38 ± 1.22
cascade	77.64 ± 0.65	85.15 ± 0.60	85.15 ± 0.60	71.85 ± 0.85	61.33 ± 0.95
R_1	80.77 ± 0.61	86.89 ± 0.41	86.89 ± 0.41	73.20 ± 0.58	64.80 ± 0.98
R_{prop}	80.75 ± 0.72	86.74 ± 0.49	86.74 ± 0.49	70.64 ± 0.60	64.31 ± 0.74
R_4	81.16 ± 0.56	86.84 ± 0.47	86.84 ± 0.47	71.27 ± 0.56	63.59 ± 0.76
ogbn-arxiv					
PDGNNs	65.61 ± 0.56	69.48 ± 0.43	69.48 ± 0.43	63.08 ± 0.24	56.97 ± 0.63
DeLoMe	52.82 ± 0.94	39.38 ± 2.84	39.38 ± 2.84	34.59 ± 2.22	48.06 ± 0.52
SSM	56.35 ± 1.11	69.91 ± 0.92	69.91 ± 0.92	56.08 ± 0.88	42.34 ± 0.61
TPP	89.84 ± 0.45	97.02 ± 0.06	48.01 ± 0.13	32.13 ± 0.15	14.55 ± 0.14
TAAM*	93.54 ± 0.36	94.60 ± 0.11	45.15 ± 0.32	43.28 ± 0.32	29.44 ± 0.30
DyMoE*	23.88 ± 1.21	45.33 ± 1.97	45.33 ± 1.97	42.92 ± 2.01	22.13 ± 1.07
cascade	71.03 ± 0.94	88.65 ± 0.39	88.65 ± 0.39	73.55 ± 0.28	51.32 ± 0.54
R_1	47.67 ± 0.41	76.80 ± 0.21	76.80 ± 0.21	67.47 ± 0.22	38.47 ± 0.44
R_{prop}	74.33 ± 0.24	89.95 ± 0.12	89.95 ± 0.12	74.46 ± 0.23	53.44 ± 0.53
R_4	72.45 ± 0.34	89.63 ± 0.11	89.63 ± 0.11	73.63 ± 0.18	51.23 ± 0.59
Cora					
PDGNNs	94.36 ± 1.25	94.41 ± 0.87	94.41 ± 0.87	90.08 ± 1.10	90.02 ± 1.43
DeLoMe	93.39 ± 1.50	92.89 ± 1.08	92.89 ± 1.08	88.03 ± 1.58	88.58 ± 1.61
SSM	91.33 ± 0.97	90.68 ± 0.82	90.68 ± 0.82	84.30 ± 1.46	84.71 ± 2.39
TPP	96.64 ± 0.83	97.05 ± 0.58	88.05 ± 1.18	82.19 ± 1.45	81.89 ± 1.44
TAAM*	96.95 ± 0.74	97.32 ± 0.73	79.10 ± 1.53	77.34 ± 1.56	78.56 ± 1.58
DyMoE*	78.88 ± 3.41	79.24 ± 2.96	79.24 ± 2.96	78.26 ± 3.05	78.00 ± 3.54
cascade	93.85 ± 1.29	94.45 ± 1.00	94.45 ± 1.00	87.62 ± 1.59	86.79 ± 2.64
R_1	94.24 ± 0.80	94.67 ± 0.75	94.67 ± 0.75	89.90 ± 0.93	89.25 ± 0.92
R_{prop}	94.25 ± 1.12	94.57 ± 0.88	94.57 ± 0.88	87.56 ± 1.65	87.33 ± 1.94
R_4	93.83 ± 1.17	94.22 ± 0.96	94.22 ± 0.96	88.55 ± 1.11	88.09 ± 1.36
CiteSeer					
PDGNNs	77.34 ± 1.42	80.22 ± 1.11	80.22 ± 1.11	76.53 ± 0.96	73.19 ± 1.30
DeLoMe	77.30 ± 1.19	79.45 ± 1.01	79.45 ± 1.01	75.65 ± 0.98	72.98 ± 1.30
SSM	73.57 ± 1.78	76.39 ± 1.61	76.39 ± 1.61	72.23 ± 1.49	68.75 ± 1.46
TPP	80.98 ± 1.27	84.34 ± 1.31	76.85 ± 1.26	71.49 ± 1.05	67.14 ± 1.40
TAAM*	83.12 ± 1.65	85.30 ± 1.38	72.07 ± 1.23	71.37 ± 1.23	68.41 ± 1.48
DyMoE*	58.58 ± 2.07	62.53 ± 1.88	62.53 ± 1.88	62.44 ± 2.19	58.26 ± 2.44
cascade	77.31 ± 1.48	80.09 ± 1.35	80.09 ± 1.35	74.04 ± 1.00	70.39 ± 1.05
R_1	77.18 ± 1.16	80.06 ± 0.87	80.06 ± 0.87	76.25 ± 1.10	72.71 ± 1.43
R_{prop}	77.97 ± 1.19	80.67 ± 0.95	80.67 ± 0.95	73.10 ± 0.95	70.19 ± 1.22

	Isolated		Mixed	HIDE	
method	AA	per node	per node	per node	AA
R_4	77.24 ± 1.61	80.08 ± 1.32	80.08 ± 1.32	73.93 ± 1.19	70.45 ± 1.26
Amazon Photo					
PDGNNs	96.73 ± 0.50	96.59 ± 0.62	96.59 ± 0.62	91.82 ± 1.49	92.57 ± 0.88
DeLoMe	96.49 ± 0.68	96.64 ± 0.33	96.64 ± 0.33	89.98 ± 0.91	91.14 ± 1.09
SSM	90.00 ± 4.93	88.62 ± 7.72	88.62 ± 7.72	71.48 ± 10.46	74.02 ± 5.76
TPP	96.52 ± 0.92	98.20 ± 0.27	73.78 ± 0.90	62.82 ± 1.08	52.84 ± 1.17
TAAM*	97.52 ± 0.54	98.33 ± 0.32	87.05 ± 0.92	84.18 ± 1.20	80.56 ± 1.09
DyMoE*	81.11 ± 5.02	82.38 ± 6.25	82.38 ± 6.25	82.09 ± 6.09	80.77 ± 4.82
cascade	95.93 ± 0.41	97.48 ± 0.21	97.48 ± 0.21	94.34 ± 0.45	93.54 ± 0.65
R_1	95.83 ± 0.68	97.22 ± 0.74	97.22 ± 0.74	92.82 ± 0.55	90.55 ± 2.10
R_{prop}	95.11 ± 0.70	97.17 ± 0.36	97.17 ± 0.36	94.44 ± 0.31	92.19 ± 0.64
R_4	94.83 ± 0.58	97.21 ± 0.29	97.21 ± 0.29	93.25 ± 0.65	90.75 ± 0.97
Amazon Computers					
PDGNNs	96.54 ± 0.41	94.44 ± 0.42	94.44 ± 0.42	87.13 ± 0.62	88.36 ± 0.99
DeLoMe	96.00 ± 0.43	94.12 ± 0.95	94.12 ± 0.95	84.47 ± 1.10	84.04 ± 1.97
SSM	10.00 ± 0.01	4.43 ± 3.90	4.43 ± 3.90	4.43 ± 3.90	10.00 ± 0.01
TPP	98.76 ± 0.31	99.29 ± 0.11	72.76 ± 0.65	55.83 ± 0.92	34.64 ± 1.21
TAAM*	98.62 ± 0.30	99.17 ± 0.15	78.28 ± 0.86	75.02 ± 1.03	59.63 ± 1.53
DyMoE*	56.06 ± 10.30	69.59 ± 6.52	69.59 ± 6.52	68.72 ± 6.12	54.81 ± 9.92
cascade	97.17 ± 0.32	97.32 ± 0.30	97.32 ± 0.30	90.49 ± 0.28	88.44 ± 0.66
R_1	95.73 ± 2.38	95.82 ± 3.73	95.82 ± 3.73	87.00 ± 0.98	81.98 ± 2.98
R_{prop}	96.93 ± 0.51	97.29 ± 0.32	97.29 ± 0.32	90.18 ± 0.25	87.90 ± 0.62
R_4	97.15 ± 0.34	97.57 ± 0.25	97.57 ± 0.25	89.54 ± 0.34	85.59 ± 1.03
WikiCS					
PDGNNs	90.43 ± 0.43	88.73 ± 0.65	88.73 ± 0.65	80.62 ± 0.99	80.74 ± 1.16
DeLoMe	88.68 ± 0.87	88.53 ± 0.67	88.53 ± 0.67	78.79 ± 0.67	76.81 ± 0.78
SSM	84.22 ± 1.95	85.10 ± 2.58	85.10 ± 2.58	61.11 ± 4.29	53.75 ± 4.44
TPP	94.12 ± 0.31	94.22 ± 0.34	64.74 ± 0.84	42.66 ± 0.74	30.53 ± 0.72
TAAM*	94.50 ± 0.50	93.90 ± 0.36	70.30 ± 0.88	66.28 ± 0.93	61.30 ± 1.26
DyMoE*	76.77 ± 3.19	79.92 ± 1.66	79.92 ± 1.66	73.45 ± 1.63	69.31 ± 2.78
cascade	90.61 ± 0.49	91.65 ± 0.32	91.65 ± 0.32	82.93 ± 0.71	78.62 ± 1.00
R_1	90.18 ± 0.50	91.30 ± 0.36	91.30 ± 0.36	81.13 ± 0.48	75.62 ± 1.03
R_{prop}	91.45 ± 0.49	92.43 ± 0.27	92.43 ± 0.27	84.51 ± 0.60	81.23 ± 0.86
R_4	91.33 ± 0.68	92.49 ± 0.28	92.49 ± 0.28	83.00 ± 0.69	78.18 ± 1.57
Roman-empire					
PDGNNs	60.87 ± 0.56	60.21 ± 0.67	60.21 ± 0.67	32.51 ± 1.25	24.84 ± 0.88
DeLoMe	54.55 ± 1.00	51.34 ± 1.40	51.34 ± 1.40	26.43 ± 1.33	23.28 ± 1.31
SSM	51.50 ± 0.67	48.91 ± 0.50	48.91 ± 0.50	25.13 ± 1.38	23.26 ± 1.75
TPP	87.33 ± 0.36	87.14 ± 0.29	70.89 ± 0.32	12.16 ± 0.44	12.46 ± 0.47
TAAM*	87.52 ± 0.37	86.34 ± 0.45	44.95 ± 0.86	32.54 ± 0.74	28.22 ± 1.04
DyMoE*	34.37 ± 1.12	40.48 ± 2.45	40.48 ± 2.45	35.80 ± 3.77	27.07 ± 1.83
cascade	53.78 ± 1.15	62.77 ± 0.61	62.77 ± 0.61	27.32 ± 1.01	18.94 ± 0.64
R_1	54.40 ± 0.41	63.46 ± 0.56	63.46 ± 0.56	27.07 ± 1.02	16.20 ± 0.59
R_{prop}	54.85 ± 0.67	59.77 ± 0.60	59.77 ± 0.60	43.36 ± 0.58	37.83 ± 0.85
R_4	51.32 ± 0.71	49.85 ± 0.56	49.85 ± 0.56	13.97 ± 0.48	11.46 ± 0.62
Reddit, seeds 0 *to* 4					
PDGNNs	95.86 ± 0.15	96.17 ± 0.14	96.17 ± 0.14	91.59 ± 0.15	88.73 ± 0.20
DeLoMe	95.97 ± 1.09	97.11 ± 0.40	97.11 ± 0.40	90.30 ± 0.82	84.28 ± 1.89
SSM	76.77 ± 4.89	70.51 ± 9.67	70.51 ± 9.67	40.41 ± 3.41	35.80 ± 1.92
TPP	99.40 ± 0.06	99.51 ± 0.03	72.33 ± 0.26	54.34 ± 0.11	37.98 ± 0.22
TAAM*	99.53 ± 0.03	99.54 ± 0.02	52.54 ± 0.18	49.05 ± 0.23	39.85 ± 0.21
DyMoE*	68.57 ± 0.76	75.03 ± 0.78	75.03 ± 0.78	65.38 ± 0.40	56.32 ± 0.70

O WHAT THE DECOMPOSITION ASSUMES

What a message-passing model conditions on when it is evaluated at v is the pair $\big(x_v, \mathcal{N}_{G'}(v)\big)$, its own features and its neighborhood in whichever graph the protocol supplied. Writing the condition-

ing out, class-incremental prediction factors as

$$P\big(y \mid v, G'\big) \;=\; \sum_{t=1}^{T} \underbrace{P\big(y \mid v, t, G'\big)}_{\text{within-task}} \; \underbrace{P\big(t \mid v, G'\big)}_{\text{task identity}}, \tag{4}$$

and with class sets disjoint across tasks a single term survives. The bound of Kim et al. (2022) pairs a task-identity predictor with a within-task classifier and poses the first as out-of-distribution detection over a free-standing sample, whereas the pair above conditions on the graph.

Both factors are conditioned on G'. Scoring a node on the subgraph its own task induces makes $\mathcal{N}_{G'}(v) \subseteq V_t$ hold by construction, so the graph a node is scored on already carries its true t before the model infers it, and the within-task factor is evaluated on a representation the model would never receive on G. A choice of G' changes what both terms measure, and the product of the two is the only quantity a reported accuracy exposes.

The two columns of Table 22 that split accuracy follow from an identity over events, separate from the factorization of Equation equation 4. Class sets are disjoint across tasks, so a predicted class $\hat{y}_v$ names one task, $\hat{t}_v$ with $\hat{y}_v \in C_{\hat{t}_v}$, and a correct class implies a correct task; with t_v the task that owns y_v, over the test nodes S of one run

$$\frac{1}{|S|}\sum_{v\in S} \mathbf{1}[\hat{y}_v = y_v] \;=\; \underbrace{\frac{1}{|S|}\sum_{v\in S} \mathbf{1}[\hat{t}_v = t_v]}_{\text{task hit}} \cdot \underbrace{\frac{\sum_{v\in S} \mathbf{1}[\hat{y}_v = y_v]}{\sum_{v\in S} \mathbf{1}[\hat{t}_v = t_v]}}_{\text{class accuracy} \mid \text{correct task assignment}} . \tag{5}$$

Equation equation 4 factors a posterior at one node, and Equation equation 5 factors an accuracy over a set of nodes, whose second factor is the accuracy conditional on the event that the task assignment was correct, read on the nodes that event selects. The two share the order, task then class, and a column of Table 22 is a factor of Equation equation 5 only.

The column *class | true task* measures a third quantity and is kept as a diagnostic. It restricts the argmax to the classes of the node's own task and reads the result over every test node, so it is the within-task accuracy reached when the task is supplied, whereas the conditional factor is read on the nodes the model assigned correctly, which are the nodes it placed with the least difficulty. On PDGNNs under HIDE on CoraFull the task hit times the conditional factor returns the accuracy, $75.36 \times 98.78\% = 74.44$, and the task hit times the oracle column gives 73.63, below the accuracy the model reaches. The oracle column answers what the task assignment costs over every node, and the conditional column completes the accuracy.

Equation equation 5 holds within one run, and the columns of Table 22 are means over ten seeds, so the product of two column means is an approximation of the accuracy column. On PDGNNs under HIDE on CoraFull the product agrees with the accuracy to the printed precision; on SSM under HIDE on Amazon Photo (Table 23) it gives $81.02 \times 87.88\% = 71.20$ against an accuracy of 71.48. The exact identity is read per seed.

A reference's predicted class names a task as well, and its task-hit column reads exactly that. Its conditional column is left blank as a choice of presentation: within one run the blank cell equals the accuracy divided by the task hit, both of which the table prints.

P TASK HIT AND THE CLASS GIVEN THE TASK

Table 22 gives the accuracy, the task hit, the class accuracy given a correct task assignment and the class given the true task on CoraFull and ogbn-arxiv, and Table 23 gives them on the six other datasets and on Reddit. The class accuracy given a correct task assignment is read on the nodes whose predicted class falls in their own task, and Appendix O states the identity that ties it to the task hit and the accuracy, and what the class given the true task measures instead. The references appear with that column blank: their predicted class names a task, which their task-hit column reads, and within one run the blank cell equals the accuracy divided by the task hit.

Table 22: Accuracy and task hit (share of test nodes whose predicted class belongs to their own task), under Mixed and under HIDE. *Class* | *true task* restricts the argmax to the node's own task; *class* | *predicted task* is the class accuracy given a correct task assignment, read on the nodes whose predicted class falls in their own task (Appendix O). Mixed scores one batch of every task's test nodes on the task subgraphs. A reference's predicted class names a task, which its task-hit column reads; its class accuracy given a correct task assignment is left blank, and within one run equals the accuracy divided by the task hit. Shared graph, mean and standard deviation over seeds 0 to 9; R_1 is not shown. The R_4 rows come from the scoring pass that yields its task hit and differ from Table 21 by at most 0.02 points. Other datasets in Table 23.

		Mixed, per node		HIDE (whole graph)			
	method	Acc	task hit	Acc	task hit	class | predicted task	class | true task
CoraFull	PDGNNs	86.27 ± 0.43	87.54 ± 0.46	74.44 ± 0.64	75.36 ± 0.59	98.78 ± 0.15	97.71 ± 0.21
	DeLoMe	84.15 ± 0.59	85.38 ± 0.54	71.37 ± 1.08	72.30 ± 1.01	98.71 ± 0.23	97.19 ± 0.27
	SSM	81.22 ± 0.85	82.61 ± 0.79	63.16 ± 0.87	64.16 ± 0.78	98.44 ± 0.31	96.31 ± 0.33
	TPP	79.88 ± 0.69	82.51 ± 0.66	55.12 ± 0.62	56.26 ± 0.53	97.97 ± 0.33	96.90 ± 0.16
	TAAM*	60.54 ± 0.90	61.78 ± 0.86	56.55 ± 0.88	57.67 ± 0.85	98.04 ± 0.31	97.18 ± 0.26
	DyMoE*	61.33 ± 1.15	62.40 ± 1.13	60.92 ± 1.13	62.11 ± 1.15	98.09 ± 0.29	88.79 ± 0.63
	R_{prop}	86.74 ± 0.49	88.03 ± 0.53	70.64 ± 0.60	71.40 ± 0.56	—	97.15 ± 0.15
	R_4	86.86 ± 0.48	88.05 ± 0.55	71.29 ± 0.55	71.98 ± 0.56	—	97.74 ± 0.15
ogbn-arxiv	PDGNNs	69.48 ± 0.43	73.13 ± 0.39	63.08 ± 0.24	66.41 ± 0.20	94.97 ± 0.20	93.23 ± 0.24
	DeLoMe	39.38 ± 2.84	43.31 ± 2.81	34.59 ± 2.22	38.64 ± 2.30	89.52 ± 2.21	85.49 ± 1.50
	SSM	69.91 ± 0.92	72.85 ± 0.99	56.08 ± 0.88	58.44 ± 0.86	95.97 ± 0.65	92.87 ± 1.17
	TPP	48.01 ± 0.13	49.28 ± 0.15	32.13 ± 0.15	32.77 ± 0.16	98.03 ± 0.07	97.14 ± 0.07
	TAAM*	45.15 ± 0.32	47.46 ± 0.30	43.28 ± 0.32	45.60 ± 0.29	94.92 ± 0.16	94.79 ± 0.11
	DyMoE*	45.33 ± 1.97	46.78 ± 2.04	42.92 ± 2.01	44.30 ± 2.07	96.88 ± 0.36	89.85 ± 1.15
	R_{prop}	89.95 ± 0.12	91.78 ± 0.11	74.46 ± 0.23	75.58 ± 0.23	—	97.66 ± 0.06
	R_4	89.63 ± 0.11	91.52 ± 0.10	73.63 ± 0.18	74.79 ± 0.18	—	97.51 ± 0.06

Table 23: The six datasets Table 22 does not show, in its columns and form, shared graph, seeds 0 to 9, and Reddit over seeds 0 to 4. SSM on Amazon Computers ends at the classes of the first task on every seed. TAAM and DyMoE are the reconstructions of Appendix H. The references are not shown.

	Mixed, per node		HIDE (whole graph)			
method	Acc	task hit	Acc	task hit	class | predicted task	class | true task
Cora						
PDGNNs	94.41 ± 0.87	96.50 ± 0.76	90.08 ± 1.10	91.74 ± 1.06	98.19 ± 0.69	97.64 ± 0.75
DeLoMe	92.89 ± 1.08	94.83 ± 0.86	88.03 ± 1.58	89.55 ± 1.53	98.30 ± 0.37	97.52 ± 0.47
SSM	90.68 ± 0.82	93.30 ± 0.67	84.30 ± 1.46	86.48 ± 1.20	97.47 ± 0.65	95.51 ± 1.23
TPP	88.05 ± 1.18	90.62 ± 1.03	82.19 ± 1.45	84.00 ± 1.52	97.84 ± 0.30	97.09 ± 0.43
TAAM*	79.10 ± 1.53	81.00 ± 1.46	77.34 ± 1.56	79.12 ± 1.54	97.75 ± 0.55	97.13 ± 0.68
DyMoE*	79.24 ± 2.96	81.43 ± 2.75	78.26 ± 3.05	80.35 ± 2.73	97.39 ± 0.83	94.73 ± 1.51
CiteSeer						
PDGNNs	80.22 ± 1.11	90.76 ± 0.91	76.53 ± 0.96	86.13 ± 0.91	88.86 ± 0.99	85.95 ± 1.13
DeLoMe	79.45 ± 1.01	89.16 ± 0.96	75.65 ± 0.98	84.75 ± 1.06	89.26 ± 0.82	86.33 ± 0.93
SSM	76.39 ± 1.61	87.23 ± 1.57	72.23 ± 1.49	82.37 ± 1.26	87.69 ± 0.96	84.65 ± 0.77
TPP	76.85 ± 1.26	90.27 ± 1.04	71.49 ± 1.05	81.66 ± 0.87	87.56 ± 1.18	84.38 ± 1.04
TAAM*	72.07 ± 1.23	82.47 ± 0.74	71.37 ± 1.23	81.48 ± 1.07	87.60 ± 1.07	85.66 ± 0.93
DyMoE*	62.53 ± 1.88	68.64 ± 1.99	62.44 ± 2.19	68.38 ± 2.14	91.32 ± 0.66	85.04 ± 1.30
Amazon Photo						
PDGNNs	96.59 ± 0.62	98.50 ± 0.30	91.82 ± 1.49	94.22 ± 0.44	97.45 ± 1.59	96.60 ± 1.37
DeLoMe	96.64 ± 0.33	98.35 ± 0.29	89.98 ± 0.91	92.93 ± 0.91	96.82 ± 0.58	96.19 ± 0.49
SSM	88.62 ± 7.72	94.00 ± 4.33	71.48 ± 10.46	81.02 ± 4.04	87.88 ± 9.64	86.67 ± 8.50
TPP	73.78 ± 0.90	75.14 ± 0.75	62.82 ± 1.08	64.17 ± 0.95	97.89 ± 0.53	97.37 ± 0.41
TAAM*	87.05 ± 0.92	88.49 ± 0.74	84.18 ± 1.20	85.88 ± 0.89	98.03 ± 0.57	97.74 ± 0.57
DyMoE*	82.38 ± 6.25	86.00 ± 1.93	82.09 ± 6.09	85.71 ± 1.80	95.73 ± 6.01	93.58 ± 5.93
Amazon Computers						
PDGNNs	94.44 ± 0.42	95.00 ± 0.38	87.13 ± 0.62	87.57 ± 0.64	99.50 ± 0.15	98.24 ± 0.62
DeLoMe	94.12 ± 0.95	94.72 ± 0.92	84.47 ± 1.10	84.98 ± 1.07	99.40 ± 0.15	98.07 ± 0.37
SSM	4.43 ± 3.90	18.74 ± 0.02	4.43 ± 3.90	18.74 ± 0.02	23.65 ± 20.85	70.25 ± 2.53
TPP	72.76 ± 0.65	73.08 ± 0.64	55.83 ± 0.92	56.16 ± 0.93	99.42 ± 0.11	99.12 ± 0.12
TAAM*	78.28 ± 0.86	78.55 ± 0.86	75.02 ± 1.03	76.00 ± 0.85	98.71 ± 0.49	97.96 ± 0.53
DyMoE*	69.59 ± 6.52	71.19 ± 6.45	68.72 ± 6.12	70.39 ± 6.07	97.62 ± 0.77	91.60 ± 1.79
WikiCS						
PDGNNs	88.73 ± 0.65	94.42 ± 0.56	80.62 ± 0.99	85.66 ± 0.88	94.11 ± 0.44	92.73 ± 0.31
DeLoMe	88.53 ± 0.67	94.71 ± 0.71	78.79 ± 0.67	84.88 ± 0.72	92.82 ± 0.95	91.43 ± 0.89
SSM	85.10 ± 2.58	94.87 ± 0.65	61.11 ± 4.29	71.01 ± 3.04	86.00 ± 3.63	79.75 ± 3.87
TPP	64.74 ± 0.84	69.26 ± 0.92	42.66 ± 0.74	46.06 ± 0.65	92.62 ± 0.59	92.91 ± 0.40
TAAM*	70.30 ± 0.88	74.72 ± 0.77	66.28 ± 0.93	71.50 ± 0.75	92.71 ± 0.62	92.24 ± 0.58
DyMoE*	79.92 ± 1.66	84.76 ± 1.41	73.45 ± 1.63	77.91 ± 1.44	94.26 ± 0.82	90.47 ± 0.81
Roman-empire						
PDGNNs	60.21 ± 0.67	66.02 ± 0.73	32.51 ± 1.25	40.20 ± 0.96	80.84 ± 1.55	82.80 ± 0.90
DeLoMe	51.34 ± 1.40	60.28 ± 1.30	26.43 ± 1.33	35.32 ± 1.23	74.83 ± 2.95	74.46 ± 2.76
SSM	48.91 ± 0.50	56.82 ± 0.66	25.13 ± 1.38	31.65 ± 1.17	79.36 ± 2.09	76.56 ± 1.30
TPP	70.89 ± 0.32	81.32 ± 0.38	12.16 ± 0.44	17.63 ± 0.60	68.97 ± 1.75	71.26 ± 0.83
TAAM*	44.95 ± 0.86	53.04 ± 0.77	32.54 ± 0.74	46.54 ± 0.69	69.91 ± 1.24	75.87 ± 1.06
DyMoE*	40.48 ± 2.45	43.07 ± 2.46	35.80 ± 3.77	38.06 ± 3.82	94.04 ± 1.10	81.51 ± 1.51
Reddit, seeds 0 *to* 4						
PDGNNs	96.17 ± 0.14	96.43 ± 0.16	91.59 ± 0.15	91.86 ± 0.14	99.71 ± 0.03	98.88 ± 0.05
DeLoMe	97.11 ± 0.40	97.35 ± 0.40	90.30 ± 0.82	90.53 ± 0.82	99.74 ± 0.03	98.64 ± 0.13
SSM	70.51 ± 9.67	72.48 ± 10.95	40.41 ± 3.41	41.58 ± 3.62	97.23 ± 1.46	86.34 ± 4.05
TPP	72.33 ± 0.26	72.60 ± 0.26	54.34 ± 0.11	54.46 ± 0.11	99.77 ± 0.03	97.85 ± 0.16
TAAM*	52.54 ± 0.18	52.68 ± 0.18	49.05 ± 0.23	49.56 ± 0.24	98.97 ± 0.11	98.62 ± 0.10
DyMoE*	75.03 ± 0.78	75.65 ± 0.78	65.38 ± 0.40	66.19 ± 0.32	98.78 ± 0.21	92.37 ± 0.38

Q THE TASK SIZE

Table 24 gives the six methods at five task sizes on CoraFull and five on ogbn-arxiv, each retrained at its released configuration and scored under HIDE, in the three readings of Table 22. The assignment of classes to tasks stays in class-index order and only the number of classes a task takes changes, so a column is the same graph and the same split cut into fewer and larger tasks. The cascade is not retrained at any size other than two classes per task. The six other datasets were also run at one further size each; those runs are released with the logs and enter no table.

Table 25 carries the one size at which the three settings were re-measured rather than HIDE alone.

Table 24: The six methods under HIDE at five task sizes per dataset, on the shared graph, per node, mean and standard deviation. *Accuracy* is the per-node accuracy, *task hit* the share of test nodes whose predicted class belongs to their own task, and *class | true task* the accuracy with the argmax restricted to the classes of the node's own task. The seeds row says how many seeds each column rests on; where it gives two numbers, the smaller is TAAM and DyMoE and the larger the four released implementations, except at five classes per task on ogbn-arxiv, where DyMoE holds five seeds and TAAM three. Two classes per task is the size every other table in this paper reports, except Table 25. TAAM and DyMoE are the reconstructions of Appendix H.

CoraFull, classes per task					
method	2	5	7	10	14
seeds	10	5	3/5	3/5	3/5
accuracy					
PDGNNs	74.44 ± 0.64	74.01 ± 0.54	73.65 ± 0.60	73.64 ± 0.59	73.16 ± 0.78
DeLoMe	71.37 ± 1.08	70.82 ± 0.53	70.94 ± 0.58	70.52 ± 0.91	70.71 ± 0.47
SSM	63.16 ± 0.87	62.44 ± 0.49	61.90 ± 0.69	61.28 ± 1.19	62.07 ± 0.55
TPP	55.12 ± 0.62	47.90 ± 0.63	45.71 ± 0.73	42.56 ± 0.67	43.63 ± 0.49
TAAM*	56.55 ± 0.88	46.57 ± 1.00	45.74 ± 0.68	44.36 ± 0.73	46.61 ± 0.55
DyMoE*	60.92 ± 1.13	61.18 ± 1.07	61.03 ± 0.70	61.82 ± 0.57	62.12 ± 0.39
task hit					
PDGNNs	75.36 ± 0.59	76.55 ± 0.50	77.22 ± 0.64	78.79 ± 0.49	80.09 ± 0.77
DeLoMe	72.30 ± 1.01	73.35 ± 0.55	74.66 ± 0.71	75.96 ± 0.74	78.27 ± 0.70
SSM	64.16 ± 0.78	65.07 ± 0.59	66.03 ± 0.79	67.37 ± 0.98	70.01 ± 0.88
TPP	56.26 ± 0.53	51.52 ± 0.67	50.76 ± 0.78	51.30 ± 0.65	54.93 ± 0.76
TAAM*	57.67 ± 0.85	49.70 ± 1.07	49.82 ± 1.13	49.93 ± 0.57	53.91 ± 0.53
DyMoE*	62.11 ± 1.15	64.73 ± 1.03	66.23 ± 1.05	69.10 ± 0.75	72.12 ± 0.19
class | true task					
PDGNNs	97.71 ± 0.21	94.38 ± 0.19	92.55 ± 0.28	90.32 ± 0.26	88.00 ± 0.43
DeLoMe	97.19 ± 0.27	93.59 ± 0.37	91.56 ± 0.28	89.10 ± 0.53	86.42 ± 0.31
SSM	96.31 ± 0.33	90.79 ± 0.23	87.79 ± 0.63	84.07 ± 0.68	81.47 ± 0.55
TPP	96.90 ± 0.16	90.74 ± 0.41	85.90 ± 0.69	80.10 ± 0.65	76.50 ± 0.58
TAAM*	97.18 ± 0.26	92.03 ± 0.28	89.73 ± 0.48	87.00 ± 0.40	85.03 ± 0.88
DyMoE*	88.79 ± 0.63	78.43 ± 0.17	76.61 ± 0.49	75.44 ± 0.37	73.86 ± 0.86
ogbn-arxiv, classes per task					
method	2	4	5	8	10
seeds	10	3/5	3/5	3/5	3/5
accuracy					
PDGNNs	63.08 ± 0.24	62.83 ± 0.26	62.67 ± 0.23	62.07 ± 0.15	61.37 ± 0.19
DeLoMe	34.59 ± 2.22	43.09 ± 1.50	43.52 ± 1.97	52.39 ± 0.86	48.67 ± 1.25
SSM	56.08 ± 0.88	53.34 ± 0.98	53.25 ± 1.44	49.71 ± 2.66	41.17 ± 6.18
TPP	32.13 ± 0.15	30.73 ± 0.16	29.02 ± 0.15	35.35 ± 0.36	30.45 ± 0.07
TAAM*	43.28 ± 0.32	40.21 ± 0.03	44.12 ± 0.51	40.13 ± 0.07	42.06 ± 0.41
DyMoE*	42.92 ± 2.01	46.26 ± 1.82	48.58 ± 2.26	51.98 ± 0.42	54.16 ± 0.38
task hit					
PDGNNs	66.41 ± 0.20	67.45 ± 0.19	67.39 ± 0.11	71.59 ± 0.14	71.80 ± 0.18
DeLoMe	38.64 ± 2.30	47.63 ± 1.74	48.96 ± 1.95	61.18 ± 0.78	58.66 ± 1.07
SSM	58.44 ± 0.86	57.42 ± 0.99	57.16 ± 0.84	60.57 ± 1.80	53.05 ± 4.61
TPP	32.77 ± 0.16	31.90 ± 0.16	30.79 ± 0.20	39.81 ± 0.36	36.56 ± 0.13
TAAM*	45.60 ± 0.29	43.47 ± 0.16	48.22 ± 0.26	47.24 ± 0.22	51.68 ± 0.13
DyMoE*	44.30 ± 2.07	49.12 ± 1.77	51.63 ± 2.19	59.60 ± 0.46	61.96 ± 0.40
class | true task					
PDGNNs	93.23 ± 0.24	89.50 ± 0.21	88.67 ± 0.21	82.53 ± 0.23	80.15 ± 0.21
DeLoMe	85.49 ± 1.50	82.05 ± 0.73	79.57 ± 1.17	79.07 ± 0.69	75.96 ± 0.70
SSM	92.87 ± 1.17	86.60 ± 1.27	85.22 ± 1.43	74.66 ± 1.25	63.00 ± 5.49
TPP	97.14 ± 0.07	93.43 ± 0.05	92.09 ± 0.13	84.07 ± 0.19	81.65 ± 0.12
TAAM*	94.79 ± 0.11	91.02 ± 0.24	90.34 ± 0.40	84.21 ± 0.20	80.72 ± 0.55
DyMoE*	89.85 ± 1.15	83.71 ± 0.32	83.25 ± 0.79	77.53 ± 0.33	77.47 ± 0.44

Table 25: CoraFull with five classes per task, shared graph, seeds 0 to 4: accuracy under the three settings, each under its own aggregation as in Table 2, and under HIDE the task hit and the class given the true task of Table 22. TAAM and DyMoE were run at this size on the same seeds; Table 24 gives their HIDE accuracy.

method	Isolated, AA	Mixed	HIDE	task hit, HIDE	class | true task
PDGNNs	80.39 ± 0.52	85.15 ± 0.39	74.01 ± 0.54	76.55 ± 0.50	94.38 ± 0.19
DeLoMe	79.03 ± 0.56	82.69 ± 0.45	70.82 ± 0.53	73.35 ± 0.55	93.59 ± 0.37
SSM	75.86 ± 1.28	79.90 ± 0.72	62.44 ± 0.49	65.07 ± 0.59	90.79 ± 0.23
TPP	86.36 ± 0.74	64.61 ± 0.55	47.90 ± 0.63	51.52 ± 0.67	90.74 ± 0.41

R HOMOPHILY OF THE TASK SUBGRAPHS

As Zhao et al. (2024) showed for task subgraphs, which on CoraFull they measure at a label homophily of 0.99 against 0.57 for the full graph, and as Donghi et al. (2025) note of protocols that ignore inter-task edges, scoring on the task subgraphs raises the homophily of the graph a classifier sees. With class-disjoint tasks the reason is exact: every inter-task edge joins two classes, so each edge the task subgraphs delete is heterophilous. On the shared graph, the homophily over the edges the task subgraphs keep is 0.982 against 0.567 on CoraFull, 0.982 against 0.654 on ogbn-arxiv, 0.872 to 0.993 on the seven other homophilous datasets of Table 6, and 0.435 against 0.047 on Roman-empire. HIDE scores each node at the homophily its dataset has, and the cost of the inter-task edges (Section 5.1) is paid at that homophily.

Lowering the homophily of one graph while every node keeps its degree costs each of the six methods, the cascade and each reference accuracy under HIDE, with every seed agreeing at every rung on CoraFull and at 0.15 and 0.07 on ogbn-arxiv (Table 27). The graphs are built by swapping pairs of same-class edges, so the edge count and each node's degree stay as they were; the rewiring of Ma et al. (2022) adds edges instead, and changes density with homophily. From 0.567 to 0.05 on CoraFull the losses run from 8.94 points (TAAM) to 42.61 (TPP), and from 0.654 to 0.15 on ogbn-arxiv from 9.96 (DeLoMe) to 44.07 (R_4) (Table 28). The sweep rewires the graph each run trains on as well as the one it is scored on, and in the graphs reported the share of crossing edges rose as the homophily fell, from 0.423 to 0.934 on CoraFull and from 0.334 to 0.916 on ogbn-arxiv (Table 26), so it measures what a whole run pays for homophily, its training graph included.

Deleting same-class edges costs more than deleting the same number of edges at unchanged homophily. Pruning CoraFull to 0.15 costs PDGNNs 15.43 points, of which a graph thinned by the same number of edges at unchanged homophily accounts for 4.06; for every classifier the pruning arm was run on, on CoraFull, and for every one but DeLoMe on ogbn-arxiv, the additional effect of class-selective removal relative to the equal-edge-count control exceeds the control's own loss at every rung (Table 28). The equal edge count holds density, while the two graphs still differ in degree allocation, cross-task connectivity and the rest of their topology, so the residual is a contrast between two interventions, of which the change in edge homophily is one part. DeLoMe on ogbn-arxiv gains from fewer edges by 9.94 points at 0.15 with the homophily unchanged. The residual and the loss on the rewired arm, where every node keeps its degree, agree while the loss is small and separate as it grows: for PDGNNs on CoraFull they give -4.17 and -3.89 at 0.45, and -13.14 and -25.22 at 0.05.

The order of the methods changes with homophily on both graphs, and on CoraFull so does the set of methods above R_{prop} (Table 27, means over five seeds). On CoraFull TAAM moves from fifth of the six methods at 0.567 to second at 0.05, and the methods whose mean exceeds that of R_{prop} change from PDGNNs and DeLoMe to PDGNNs, TAAM and DyMoE. On ogbn-arxiv R_{prop} stays above every method at every rung, while SSM falls from second to fifth at 0.15. A ranking read on one graph therefore holds at that graph's homophily, and a ranking read on the task subgraphs holds at 0.982, a homophily neither full graph has.

Where the homophily is removed matters as well as how much. At a homophily and a crossing share matched to within 0.004 between the two arms, 0.356 against 0.360 and 0.630 against 0.627 on CoraFull and 0.481 against 0.480 and 0.507 on both on ogbn-arxiv, taking the homophily out of the nodes whose neighbors all shared their label costs more than spreading the same loss over

Table 26: What each graph became. *Rewired* swaps pairs of same-class edges until the target edge homophily is reached, which leaves the degree of every node untouched; *pruned* deletes same-class edges to the same target, which costs edges; *thinned* deletes exactly as many edges as the pruning at that target while leaving the homophily where it was. In the graphs below, lowering the homophily raised the share of edges that cross a task with it, on the rewired arm from 0.423 to 0.934 on CoraFull and from 0.334 to 0.916 on ogbn-arxiv, with the pruned arm within 0.02 of it at every target; that share is therefore reported rather than held. The last two rows of each panel take the homophily out of the nodes whose neighbors all shared their label, against a uniform rewiring stopped at an approximately matched homophily and crossing share: 0.356 against 0.360 and 0.630 against 0.627 on CoraFull, 0.481 against 0.480 and 0.507 against 0.507 on ogbn-arxiv. Edges are counted once here and in both directions in Table 6.

graph	homophily	edges	degree	isolated test	crossing	pure nodes
CoraFull						
original	0.567	63,421	6.41	0	0.423	3953
rewired to 0.45	0.450	63,421	6.41	0	0.538	1780
pruned to 0.45	0.450	49,925	5.04	200	0.537	2751
thinned to 0.45	0.567	49,925	5.04	190	0.422	3774
rewired to 0.30	0.300	63,421	6.41	0	0.687	587
pruned to 0.30	0.300	39,227	3.96	502	0.683	1436
thinned to 0.30	0.567	39,227	3.96	411	0.423	3557
rewired to 0.15	0.150	63,421	6.41	0	0.835	103
pruned to 0.15	0.150	32,304	3.26	858	0.830	457
thinned to 0.15	0.567	32,304	3.26	574	0.423	3321
rewired to 0.05	0.050	63,421	6.41	0	0.934	9
pruned to 0.05	0.050	28,904	2.92	1070	0.927	80
thinned to 0.05	0.567	28,904	2.92	707	0.423	3140
pockets emptied	0.356	63,421	6.41	0	0.630	0
rewired, matched	0.360	63,421	6.41	0	0.627	946
ogbn-arxiv						
original	0.654	1,157,799	13.67	0	0.334	35895
rewired to 0.45	0.450	1,157,799	13.67	0	0.537	6696
pruned to 0.45	0.450	727,898	8.60	2302	0.531	21939
thinned to 0.45	0.654	727,898	8.60	2224	0.334	36217
rewired to 0.30	0.300	1,157,799	13.67	0	0.686	1811
pruned to 0.30	0.300	571,920	6.75	4222	0.676	13035
thinned to 0.30	0.654	571,920	6.75	3774	0.334	35493
rewired to 0.15	0.150	1,157,799	13.67	0	0.836	242
pruned to 0.15	0.150	470,992	5.56	6466	0.821	5041
thinned to 0.15	0.654	470,992	5.56	5009	0.334	34429
rewired to 0.07	0.070	1,157,799	13.67	0	0.916	62
pruned to 0.07	0.070	430,477	5.08	7962	0.898	1475
thinned to 0.07	0.654	430,477	5.08	5677	0.334	33891
pockets emptied	0.481	1,157,799	13.67	0	0.507	1
rewired, matched	0.480	1,157,799	13.67	0	0.507	8413

the graph, for every method and reference on CoraFull, from 1.75 to 10.48 points, and for all but DeLoMe on ogbn-arxiv, where it gains 7.85 (Table 29).

Table 27: Accuracy per node under HIDE as one graph loses homophily with every node's degree held: same-class edge pairs are swapped until the edge homophily in the header is reached (Table 26). Mean and standard deviation over seeds 0 to 4; the first column is the graph as it ships. TPP was not run at 0.07 on ogbn-arxiv. EWC, MAS, TWP, LwF, GEM, ER-GNN and fine-tuning are omitted: on the graph as it ships they score at most 4.46% on CoraFull and, all but ER-GNN, at most 2.06% on ogbn-arxiv, where ER-GNN's 21.07% carries a standard deviation of 11.36 over the five seeds.

CoraFull method	0.567	0.45	0.30	0.15	0.05
PDGNNs	74.43 ± 0.50	70.54 ± 0.40	64.36 ± 0.62	56.05 ± 0.97	49.21 ± 0.85
DeLoMe	70.93 ± 0.80	66.67 ± 0.72	58.13 ± 0.52	43.97 ± 0.92	36.72 ± 3.02
SSM	62.84 ± 0.70	55.14 ± 0.89	45.25 ± 0.82	34.59 ± 0.56	29.06 ± 0.63
TPP	54.83 ± 0.61	46.37 ± 0.70	30.29 ± 1.13	16.91 ± 1.07	12.22 ± 0.58
TAAM*	56.01 ± 0.65	54.21 ± 0.58	51.47 ± 0.57	48.72 ± 0.67	47.07 ± 0.81
DyMoE*	60.16 ± 0.70	58.05 ± 0.93	54.38 ± 0.54	48.00 ± 1.12	42.38 ± 0.75
cascade	71.97 ± 0.74	67.29 ± 0.71	58.44 ± 0.79	45.63 ± 0.97	34.63 ± 0.99
R_1	72.94 ± 0.56	68.65 ± 0.67	61.83 ± 0.70	52.44 ± 0.85	45.64 ± 0.84
R_{prop}	70.53 ± 0.51	65.02 ± 0.48	57.36 ± 0.50	47.89 ± 0.66	41.09 ± 0.62
R_4	71.14 ± 0.45	65.30 ± 0.32	55.46 ± 0.44	42.30 ± 0.57	31.61 ± 1.82
ogbn-arxiv method	0.654	0.45	0.30	0.15	0.07
PDGNNs	63.07 ± 0.33	55.58 ± 0.57	46.00 ± 0.80	34.98 ± 1.47	32.82 ± 1.28
DeLoMe	34.87 ± 2.13	32.75 ± 3.95	33.97 ± 2.62	24.91 ± 1.29	23.07 ± 1.19
SSM	55.91 ± 1.19	35.38 ± 2.43	25.16 ± 3.44	18.99 ± 3.82	22.74 ± 3.43
TPP	32.20 ± 0.12	19.97 ± 0.08	17.28 ± 0.04	16.51 ± 0.02	–
TAAM*	43.34 ± 0.43	38.37 ± 0.49	34.04 ± 0.42	26.86 ± 0.99	27.32 ± 1.26
DyMoE*	42.58 ± 2.09	42.11 ± 1.04	34.01 ± 1.61	30.17 ± 0.36	30.34 ± 1.99
cascade	73.65 ± 0.28	65.80 ± 0.20	56.25 ± 0.26	43.39 ± 0.69	36.59 ± 0.48
R_1	67.41 ± 0.27	55.12 ± 0.72	40.77 ± 1.41	28.72 ± 0.72	27.58 ± 0.58
R_{prop}	74.48 ± 0.25	66.35 ± 0.14	56.35 ± 0.30	44.65 ± 0.25	41.09 ± 0.21
R_4	73.70 ± 0.25	62.88 ± 0.23	46.52 ± 0.18	29.62 ± 0.19	26.29 ± 0.32

Table 28: CoraFull, from edge homophily 0.567, and ogbn-arxiv, from 0.654, taken down as paired differences over seeds 0 to 4 in percentage points of accuracy under HIDE. For each dataset the first column is the whole effect of pruning, the second the effect of thinning the graph by the same number of edges at unchanged homophily, and the third their difference, the residual: the additional effect of class-selective edge removal relative to the equal-edge-count control. The fourth is the loss on the rewired arm, where the degree of every node is held exactly. The last rung is 0.05 on CoraFull and 0.07 on ogbn-arxiv. A dagger marks a difference on whose sign the seeds do not all agree; a dash marks a method the pruning and thinning arms were not run on, and TPP was not run at 0.07 on ogbn-arxiv. The CGLB zoo and plain fine-tuning are omitted, as in Table 27.

	CoraFull				ogbn-arxiv			
method	pruning, all of it	of it: density	of it: residual	homophily, degrees held	pruning, all of it	of it: density	of it: residual	homophily, degrees held
edge homophily 0.45								
PDGNNs	-5.47	-1.31	-4.17	-3.89	-6.52	-2.65	-3.87	-7.49
DeLoMe	-5.23	-0.79	-4.44	-4.26	$+8.13$	$+3.05^{\dagger}$	$+5.08$	$-2.12^{\dagger}$
SSM	-4.33	-0.64	-3.69	-7.70	-7.13	$-0.99^{\dagger}$	-6.14	-20.53
TPP	–	–	–	-8.46	–	–	–	-12.22
TAAM*	–	–	–	-1.80	–	–	–	-4.97
DyMoE*	–	–	–	-2.11	–	–	–	$-0.47^{\dagger}$
cascade	-7.02	-1.99	-5.03	-4.68	-8.10	-2.20	-5.90	-7.86
R_1	-5.38	-1.14	-4.23	-4.28	-5.97	-1.92	-4.05	-12.29
R_{prop}	-8.43	-2.33	-6.10	-5.51	-8.81	-2.54	-6.27	-8.14
R_4	-7.41	-1.57	-5.84	-5.84	-9.27	-2.42	-6.85	-10.81
edge homophily 0.30								
PDGNNs	-11.02	-2.65	-8.37	-10.07	-10.02	-4.16	-5.86	-17.06
DeLoMe	-10.44	-2.40	-8.03	-12.81	$+14.08$	$+6.44$	$+7.63$	$-0.90^{\dagger}$
SSM	-8.58	-1.20	-7.38	-17.60	-11.13	-2.45	-8.68	-30.75
TPP	–	–	–	-24.54	–	–	–	-14.92
TAAM*	–	–	–	-4.55	–	–	–	-9.30
DyMoE*	–	–	–	-5.78	–	–	–	-8.58
cascade	-15.32	-4.41	-10.91	-13.53	-12.80	-3.65	-9.15	-17.41
R_1	-10.51	-2.23	-8.28	-11.11	-8.97	-3.08	-5.90	-26.64
R_{prop}	-18.78	-4.19	-14.59	-13.17	-14.07	-4.01	-10.06	-18.13
R_4	-15.33	-3.25	-12.08	-15.67	-14.49	-3.88	-10.61	-27.18
edge homophily 0.15								
PDGNNs	-15.43	-4.06	-11.37	-18.38	-12.76	-5.33	-7.43	-28.08
DeLoMe	-14.79	-3.44	-11.35	-26.96	$+14.52$	$+9.94$	$+4.58$	-9.96
SSM	-11.98	-1.56	-10.42	-28.25	-12.63	-3.73	-8.89	-36.92
TPP	–	–	–	-37.92	–	–	–	-15.69
TAAM*	–	–	–	-7.29	–	–	–	-16.48
DyMoE*	–	–	–	-12.17	–	–	–	-12.41
cascade	-24.48	-6.20	-18.29	-26.33	-18.25	-4.82	-13.43	-30.27
R_1	-15.10	-3.29	-11.81	-20.50	-11.47	-4.03	-7.44	-38.69
R_{prop}	-28.27	-5.84	-22.43	-22.64	-19.03	-5.13	-13.90	-29.84
R_4	-21.50	-4.48	-17.02	-28.83	-19.36	-4.89	-14.48	-44.07
edge homophily 0.05 on CoraFull, 0.07 on ogbn-arxiv								
PDGNNs	-17.89	-4.75	-13.14	-25.22	-14.52	-6.09	-8.43	-30.25
DeLoMe	-16.93	-4.29	-12.64	-34.21	$+11.59$	$+9.81$	$+1.78^{\dagger}$	-11.80
SSM	-13.20	-2.01	-11.18	-33.78	-13.74	-4.27	-9.47	-33.17
TPP	–	–	–	-42.61	–	–	–	–
TAAM*	–	–	–	-8.94	–	–	–	-16.02
DyMoE*	–	–	–	-17.78	–	–	–	-12.25
cascade	-29.14	-7.37	-21.77	-37.34	-21.88	-5.40	-16.48	-37.07
R_1	-16.93	-3.75	-13.18	-27.30	-11.94	-4.38	-7.55	-39.83
R_{prop}	-34.34	-7.07	-27.27	-29.44	-21.58	-5.74	-15.84	-33.39
R_4	-24.17	-5.23	-18.94	-39.52	-21.65	-5.46	-16.19	-47.40

Table 29: Where the homophily is taken from, at an edge homophily and a crossing share matched to within 0.004 between the two arms (Table 26). *Pockets emptied* spends the swaps on the nodes whose neighbors all shared their label; *uniform* is the same rule applied everywhere and stopped at approximately the same homophily, 0.360 against 0.356 on CoraFull and 0.480 against 0.481 on ogbn-arxiv. Accuracies are per-node under HIDE over seeds 0 to 4, the difference paired per seed.

method	pockets emptied	uniform	difference
CoraFull			
PDGNNs	61.35 ± 0.40	67.06 ± 0.65	-5.70
DeLoMe	53.19 ± 0.96	62.42 ± 0.61	-9.24
SSM	43.46 ± 0.39	48.94 ± 0.92	-5.48
TPP	34.58 ± 0.57	37.48 ± 0.83	-2.91
TAAM*	50.65 ± 0.57	52.46 ± 0.58	-1.82
DyMoE*	54.87 ± 0.33	56.63 ± 0.35	-1.75
cascade	56.00 ± 0.70	62.44 ± 1.00	-6.44
R_1	57.56 ± 0.89	64.69 ± 0.54	-7.13
R_{prop}	56.47 ± 0.81	60.35 ± 0.65	-3.88
R_4	49.13 ± 0.69	59.61 ± 0.39	-10.48
ogbn-arxiv			
PDGNNs	51.56 ± 0.46	57.13 ± 0.47	-5.57
DeLoMe	38.54 ± 0.98	30.69 ± 3.21	$+7.85$
SSM	35.23 ± 2.89	39.07 ± 0.54	-3.84
TPP	19.50 ± 0.14	20.90 ± 0.08	-1.40
TAAM*	37.85 ± 0.36	39.42 ± 0.60	-1.57
DyMoE*	36.03 ± 1.36	41.04 ± 1.88	-5.02
cascade	59.39 ± 0.15	67.39 ± 0.11	-8.00
R_1	51.91 ± 0.21	57.35 ± 0.65	-5.43
R_{prop}	58.13 ± 0.20	67.81 ± 0.15	-9.68
R_4	52.68 ± 0.12	65.04 ± 0.16	-12.35

S FORGETTING

Forgetting is the accuracy a task loses between the moment it was learned and the end of the sequence (Zhang et al., 2022a): for task t, the per-node accuracy of its test nodes under the final model minus that under the checkpoint saved when task t ended, averaged over the tasks before the last. The two models have to be scored on the same graph, or the difference mixes what the model forgot with the neighbors that arrived after task t. CGLB scores both on the task's own subgraph, $G[V_t]$. We also score both on the graph grown to task t, $G[V_1 \cup \cdots \cup V_t]$, which is the graph in scope when the task was learned and the one that holds every node the model had seen. Table 30 gives both readings beside the accuracy under Isolated and under HIDE, over seeds 0 to 9.

Table 30: Forgetting of the three replay methods, mean and standard deviation over seeds 0 to 9: final model minus the checkpoint saved after the task, per node, averaged over the tasks before the last, negative being loss. *Own subgraph* scores both on $G[V_t]$, as CGLB does; *graph grown to the task* scores both on the graph in scope when the task was learned. The accuracy under Isolated is the AA of Section 3.1, per task and per class, and the accuracy under HIDE is per node, both as in Tables 2 and 8 on the same seeds, from which a value can differ by 0.01 through rounding; forgetting is per node.

		accuracy		forgetting	
	method	Isolated	HIDE	own subgraph	graph grown to the task
ogbn-arxiv, own graph	PDGNNs	52.20 ± 0.58	47.18 ± 0.31	-13.50 ± 0.60	-14.38 ± 0.43
	DeLoMe	48.72 ± 0.67	44.58 ± 0.47	-10.73 ± 0.76	-12.26 ± 0.37
	SSM	44.62 ± 0.65	44.56 ± 0.73	$+22.35 \pm 0.86$	$+25.81 \pm 0.94$
ogbn-arxiv, shared graph	PDGNNs	65.61 ± 0.56	63.08 ± 0.24	-11.50 ± 0.52	-11.96 ± 0.44
	DeLoMe	52.82 ± 0.94	34.59 ± 2.22	-16.77 ± 1.63	-19.16 ± 1.51
	SSM	56.35 ± 1.11	56.08 ± 0.88	$+23.79 \pm 1.67$	$+27.04 \pm 1.71$
CoraFull, own graph	PDGNNs	81.56 ± 0.67	72.85 ± 0.81	-3.69 ± 0.38	-3.97 ± 0.38
	DeLoMe	81.76 ± 0.69	72.89 ± 0.84	-3.11 ± 0.71	-3.35 ± 0.44
	SSM	77.78 ± 1.32	63.58 ± 0.74	$+7.52 \pm 1.03$	$+11.01 \pm 0.81$
CoraFull, shared graph	PDGNNs	81.39 ± 0.55	74.44 ± 0.64	-2.51 ± 0.36	-2.79 ± 0.29
	DeLoMe	80.34 ± 0.58	71.37 ± 1.08	-4.79 ± 0.50	-6.57 ± 0.32
	SSM	76.50 ± 0.91	63.16 ± 0.87	-4.73 ± 0.71	-5.10 ± 0.76

The two readings agree to within 3.5 points in every cell, and the grown graph gives the larger magnitude in every cell. Forgetting and the cost of the inter-task edges, both read per node, do not order the cells alike: under its own graph PDGNNs forgets 14.38 points on ogbn-arxiv and pays 1.94 for the edges, while on CoraFull it forgets 3.97 and pays 12.19.

Under its own graph SSM does not forget on either dataset: its earlier tasks are more accurate at the end of the sequence than when they were learned, by 25.81 points on ogbn-arxiv and 11.01 on CoraFull. The sign changes with the graph choice: under the shared graph of CoraFull it is -5.10.

TPP and the cascade overwrite no per-task parameter: TPP freezes a prompt per task and the cascade one network per task, neither is updated once its task ends, and CGLB's own reading of TPP gives 0.0. Their accuracy on a task can still fall over the sequence, because the step that identifies a node's task changes as tasks are added, so for these two methods the difference defined above measures a loss of task identification, the step of Section 5.2, and no loss inside the frozen components. We read it with the graph held, as in Table 30: the model after task t and the final model, both scored on the graph grown to task t. For TPP the model after task t is the prompts and prototypes of tasks 1 to t, none of which the final model alters, so the nodes it loses are those the final model assigns to a later prototype. For the cascade the model after task t is the networks and subspace bases of tasks 1 to t with the label-memory router fit on the labels of those tasks; the final model keeps those networks, bases and the same diffusion, and refits the class head on the labels of every task, so a node can move to a later task or between two of the first t, and the difference counts both. That loss of task identification is 1.86 points on CoraFull and 6.87 on ogbn-arxiv for the cascade, and 3.94 and 15.48 for TPP under the shared graph, over seeds 0 to 9.

T EVERY METHOD AND THE CASCADE AGAINST THE REFERENCES

Table 31 gives each method, and the cascade, minus each of the three references under HIDE, paired per seed over seeds 0 to 9. On CoraFull PDGNNs is above all three references on every seed, and DeLoMe is below R_1 on every seed and above R_{prop} and R_4 by 0.72 and 0.09 points, with seeds that do not all agree on either sign. On ogbn-arxiv all six sit below all three, and the means of the six remain 30.95 points apart.

The last three rows compare each replay method with a reference that retains as many labels as the method does. R^b_{prop} is R_{prop} fitted on at most b training labels per class, which are also the only labels its correct-and-smooth step reads, and b is the method's own per-class buffer: 60 for DeLoMe, 100 for SSM and 400 for PDGNNs (Appendix I). It equates the labels retained; the two classifiers still differ in training procedure and parameter count. DeLoMe and PDGNNs are above it on CoraFull, by 7.46 and 4.90 points, SSM is below it there by 3.28, and all three are below on ogbn-arxiv, by 2.83 to 21.74, each on every seed.

Table 31: Each method, and the cascade, minus each reference under HIDE, as paired differences over seeds 0 to 9 in percentage points; a dagger marks a cell whose seeds do not all agree on its sign. The last three rows are each replay method minus R^b_{prop} at its own per-class buffer b. TAAM and DyMoE are the reconstructions of Appendix H.

difference	CoraFull	ogbn-arxiv
PDGNNs $- R_1$	$+1.23$	-4.39
DeLoMe $- R_1$	-1.83	-32.88
SSM $- R_1$	-10.04	-11.39
TPP $- R_1$	-18.08	-35.34
TAAM* $- R_1$	-16.66	-24.19
DyMoE* $- R_1$	-12.29	-24.55
cascade $- R_1$	-1.36	$+6.08$
PDGNNs $- R_{\text{prop}}$	$+3.79$	-11.38
DeLoMe $- R_{\text{prop}}$	$+0.72^\dagger$	-39.87
SSM $- R_{\text{prop}}$	-7.49	-18.38
TPP $- R_{\text{prop}}$	-15.52	-42.33
TAAM* $- R_{\text{prop}}$	-14.10	-31.18
DyMoE* $- R_{\text{prop}}$	-9.73	-31.54
cascade $- R_{\text{prop}}$	$+1.20$	-0.91
PDGNNs $- R_4$	$+3.16$	-10.55
DeLoMe $- R_4$	$+0.09^\dagger$	-39.04
SSM $- R_4$	-8.12	-17.55
TPP $- R_4$	-16.15	-41.50
TAAM* $- R_4$	-14.73	-30.35
DyMoE* $- R_4$	-10.36	-30.72
cascade $- R_4$	$+0.57^\dagger$	$-0.08^\dagger$
DeLoMe $- R^{60}_{\text{prop}}$	$+7.46$	-21.74
SSM $- R^{100}_{\text{prop}}$	-3.28	-3.63
PDGNNs $- R^{400}_{\text{prop}}$	$+4.90$	-2.83

U INTERVALS OF THE PAIRED DIFFERENCES

Table 32 repeats every method row and the cascade row of Table 20, and every row of Table 7, on CoraFull and ogbn-arxiv, and adds each method minus each reference; the R_1, R_{prop} and R_4 rows of Table 20 are not repeated. Each cell carries a percentile bootstrap interval of the paired mean at a nominal level of 95%: the per-seed differences are resampled $10{,}000$ times with replacement, at a pinned generator seed, and the interval runs from the 2.5th to the 97.5th percentile of the resampled means, the same interval Appendix V gives for the references. Ten paired seeds support a direction and an interval. Two runs at one seed do not repeat an accuracy exactly, by 0.08 to 0.16 points for the replay methods on CoraFull (Appendix B) and by less than 0.7 points in a ten-seed mean for a reference, two collapsed seeds of R_1 set aside (Appendix N), so a difference is read from its interval and its seeds.

Table 32: Every method row and the cascade row of Table 20, and every row of Table 7, on CoraFull and ogbn-arxiv (the R_1, R_{prop} and R_4 rows of Table 20 are not repeated), the per-node sum of the batch and the edges, the graph choice of TAAM and DyMoE over the five seeds of their own-graph runs (marked $^{(5)}$), and the spread of the six methods, with a percentile bootstrap interval of each paired mean, over seeds 0 to 9, and the differences against each reference below the rule. A dagger marks a cell whose seeds do not all agree with its sign. TAAM and DyMoE are the reconstructions of Appendix H.

difference	CoraFull	ogbn-arxiv
aggregation, PDGNNs	$+4.88$ [$+4.60$, $+5.16$]	$+3.88$ [$+3.52$, $+4.22$]
aggregation, DeLoMe	$+3.81$ [$+3.58$, $+4.04$]	-13.44 [-14.66, -12.14]
aggregation, SSM	$+4.71$ [$+4.26$, $+5.20$]	$+13.55$ [$+12.96$, $+14.10$]
aggregation, TPP	$+2.83$ [$+2.56$, $+3.09$]	$+7.18$ [$+6.93$, $+7.42$]
aggregation, TAAM*	$+2.78$ [$+2.29$, $+3.39$]	$+1.06$ [$+0.88$, $+1.22$]
aggregation, DyMoE*	$+10.50$ [$+10.01$, $+10.98$]	$+21.45$ [$+20.81$, $+22.04$]
batch, PDGNNs	0	0
batch, DeLoMe	0	0
batch, SSM	0	0
batch, TPP	-16.89 [-17.35, -16.45]	-49.00 [-49.09, -48.91]
batch, TAAM*	-36.46 [-36.91, -35.88]	-49.45 [-49.62, -49.27]
batch, DyMoE*	0	0
inter-task edges, PDGNNs	-11.84 [-12.07, -11.59]	-6.40 [-6.58, -6.22]
inter-task edges, DeLoMe	-12.78 [-13.10, -12.44]	-4.78 [-5.21, -4.36]
inter-task edges, SSM	-18.06 [-18.47, -17.66]	-13.83 [-14.12, -13.52]
inter-task edges, TPP	-24.76 [-25.09, -24.42]	-15.88 [-15.98, -15.78]
inter-task edges, TAAM*	-3.99 [-4.22, -3.73]	-1.87 [-1.95, -1.79]
inter-task edges, DyMoE*	-0.41 [-0.51, -0.33]	-2.41 [-2.71, -2.13]
batch and edges per node, PDGNNs	-11.84 [-12.07, -11.59]	-6.40 [-6.58, -6.22]
batch and edges per node, DeLoMe	-12.78 [-13.10, -12.44]	-4.78 [-5.21, -4.36]
batch and edges per node, SSM	-18.06 [-18.47, -17.66]	-13.83 [-14.12, -13.52]
batch and edges per node, TPP	-41.65 [-41.98, -41.29]	-64.89 [-64.98, -64.80]
batch and edges per node, TAAM*	-40.45 [-40.94, -39.92]	-51.32 [-51.49, -51.14]
batch and edges per node, DyMoE*	-0.41 [-0.51, -0.33]	-2.41 [-2.71, -2.13]
batch and edges as AA, PDGNNs	-13.14 [-13.53, -12.77]	-8.64 [-8.95, -8.29]
batch and edges as AA, DeLoMe	-12.72 [-13.12, -12.31]	-4.76 [-5.17, -4.37]
batch and edges as AA, SSM	-20.27 [-20.97, -19.67]	-14.01 [-14.45, -13.58]
batch and edges as AA, TPP	-50.27 [-50.80, -49.78]	-75.28 [-75.61, -74.99]
batch and edges as AA, TAAM*	-46.95 [-47.61, -46.30]	-64.11 [-64.41, -63.83]
batch and edges as AA, DyMoE*	-0.44 [-0.61, -0.29]	-1.74 [-1.97, -1.54]
aggregation, cascade	$+7.51$ [$+7.24$, $+7.82$]	$+17.62$ [$+17.23$, $+18.00$]
batch, cascade	0	0
inter-task edges, cascade	-13.30 [-13.77, -12.78]	-15.09 [-15.31, -14.86]
batch and edges per node, cascade	-13.30 [-13.77, -12.78]	-15.09 [-15.31, -14.86]
batch and edges as AA, cascade	-16.31 [-16.84, -15.71]	-19.71 [-20.05, -19.32]
graph choice, PDGNNs	$+1.59$ [$+1.35$, $+1.80$]	$+15.89$ [$+15.66$, $+16.10$]
graph choice, DeLoMe	-1.53 [-1.81, -1.22]	-9.99 [-11.12, -8.72]
graph choice, SSM	$-0.42^{\dagger}$[-0.56, -0.27]	$+11.52$ [$+11.10$, $+11.96$]
graph choice, TPP	-4.59 [-4.97, -4.20]	-1.06 [-1.29, -0.84]
graph choice, TAAM*	$-3.12^{(5)}$ [-3.55, -2.79]	$+10.50^{(5)}$ [$+10.22$, $+10.78$]
graph choice, DyMoE*	$+28.04^{(5)}$ [$+26.32$, $+29.47$]	$+0.45^{\dagger(5)}$ [-1.79, $+2.19$]
edges at training, PDGNNs	-2.76 [-3.09, -2.47]	-1.56 [-1.91, -1.23]
spread under HIDE, mean over seeds	$+19.32$	$+30.99$
PDGNNs $-$ R_{prop}	$+3.79$ [$+3.61$, $+3.97$]	-11.38 [-11.57, -11.20]
DeLoMe $-$ R_{prop}	$+0.72^{\dagger}$[$+0.32$, $+1.12$]	-39.87 [-41.13, -38.45]
SSM $-$ R_{prop}	-7.49 [-7.82, -7.14]	-18.38 [-19.00, -17.82]
TPP $-$ R_{prop}	-15.52 [-15.78, -15.21]	-42.33 [-42.48, -42.20]
TAAM* $-$ R_{prop}	-14.10 [-14.46, -13.77]	-31.18 [-31.40, -30.97]
DyMoE* $-$ R_{prop}	-9.73 [-10.17, -9.28]	-31.54 [-32.77, -30.43]
cascade $-$ R_{prop}	$+1.20$ [$+0.86$, $+1.57$]	-0.91 [-1.02, -0.80]
PDGNNs $-$ R_1	$+1.23$ [$+0.98$, $+1.46$]	-4.39 [-4.57, -4.21]
DeLoMe $-$ R_1	-1.83 [-2.27, -1.40]	-32.88 [-34.14, -31.48]

difference	CoraFull	ogbn-arxiv
SSM $- R_1$	-10.04 $[-10.31, -9.76]$	-11.39 $[-11.98, -10.82]$
TPP $- R_1$	-18.08 $[-18.29, -17.86]$	-35.34 $[-35.48, -35.20]$
TAAM* $- R_1$	-16.66 $[-16.94, -16.37]$	-24.19 $[-24.41, -23.96]$
DyMoE* $- R_1$	-12.29 $[-12.66, -11.88]$	-24.55 $[-25.78, -23.44]$
cascade $- R_1$	-1.36 $[-1.84, -0.90]$	$+6.08$ $[+5.88, +6.28]$
PDGNNs $- R_4$	$+3.16$ $[+2.96, +3.35]$	-10.55 $[-10.74, -10.38]$
DeLoMe $- R_4$	$+0.09^{\dagger}$ $[-0.25, +0.45]$	-39.04 $[-40.27, -37.67]$
SSM $- R_4$	-8.12 $[-8.37, -7.83]$	-17.55 $[-18.17, -16.99]$
TPP $- R_4$	-16.15 $[-16.45, -15.80]$	-41.50 $[-41.61, -41.41]$
TAAM* $- R_4$	-14.73 $[-15.10, -14.38]$	-30.35 $[-30.59, -30.13]$
DyMoE* $- R_4$	-10.36 $[-10.82, -9.94]$	-30.72 $[-31.96, -29.60]$
cascade $- R_4$	$+0.57^{\dagger}$ $[+0.19, +0.97]$	$-0.08^{\dagger}$ $[-0.23, +0.05]$
DeLoMe $- R^{60}_{\mathrm{prop}}$	$+7.46$ $[+7.15, +7.75]$	-21.74 $[-23.07, -20.21]$
SSM $- R^{100}_{\mathrm{prop}}$	-3.28 $[-3.69, -2.85]$	-3.63 $[-4.17, -3.09]$
PDGNNs $- R^{400}_{\mathrm{prop}}$	$+4.90$ $[+4.63, +5.20]$	-2.83 $[-3.02, -2.61]$

V SEED AGREEMENT AND INTERVALS OF THE REFERENCE DIFFERENCES

Table 17 is repeated here cell by cell in Table 33, together with each tier of the cascade against a reference, with the seeds agreeing on the sign and a percentile bootstrap interval of the paired mean at a nominal level of 95%, from the 2.5th to the 97.5th percentile of $10{,}000$ resamples with replacement of the per-seed differences the row is taken over, at a pinned generator seed. Ten paired seeds all agreeing on a nonzero sign reach $p = 2/2^{10} = 0.001953125$ under an exact two-sided sign test, a value that depends on the count of agreeing seeds alone and is distinct from the bootstrap interval, which depends on the magnitudes. The count of agreeing seeds and the interval are reported in place of a test statistic. The interval is itself crude at these sample sizes.

The row printed as R_{prop} *without its labels* is R_{prop} without the calibration labels. Those labels are removed from the three places R_{prop} reads labels: the fit of its linear classifier, the residual correction, and the reset of the labeled rows before smoothing; the training labels remain in all three.

Table 33: Every difference of Table 17 and each tier of the cascade against a reference: the paired mean, the seeds agreeing with its sign, and a percentile bootstrap interval of the paired mean.

dataset	difference	paired mean	seeds agreeing	bootstrap interval
ogbn-arxiv	$\delta = R_{\mathrm{prop}} - R_4$	$+0.83$	10/10	$[+0.75, +0.90]$
ogbn-arxiv	$R_{\mathrm{prop}} - R_1$	$+6.99$	10/10	$[+6.85, +7.12]$
ogbn-arxiv	$R_2 - R_1$, training graph	$+0.53$	10/10	$[+0.39, +0.68]$
ogbn-arxiv	$R_3 - R_2$, calibration labels	$+0.01$	6/10	$[-0.07, +0.08]$
ogbn-arxiv	$R_4 - R_3$, correct and smooth	$+5.62$	10/10	$[+5.50, +5.73]$
ogbn-arxiv	R_{prop} without its labels	-0.57	10/10	$[-0.62, -0.52]$
ogbn-arxiv	R_{prop} without smoothing	-4.16	10/10	$[-4.30, -4.04]$
ogbn-arxiv	cascade streaming $- R_{\mathrm{prop}}$	-11.11	10/10	$[-11.36, -10.89]$
ogbn-arxiv	cascade label memory $- R_{\mathrm{prop}}$	-0.91	10/10	$[-1.02, -0.80]$
ogbn-arxiv	cascade transductive $- R_4$	$+0.30$	9/10	$[+0.17, +0.43]$
CiteSeer	$\delta = R_{\mathrm{prop}} - R_4$	-0.83	7/10	$[-1.49, -0.13]$
CiteSeer	$R_{\mathrm{prop}} - R_1$	-3.15	10/10	$[-3.59, -2.68]$
CiteSeer	$R_2 - R_1$, training graph	-4.23	10/10	$[-4.70, -3.74]$
CiteSeer	$R_3 - R_2$, calibration labels	$+1.18$	8/10	$[+0.65, +1.68]$
CiteSeer	$R_4 - R_3$, correct and smooth	$+0.74$	8/10	$[+0.48, +0.98]$
CiteSeer	R_{prop} without its labels	-1.58	10/10	$[-2.01, -1.13]$
CiteSeer	R_{prop} without smoothing	-0.75	9/10	$[-1.22, -0.30]$
CiteSeer	cascade streaming $- R_{\mathrm{prop}}$	-2.27	10/10	$[-2.92, -1.60]$
CiteSeer	cascade label memory $- R_{\mathrm{prop}}$	$+0.94$	7/10	$[+0.29, +1.59]$
CiteSeer	cascade transductive $- R_4$	$+0.05$	4/10	$[-0.65, +0.81]$
Comp.	$\delta = R_{\mathrm{prop}} - R_4$	$+0.64$	9/10	$[+0.36, +0.90]$
Comp.	$R_{\mathrm{prop}} - R_1$	$+3.18$	10/10	$[+2.60, +3.83]$

dataset	difference	paired mean	seeds agreeing	bootstrap interval
Comp.	$R_2 - R_1$, training graph	$+2.92$	10/10	$[+2.40, +3.47]$
Comp.	$R_3 - R_2$, calibration labels	$+0.18$	6/10	$[-0.22, +0.57]$
Comp.	$R_4 - R_3$, correct and smooth	-0.56	8/10	$[-0.85, -0.23]$
Comp.	R_{prop} without its labels	-0.31	10/10	$[-0.43, -0.20]$
Comp.	R_{prop} without smoothing	$+0.41$	8/10	$[+0.18, +0.65]$
Comp.	cascade streaming $- R_{\text{prop}}$	-0.28	6/10	$[-0.63, +0.05]$
Comp.	cascade label memory $- R_{\text{prop}}$	$+0.32$	10/10	$[+0.23, +0.41]$
Comp.	cascade transductive $- R_4$	$+1.72$	10/10	$[+1.38, +2.08]$
Cora	$\delta = R_{\text{prop}} - R_4$	-1.00	9/10	$[-1.56, -0.49]$
Cora	$R_{\text{prop}} - R_1$	-2.34	10/10	$[-3.30, -1.56]$
Cora	$R_2 - R_1$, training graph	-2.73	10/10	$[-3.32, -2.13]$
Cora	$R_3 - R_2$, calibration labels	$+0.49$	7/10	$[+0.00, +1.00]$
Cora	$R_4 - R_3$, correct and smooth	$+0.90$	9/10	$[+0.59, +1.23]$
Cora	R_{prop} without its labels	-0.41	6/10	$[-0.99, +0.16]$
Cora	R_{prop} without smoothing	-1.27	10/10	$[-1.43, -1.11]$
Cora	cascade streaming $- R_{\text{prop}}$	-1.05	7/10	$[-1.93, -0.21]$
Cora	cascade label memory $- R_{\text{prop}}$	$+0.08$	6/10	$[-0.72, +0.86]$
Cora	cascade transductive $- R_4$	-0.31	6/10	$[-0.86, +0.28]$
CoraFull	$\delta = R_{\text{prop}} - R_4$	-0.63	10/10	$[-0.79, -0.47]$
CoraFull	$R_{\text{prop}} - R_1$	-2.56	10/10	$[-2.81, -2.34]$
CoraFull	$R_2 - R_1$, training graph	-2.22	10/10	$[-2.34, -2.09]$
CoraFull	$R_3 - R_2$, calibration labels	$+0.97$	10/10	$[+0.72, +1.20]$
CoraFull	$R_4 - R_3$, correct and smooth	-0.68	10/10	$[-0.86, -0.51]$
CoraFull	R_{prop} without its labels	-1.04	10/10	$[-1.34, -0.78]$
CoraFull	R_{prop} without smoothing	-0.49	10/10	$[-0.59, -0.40]$
CoraFull	cascade streaming $- R_{\text{prop}}$	-2.36	10/10	$[-2.81, -1.86]$
CoraFull	cascade label memory $- R_{\text{prop}}$	$+1.20$	10/10	$[+0.86, +1.56]$
CoraFull	cascade transductive $- R_4$	$+2.17$	10/10	$[+1.87, +2.51]$
Photo	$\delta = R_{\text{prop}} - R_4$	$+1.19$	10/10	$[+0.93, +1.47]$
Photo	$R_{\text{prop}} - R_1$	$+1.62$	10/10	$[+1.38, +1.89]$
Photo	$R_2 - R_1$, training graph	$+0.94$	9/10	$[+0.47, +1.37]$
Photo	$R_3 - R_2$, calibration labels	-0.04	6/10	$[-0.35, +0.25]$
Photo	$R_4 - R_3$, correct and smooth	-0.47	10/10	$[-0.61, -0.34]$
Photo	R_{prop} without its labels	-0.21	9/10	$[-0.34, -0.07]$
Photo	R_{prop} without smoothing	$+1.18$	10/10	$[+1.02, +1.30]$
Photo	cascade streaming $- R_{\text{prop}}$	-0.53	10/10	$[-0.70, -0.35]$
Photo	cascade label memory $- R_{\text{prop}}$	-0.10	8/10	$[-0.26, +0.06]$
Photo	cascade transductive $- R_4$	$+1.24$	10/10	$[+0.93, +1.59]$
prod.	$\delta = R_{\text{prop}} - R_4$	-0.98	10/10	$[-1.00, -0.97]$
prod.	$R_{\text{prop}} - R_1$	$+3.03$	10/10	$[+3.01, +3.05]$
prod.	$R_2 - R_1$, training graph	$+0.27$	10/10	$[+0.26, +0.28]$
prod.	$R_3 - R_2$, calibration labels	-0.00	7/10	$[-0.01, +0.00]$
prod.	$R_4 - R_3$, correct and smooth	$+3.75$	10/10	$[+3.73, +3.77]$
prod.	R_{prop} without its labels	-0.75	10/10	$[-0.76, -0.74]$
prod.	R_{prop} without smoothing	-10.03	10/10	$[-10.05, -10.01]$
Reddit	$\delta = R_{\text{prop}} - R_4$	$+0.46$	10/10	$[+0.42, +0.50]$
Reddit	$R_{\text{prop}} - R_1$	$+2.67$	10/10	$[+2.56, +2.79]$
Reddit	$R_2 - R_1$, training graph	$+2.08$	10/10	$[+1.97, +2.19]$
Reddit	$R_3 - R_2$, calibration labels	$+0.01$	6/10	$[-0.00, +0.03]$
Reddit	$R_4 - R_3$, correct and smooth	$+0.12$	10/10	$[+0.09, +0.14]$
Reddit	R_{prop} without its labels	-0.24	10/10	$[-0.27, -0.22]$
Reddit	R_{prop} without smoothing	-2.07	10/10	$[-2.10, -2.04]$
Reddit	cascade streaming $- R_{\text{prop}}$	-18.18	10/10	$[-18.65, -17.75]$
Reddit	cascade label memory $- R_{\text{prop}}$	-2.91	10/10	$[-3.27, -2.66]$
Reddit	cascade transductive $- R_4$	-2.12	10/10	$[-2.46, -1.89]$
Roman	$\delta = R_{\text{prop}} - R_4$	$+29.39$	10/10	$[+29.00, +29.75]$
Roman	$R_{\text{prop}} - R_1$	$+16.28$	10/10	$[+15.57, +16.95]$
Roman	$R_2 - R_1$, training graph	$+21.32$	10/10	$[+20.47, +21.98]$
Roman	$R_3 - R_2$, calibration labels	$+0.36$	9/10	$[+0.21, +0.51]$
Roman	$R_4 - R_3$, correct and smooth	-34.78	10/10	$[-35.24, -34.31]$
Roman	R_{prop} without its labels	$+0.87$	10/10	$[+0.66, +1.08]$
Roman	R_{prop} without smoothing	$+18.81$	10/10	$[+18.35, +19.26]$
Roman	cascade streaming $- R_{\text{prop}}$	-20.21	10/10	$[-21.02, -19.27]$

dataset	difference	paired mean	seeds agreeing	bootstrap interval
Roman	cascade label memory $- R_{\mathrm{prop}}$	-16.04	10/10	$[-16.66, -15.42]$
Roman	cascade transductive $- R_4$	$+10.81$	10/10	$[+10.14, +11.62]$
WikiCS	$\delta = R_{\mathrm{prop}} - R_4$	$+1.51$	10/10	$[\ +1.24,\ +1.76]$
WikiCS	$R_{\mathrm{prop}} - R_1$	$+3.39$	10/10	$[\ +3.04,\ +3.75]$
WikiCS	$R_2 - R_1$, training graph	$+1.05$	10/10	$[\ +0.70,\ +1.41]$
WikiCS	$R_3 - R_2$, calibration labels	$+0.19$	5/10	$[\ -0.15,\ +0.51]$
WikiCS	$R_4 - R_3$, correct and smooth	$+0.63$	9/10	$[\ +0.42,\ +0.82]$
WikiCS	R_{prop} without its labels	-0.61	10/10	$[\ -0.76,\ -0.45]$
WikiCS	R_{prop} without smoothing	-0.88	10/10	$[\ -0.99,\ -0.75]$
WikiCS	cascade streaming $- R_{\mathrm{prop}}$	-4.59	10/10	$[\ -5.04,\ -4.13]$
WikiCS	cascade label memory $- R_{\mathrm{prop}}$	-1.58	10/10	$[\ -1.79,\ -1.36]$
WikiCS	cascade transductive $- R_4$	$+0.55$	9/10	$[\ +0.29,\ +0.83]$