CLOSED-FORM NODE CLASSIFICATION WITH EXACT GRAPH UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks for node classification are typically trained by gradient descent over hundreds or thousands of epochs. Recent work shows that carefully tuned classic GCN, GraphSAGE and GAT architectures can rival graph transformers. We ask a complementary question: can closed-form solvers recover this predictive strength and make exact graph unlearning practical?

One validation-selected protocol combines multiscale diffusion, nonlinear graph kernels and training-label context with analytical readouts. Across generic and task-aware tracks, **closed-form predictors exceed or come within one point** of the strongest reproduced tuned GNN on all 16 reported benchmarks. The generic protocol achieves this on 14/16, including both OGB graphs; task-aware constructions surpass the remaining two references, setting a new Minesweeper state of the art at 99.95 ROC–AUC with an 11 ms clue-based CPU fit, and reaching 91.60 accuracy on Roman-empire. The generic protocol also exceeds cited retuned GraphGPS and SGFormer means on 14/15 and 12/15 comparisons.

We derive exact graph-unlearning procedures that recover the result of retraining the same model from scratch after deleting labels, features, edges, nodes or subgraphs. They reuse unaffected information and recompute only what changes. Across 280 deterministic verification cases, 232 match retrained prediction scores exactly; 43 double-precision label updates differ by less than 2.6×10^{-11}, and five Questions updates preserve every predicted class against the single-precision reference. Timed in default GPU mode, cached one-label model updates are a **median 93× faster** than tuned-GNN training steps alone across fifteen same-hardware comparisons, with speedups up to approximately $20{,}000\times$. Closed-form fitting therefore supports both strong node prediction and exact, verifiable unlearning.

1 INTRODUCTION

Graph neural networks dominate node classification, with prediction learned through hundreds or thousands of gradient steps. Luo et al. (2024) show that carefully tuned classic GCN, GraphSAGE and GAT can rival much more elaborate graph transformers. We ask the complementary question: how far can explicit closed-form solvers recover this predictive strength, and what do they enable when learned information must be removed?

Our approach is to construct informative graph representations, then fit their labels through regularized linear systems. Multiscale diffusion supplies neighborhood features, nonlinear graph kernels supply richer interactions, and optional training-label context supplies transductive information. One validation-selected procedure chooses among these candidates, using landmark representations at large scale (Figure 1).

Empirical headline. A single validation-selected generic protocol exceeds the best reproduced tuned GNN on seven datasets and comes within one point on seven more. Task-aware closed forms reach 99.95 ROC–AUC on Minesweeper and 91.60 accuracy on Roman-empire, surpassing the remaining two references. The generic method exceeds the cited retuned GraphGPS and SGFormer means on 14/15 and 12/15 comparisons. ACR sets a new Minesweeper state of the art, above CSNN's 99.07 and GraphTARIF's 99.03 (Ribeiro et al., 2026; Hu et al., 2026), by exploiting the benchmark's declared clue structure.

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.

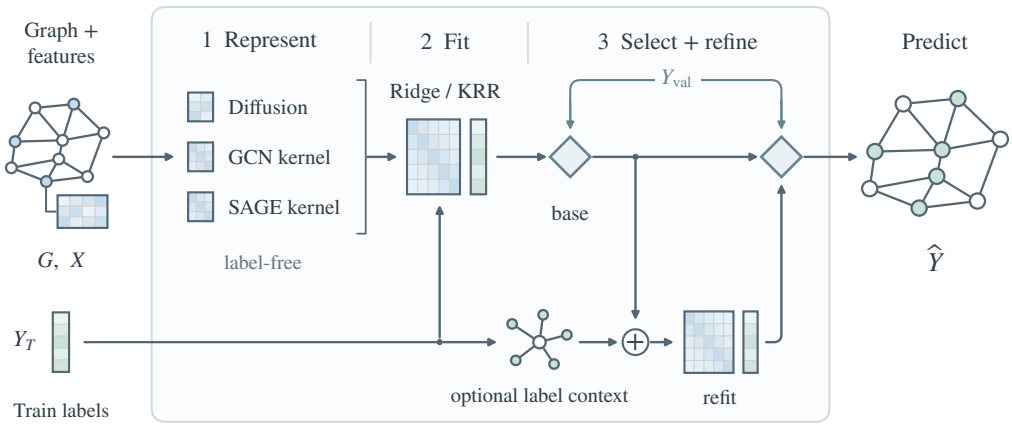

Figure 1: **Generic prediction in two information stages.** Diffusion or kernel representations feed an analytical solve selected by validation. The optional lower path adds label context and refits (§3.3); validation chooses the base or refinement.

Exact graph unlearning. The explicit systems also identify which computations a deletion changes. A *label-free representation*, built without training labels, can be reused after label deletion; its readout is downdated, while label-dependent stages are rebuilt. The target is the *retained-data fit*: the predictor obtained by fitting after the edit, including reselection when the learning procedure requires it. We formalize this target for label, feature, edge, node and subgraph edits, and verify 280 cases: 158 label deletions and 122 structural edits.

Contributions. We establish three linked results:

1. **Competitive prediction without gradient training.** One validation-selected protocol exceeds the tuned reference on seven datasets and comes within one point on seven more.
2. **Task-aware closed-form gains.** Analytic Constraint Reconstruction (ACR) and Typed LCF-Net (TLCF) exploit declared clue and sequence structure, reaching 99.95 on Minesweeper and 91.60 on Roman-empire.
3. **Verified unlearning across five edit types.** We formalize the applicable updates and recomputations, verify 280 deletion cases, and enable reproducible GPU execution with a fixed-order sparse kernel. Cached label updates are a median 93× faster than same-hardware GNN training-step references.

2 Related Work

Graph models and kernels. GCN, GraphSAGE, and GAT are the underlying trained architectures (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018). SGC, S^2GC, SIGN, and Correct-and-Smooth decouple propagation and learning (Wu et al., 2019; Zhu & Koniusz, 2021; Frasca et al., 2020; Huang et al., 2021). Our GCN-form, GraphSAGE-form and landmark representations build on Niu et al. (2023), adding label context and retained-data computation. GraphAny analytically fits LinearGNNs and learns an attention-based combination for fully inductive transfer (Zhao et al., 2025); our protocol selects nonlinear closed-form predictors for each graph and tracks their deletion dependencies. Tuned GNNs (Luo et al., 2024) and GraphGPS, SGFormer, and Polynormer (Rampášek et al., 2022; Wu et al., 2023c; Deng et al., 2024) provide strong predictive references.

Unlearning. SISA limits deletion cost through partitioned training (Bourtoule et al., 2021); GraphEraser adapts partitioning and aggregation to graphs (Chen et al., 2022). GNNDelete learns deletion operators (Cheng et al., 2023), and GIF uses graph-aware influence functions (Wu et al.,

2023a). Certified removal and its graph extensions give (ϵ, δ)-approximate guarantees under their convex, linear or decoupled model assumptions (Guo et al., 2020; Chien et al., 2023; Wu et al., 2023b; Yi & Wei, 2025). GraphEditor gives exact Woodbury updates of a linear-GNN ridge solution for node/edge deletion, insertion and feature updates (Cong & Mahdavi, 2023b); Projector removes node information by projection (Cong & Mahdavi, 2023a). We extend the empirical focus to competitive nonlinear kernels, label context and task-aware predictors, with replayed selection and fixed-order GPU verification. Retrain-equivalence is distinct from structural-inference resistance, as studied by TrendAttack (Zhang et al., 2026); retained attack measurements appear in Appendix F.

3 Closed-form graph prediction

The backbone builds a representation, solves its readout and selects by validation. Optional label context enriches that representation before refitting.

3.1 A shared analytical readout

For graph $G = (V, E)$, node features $X \in \mathbb{R}^{n \times d}$, training nodes T, and targets Y_T, let H be a feature map and K a positive-semidefinite kernel. Ridge regression and kernel ridge regression (KRR) fit the primal and dual readouts:

$$W = (H_T^\top H_T + \lambda I)^{-1} H_T^\top Y_T, \qquad \widehat{Y} = HW, \tag{1}$$

$$\alpha = (K_{TT} + \lambda I)^{-1} Y_T, \qquad \widehat{Y} = K_{VT}\alpha. \tag{2}$$

Targets are one-hot or multilabel binary vectors. Inverse notation defines the solution; numerical linear-system solvers implement it.

3.2 Representations: from diffusion to nonlinear neighborhoods

Diffusion and multiscale features. The linear family uses the S^{2}GC multiscale filter (Zhu & Koniusz, 2021):

$$H = \mathrm{RowNorm}\left[(1-\beta)q^{-1}\sum_{k=1}^{q}\widetilde{A}^k\widehat{X} + \beta\widehat{X}\right], \qquad \beta = 0.1, \quad q \in \{2, 16, 48\}.$$

Here $\widetilde{A}$ symmetrically normalizes $A+I$ and $\widehat{X}$ is row-normalized. A SIGN-style family (Frasca et al., 2020) concatenates standardized features with one- and two-step symmetric, mean, and available edge-channel propagations, followed by ridge.

Nonlinear graph kernels. Covariance recursion captures nonlinear neighborhood interactions without gradient-trained message-passing weights. Let Φ be the normalized ReLU (arc-cosine) covariance transform (Cho & Saul, 2009) and $\mathcal{N}$ normalize a covariance to unit diagonal. Following Niu et al. (2023), starting from the normalized feature Gram matrix, compute

$$K_{\ell+1} = \mathcal{N}\left[rK_\ell + w_e\Phi(K_\ell) + w_n A_\star \Phi(K_\ell) A_\star^\top\right]. \tag{3}$$

Here r weights the residual covariance, while w_e and w_n weight ego and neighbor terms. GCN-form candidates use symmetric self-loop propagation with $w_e = 0$. GraphSAGE-form candidates retain separate ego and mean-neighbor terms. Input preprocessing, residual structure, depth, and regularization are selected by validation; the final covariance head applies Φ before kernel ridge fitting. Above 50,000 nodes, a layer-wise Nyström landmark recursion (Williams & Seeger, 2000; Niu et al., 2023) with 8,192 seeded landmarks provides explicit features for primal ridge. The representation is approximate; its readout solves the specified represented model. Appendix A gives the covariance transform and all candidate settings.

3.3 Label context with explicit dependencies

Label propagation and labels-as-features provide complementary transductive information (Zhu et al., 2003; Zhou et al., 2003; Wang & Leskovec, 2021; Sato, 2024). Our optional second stage keeps these

dependencies explicit. Context comprises retained-label neighbor distributions, return-path-excluded two-hop aggregates and cross-fitted label propagation, normalized by available label mass. Direct self-contributions are removed and each training query's fold is excluded from propagation seeds. Validation/test queries use training labels only. Label-dependent global normalization and bandwidths are also recomputed on deletion.

Explicit-feature models concatenate context and multiscale features with the selected base; exact-kernel models add their weighted linear kernel. Stage one selects a label-free candidate, and stage two compares eligible refinements against that candidate, reselecting weight and regularization. Within each frozen procedure, both stages select candidates using validation scores only. Appendix A specifies the grids and the precise eligibility of the earlier Gaussian context variant.

3.4 A SEPARATE TRACK FOR DECLARED TASK STRUCTURE

Two task-aware closed forms exploit declared clue or sequence structure beyond generic neighborhood similarity. They are reported separately from the common protocol and specified in Appendix B.

Minesweeper: constraint reconstruction. Analytic Constraint Reconstruction (ACR) turns visible clue features into equations for neighboring mine counts. With unknown-node scores z, clue-to-unknown matrix B, residual counts b after subtracting known training labels, and training mine rate π, ACR solves

$$z^\star = (B^\top B + \lambda I)^{-1}(B^\top b + \lambda\pi\mathbf{1}). \tag{4}$$

Regularization and censored-clue handling are validation-selected. This explicitly task-aware model uses supplied clue semantics, graph structure, and training labels.

Roman-empire: typed closed-form refinement. Typed LCF-Net (TLCF) is a typed *Layer-wise Closed-Form Deep Network*. It separates token-chain offsets from non-adjacent graph relations, using node order as token order. Each layer solves ridge or kernel regression and passes prediction context to the next; label counts are leave-one-out and training-row predictions cross-fitted. Three fold-map runs are averaged; validation selects a layer or forward–backward chain readout. No external parser or text corpus is used (Appendix B.2).

4 FROM PREDICTION TO EXACT UNLEARNING

The computational separation above identifies what must change after a deletion. Our rule is to *reuse only information independent of the edit and rebuild its affected descendants*. Figure 2 follows a removed training label through the resulting paths.

4.1 SPECIFY THE RETAINED-DATA TARGET

Definition 1 (Exact graph unlearning (Cao & Yang, 2015; Ginart et al., 2019))**.** For a deterministic learning procedure $\mathcal{A}$ and edit F, an unlearner $\mathcal{U}$ is exact when its returned predictor satisfies

$$\mathcal{U}(\mathcal{A}(D), D \setminus F, F) = \mathcal{A}(D \setminus F),$$

where the unlearner can use the fitted model's caches and $D \setminus F$ denotes the edited data. Edits may remove labels, features, edges, nodes or subgraphs; the target always uses the same specified learning procedure.

Generic-model checks fix preprocessing, seeds, regularization and configuration; ACR/TLCF replay validation selection. Algebraic equality is defined in real arithmetic; numerical execution is checked separately. A training-label deletion removes a supervised record, retaining the graph node and features.

Proposition 1 (Dependency-complete recomputation)**.** *For a finite deterministic computation of transformations and uniquely defined regularized solves, reuse only objects independent of the edited information and recompute all affected descendants. This returns the fixed-configuration retained-data predictor. Replaying selection and its deterministic tie rule yields the selection-aware target.*

The proof is induction over the computation graph (Appendix C).

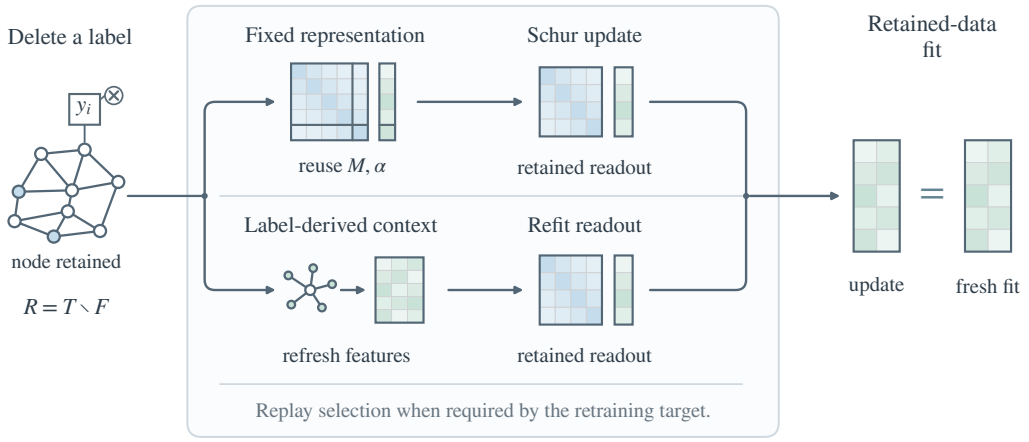

Figure 2: **Delete a label, then follow its dependencies.** The node remains. Unchanged representations permit a cached Schur correction; label-dependent context is rebuilt. Both paths target the retained-data fit in real arithmetic, with selection replayed when required. Numerical checks appear in §6.5.

4.2 Downdate the readout when the representation is unchanged

If H is unchanged, deleting $F \subset T$ changes *both* $H_T^\top H_T + \lambda I$ and $H_T^\top Y_T$ by subtracting $H_F^\top H_F$ and $H_F^\top Y_F$. For a fixed kernel, set $R = T \setminus F$, $M = (K_{TT} + \lambda I)^{-1}$, and $\alpha = MY_T$. The classical block-inverse identity (Hager, 1989) gives

$$\alpha'_R = \alpha_R - M_{RF} M_{FF}^{-1} \alpha_F. \tag{5}$$

This follows the decremental-learning principle (Cauwenberghs & Poggio, 2000), with linear primal graph updates also used by GraphEditor (Cong & Mahdavi, 2023b). The inverse cache admits a block update for subsequent requests (Appendix C), without rebuilding the label-free kernel.

Rebuild the stages that use labels. For a context-enriched predictor, the kernel cache alone is not the entire model. Deletion also changes context features, their normalization, applicable bandwidths and downstream solves. These stages are recomputed, while independent graph work is reused. When selection is included in the target, validation scores and the deterministic selection rule are replayed before fitting the new winner (Figure 2).

When is the update local? Locality requires local representation dependencies. Fixed propagation can leave most rows unchanged; global preprocessing and label-fitted refinement can spread an edit. Appendix C retains the conditional ridge-statistic locality result and separates representation refresh, factor updates, and prediction refresh. Diffusion controls confirm this dependence on radius: up to $3.9\times$ at $q = 2$ on arxiv, but no speed gain at the tested large radii (Appendix C.3).

5 Evaluation protocol

Prediction. We report sixteen citation, co-purchase, coauthor, Wikipedia, heterophily (Platonov et al., 2023), and OGB (Hu et al., 2020) datasets. Minesweeper, Tolokers, Questions, and multilabel proteins use ROC–AUC; others use accuracy. Primary splits are seed-123 class-random Planetoid masks, fixed Amazon/Coauthor masks, splits 0–2 for WikiCS and the five Platonov benchmarks, ten filtered Squirrel splits, and official OGB splits. Sample standard deviations are reported where primary splits permit them. Dataset sizes and selected configurations appear in Appendix K.

References and development. The primary GNN reference is the largest mean validation-selected *Final Test* score among available local tuned GCN/SAGE/GAT reproductions. GNNs are rerun

Table 1: Predictive means (%). **Bold**: a closed-form score above or within one point of the best local Final-Test reference (underlined); this descriptive margin is not a significance test. Generic and task-aware results are distinct tracks. Published and historical columns retain their source protocols.

		Closed form		Historical	Local tuned: Final			Published references			
Dataset	Metric	Generic	Task	2L†	GCN	SAGE	GAT	GPS	SGF	Poly	OGB
Cora	Acc.	**84.50**	—	81.80	83.50	83.48	83.56	83.87	84.82	83.43	—
CiteSeer	Acc.	**73.00**	—	71.90	72.44	71.58	71.38	72.73	72.72	72.19	—
PubMed	Acc.	**81.40**	—	78.70	80.54	78.40	79.88	79.94	80.60	79.35	—
Photo	Acc.	**96.54**	—	91.40	96.05	96.30	96.40	94.89	95.58	96.57	—
Computer	Acc.	**94.00**	—	—	94.13	93.26	93.94	91.79	92.42	93.78	—
CS	Acc.	**96.07**	—	91.30	95.88	96.05	96.13	94.04	95.71	95.42	—
Physics	Acc.	**96.68**	—	93.00	97.50	97.03	97.24	96.71	96.75	97.18	—
WikiCS	Acc.	**80.89**	—	79.63	80.40	80.45	80.93	78.66	80.05	80.26	—
Ratings	Acc.	**56.31**	—	54.49	53.66	54.99	55.30	53.27	54.14	54.96	—
Roman	Acc.	84.20	**91.60**	83.73	91.14	90.70	90.05	82.72	80.01	92.66	—
Minesweeper	AUC	92.92	**99.95**	90.72	97.77	98.08	97.83	90.75	91.42	97.49	—
Tolokers	AUC	**85.52**	—	—	86.03	84.48	85.45	—	—	—	—
Questions	AUC	**78.36**	—	—	78.19	76.11	77.31	72.56	73.81	78.94	—
Squirrel	Acc.	**43.65**	—	—	44.47	40.95	40.12	39.81	42.65	41.97	—
arxiv	Acc.	**72.22**	—	—	72.93	72.79	—	71.23	72.76	73.40	71.74
proteins	AUC	**85.73**	—	—	75.65	79.35‡	83.78‡	77.15	79.92	79.53	72.51

Task: ACR on Minesweeper, TLCF on Roman-empire. GPS/SGF/Poly: retuned GraphGPS/SGFormer/Polynormer; OGB: plain GCN, all from Luo et al. (2024). †Cross-protocol historical two-layer references (Appendix H). ‡Two completed seeds; proteins GAT is L5H4. Stronger published edge-aware proteins GAT: 87.82 (§6.4). Full source distinctions, split variation and reproduction means appear in the appendices.

with the released code of Luo et al. (2024) on our hardware and splits; the more expensive graph-transformer references are cited from their retuned (starred) tables. Proteins uses a two-completed-seed local GAT-L5H4 reference; its stronger published edge-aware reference is separate. The procedure and reporting suite underwent iterative benchmark development, including extensions informed by earlier test outcomes; within each frozen procedure, candidate selection is validation-only (Appendix A).

Deletion and numerical checks. Unlearning removes training-label sets drawn from $\{1, 10, 100, \lfloor .01|T| \rfloor, \lfloor .05|T| \rfloor\}$, with invalid and duplicate sizes removed. Fresh fitting uses retained masks with deleted labels overwritten by incorrect values. Comparing those outputs with the original-label retained-mask computation checks whether deleted labels are still accessed. The structural suite adds seven edge, node, feature-row and connected-subgraph edits per tested model; Appendix O specifies sizes, reindexing and cache reuse. Deterministic verification uses CPU-CSR or fixed-order GPU sparse products; default-GPU timing uses synchronized windows on RTX PRO 5000 Blackwell hardware. Its numerical behavior is reported separately.

6 Results

Prediction and deletion checks use the selected predictors themselves.

6.1 Fourteen benchmarks within one point or better

The generic closed-form protocol exceeds the strongest available local tuned-GNN reference on seven datasets and is at most one point below it on seven more (Table 1). Validation chooses SAGE-form kernels on Amazon-Ratings, nonlinear kernels on WikiCS and diffusion on PubMed: complementary representations under one fitting principle.

Against the consistently retuned transformer means in Luo et al. (2024), its scores are higher on 14/15 GraphGPS, 12/15 SGFormer, and 9/15 Polynormer comparisons (Appendix G). It also numerically

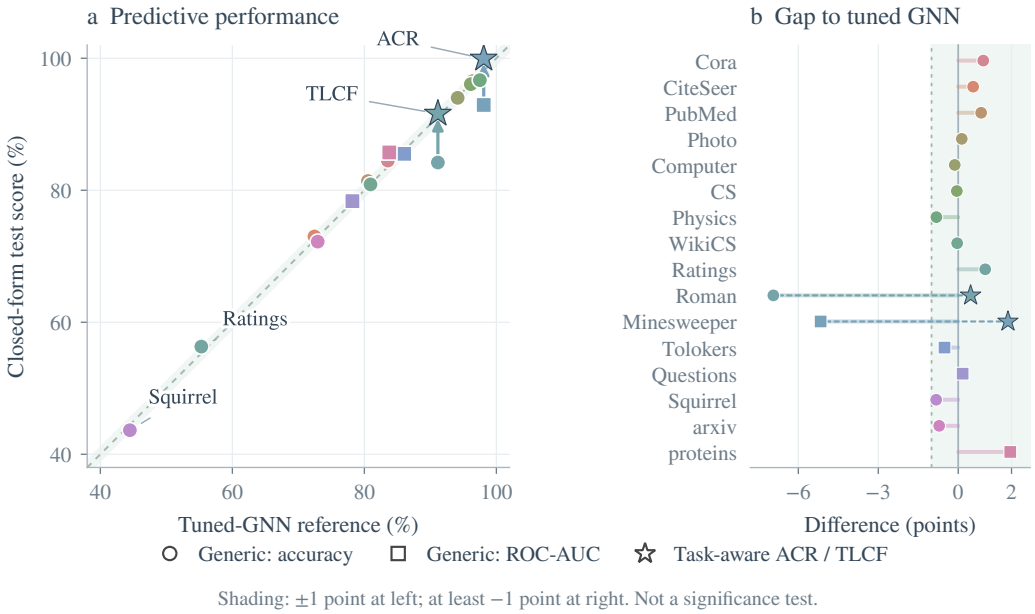

Figure 3: **Where the predictive gaps close.** Parity and dataset-wise differences use the best local Final-Test references; stars denote task-aware ACR/TLCF. Colors identify datasets and shapes identify metrics. Shading marks ± 1 point at left and at least -1 point at right, not uncertainty.

exceeds all ten historical two-layer reference entries. That latter comparison is explicitly cross-protocol, with label-budget and source distinctions in Appendix H.

6.2 Declared task structure closes the two largest gaps

Minesweeper and Roman-empire expose clue and sequence structure that generic neighborhood kernels do not explicitly encode. ACR and TLCF exploit this declared information while retaining analytical fitting.

Clues become constraints. ACR sets a new Minesweeper state of the art, to our knowledge: 99.95 ± 0.02 ROC–AUC, above CSNN's 99.07 ± 0.25 (Ribeiro et al., 2026), GraphTARIF's 99.03 ± 0.19 (Hu et al., 2026), and every Minesweeper entry in Luo et al. (2024). The selected CPU fit takes approximately 11 ms. The reproduced 15-layer, 2,000-epoch SAGE run takes approximately 27 s on a Blackwell GPU. All ten official splits yield 99.96 ± 0.02.

Order becomes typed context. TLCF reaches 91.60 ± 0.06 accuracy on Roman-empire, against local GCN 91.14. All ten splits yield 91.36 ± 0.38; development-unused splits 3–9 average 91.25. Its result combines typed refinement, label-count context, and selected chain decoding. Together the two evaluation tracks supply an above-or-within-one-point result on every reported dataset, while keeping task-aware input assumptions distinct.

6.3 What contributes predictive information?

Each representation restriction selects its own candidate by validation.

Representation choice. The full label-free candidate pool exceeds its shallow restriction by 1.67 points on Amazon-Ratings and 1.28 on Minesweeper. Larger gains over a restricted nonresidual GCN kernel jointly change architecture, inputs, and selection; they do not isolate depth. The controls and their selection rules are visualized in Appendix J. Appendix E retains historical LCF-Net controls.

Additional label context. On proteins, context plus multiscale features reaches 85.73, versus 77.87 for the selected label-free candidate. This is the gain of the combined add-on and regularization choice. On arxiv, validation retains the 72.22 context model even though the separate label-free candidate scores 72.34 on test. We retain the validation decision.

6.4 Landmarks extend the same principle to OGB scale

Landmark recursion gives 72.22 accuracy on arxiv and 85.73 ROC–AUC on proteins, against 68.87 and 62.88 for the initial linear-only large-graph protocol. Both exceed the plain-GCN references of 71.74 and 72.51 (Luo et al., 2024, Table 4). The stronger published references are 73.53 and 87.82, respectively; the latter is the edge-aware proteins value in Table 11 of the cited arXiv version (Luo et al., 2024).

Dense recursion stores $O(n^2)$ entries; m landmarks reduce storage to $O(nm + m^2)$, with rank-dependent construction and solve costs detailed in Appendix A. Appendix E separately retains the earlier controlled Gaussian-kernel comparison.

6.5 Deletion reproduces the retained-data predictor

All 232 recomputation checks reproduce retained-data scores exactly; all 48 label downdates preserve class predictions (Table 2). Label deletion covers all sixteen generic datasets and both task-aware protocols: 141 generic cases plus 17 selection-aware cases. Another 122 checks cover structural edits: 98 on fourteen split-0 generic winners, 21 ACR cases and three TLCF cases with reselection. Proteins is checked by multilabel score equality.

Table 2: Completed deterministic deletion checks. Discrepancy is the maximum absolute score difference from retained-data fitting, not a full-state erasure test.

Computation / edit family	Cases	Max. $\lvert\Delta\rvert$ (upper bound)
Generic label recomputation	93	0
Label downdate, fp64 kernels	43	2.51×10^{-11}
Questions label downdate, fp32 reference	5	2.44×10^{-4}
ACR labels, including reselection	15	0
TLCF labels, including reselection	2	0
Generic structural recomputation	98	0
ACR structural, including reselection	21	0
TLCF structural, including reselection	3	0

The Questions reference evaluates outputs in fp32, while the downdate evaluates in fp64; a cached fp64 full re-solve exhibits the same maximum-discrepancy scale. Across up to 100 sequential deletions, drift stays below 2.76×10^{-11} for fp64 cases and 2.33×10^{-4} for Questions. ACR/TLCF also replay selection, including requests that change the selected configuration.

Structural edits change the graph, its node population or its features, yet the retained-data target is recovered with zero score discrepancy in all 122 cases. ACR and TLCF change their selected configuration in thirteen and two requests. Fixed-order GPU sparse products make this verification practical on the GPU: deterministic Minesweeper structural recomputation takes 1.45–1.92 s. Across three sparse-product benchmarks, this kernel is 6.0–$18.4\times$ faster than eight-thread CPU-CSR (Appendix O).

Default-GPU timing and deterministic verification have different numerical envelopes; Appendix D reports both, together with poison-label and repeat checks.

6.6 Reusing graph computation makes updates inexpensive

Cached one-record updates take 0.9 ms–23.1 s in default-GPU mode. Across fifteen same-hardware comparisons they are a median $93\times$ faster than tuned-GNN training steps (Figure 4). Label dependence organizes the ratios: 560–$19{,}984\times$ for the five consistently label-free winners and 4.2–$114.7\times$ for the other dataset medians, rebuilding context where selected. Complete cache setup takes 0.16–10.57 s on the five label-free winners; setup plus one update still costs less than one GNN training-step

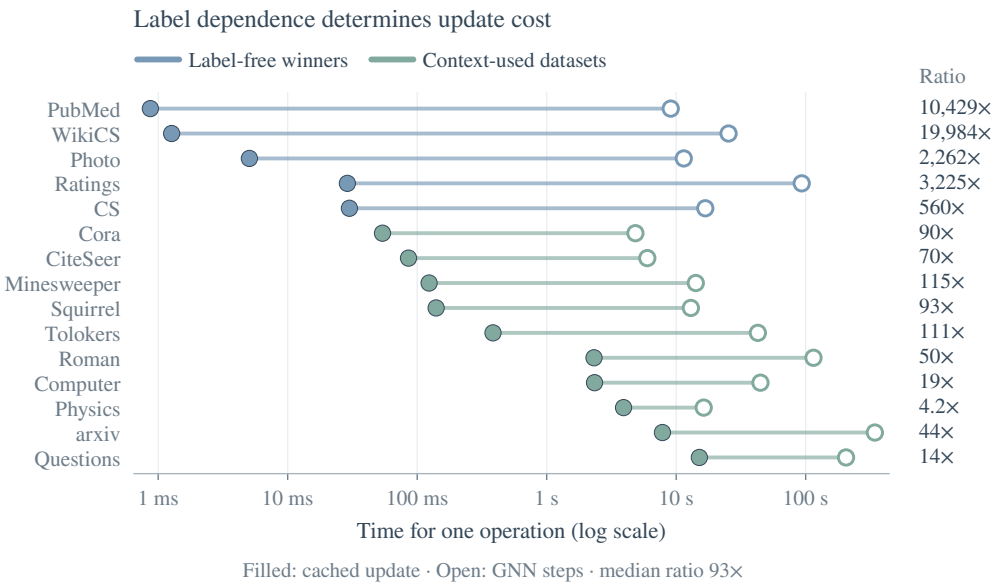

Figure 4: Cached label updates (filled) versus GNN training steps (open), paired by dataset on a logarithmic time axis. Ratios appear at right; updates include prediction refresh. Blue denotes consistently label-free winners; teal denotes context-used datasets (Questions mixes paths across splits). Default-GPU mode; proteins lacks a matched-hardware GNN reference and is excluded.

reference. GNN references are original-data fixed-epoch recipes; deletion preserves graph size and epoch budget. Caches, mask preparation and reselection are separate (Appendix I). Against fresh fitting of the same closed-form model, one-record downdates and cached recomputation are a median $272\times$ and $6\times$ faster (ten and twenty split-model pairs).

For thirteen structural-suite datasets, edit-family median recomputations take 0.47–122.94 s, 1.9–$92.1\times$ below full GNN-recipe wall times. Physics is the exception: global feature transforms dominate node/feature edits at 296–340 s; edge edits reuse them and take 25–27 s. These shared-GPU timers start from prepared edited data, not graph mutation (Appendix O).

7 Limitations and Scope

Global preprocessing can dominate refitting, as on Physics; landmark rank also trades accuracy for cost. Single-split results and the documented iterative development history bound generalization claims. Task-aware results require their declared input semantics. Structural verification covers fourteen generic datasets and both task-aware models; Questions and proteins remain label-only. Recomputations do not imply universal local speedups. Generic tests fix configuration, whereas task-aware tests replay selection. Predictor equivalence does not certify secure deletion of obsolete caches, differential privacy (Thudi et al., 2022), or online adaptation under distribution shift.

8 Conclusion

Closed-form solvers combine strong prediction with checkable unlearning: fourteen benchmarks within one point or better under one generic protocol, and task-aware gains on the other two, including Minesweeper state of the art. Across five edit types, 280 checks establish the retained-data computation; cached label updates are a median $93\times$ faster than GNN training steps.

AI USE STATEMENT

We used LLMs to aid and polish the writing; for retrieval and discovery, such as finding related work; and for research ideation and execution, including implementing and running experiments and analyzing their results. All reported numbers were checked against the retained experiment artifacts. The authors take full responsibility for the content of this paper.

ETHICS STATEMENT

This study uses existing graph benchmarks and reports aggregate predictive and unlearning results. Retrain-equivalence is a model-relative computational property, not a guarantee of legal compliance, differential privacy, or the impossibility of inferring information from retained graph structure. Structural-inversion measurements are reported as a benchmark diagnostic. Deployment requires separate attention to dataset terms, access control, and removal of retained records and obsolete computational state.

REPRODUCIBILITY STATEMENT

The appendices specify candidate grids, preprocessing, split and metric conventions, task-aware input assumptions, unlearning targets, numerical modes, and proofs. Tables and figures are generated from an artifact-backed numerical snapshot. The accompanying source includes the plotting scripts and data required to regenerate the figures, with fixed seeds and explicit separation of historical and current results.

REFERENCES

Piotr Bojanowski, Édouard Grave, Armand Joulin, and Tomáš Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010/.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *IEEE Symposium on Security and Privacy*, 2021. URL https://arxiv.org/abs/1912.03817.

Yinzhi Cao and Junfeng Yang. Towards Making Systems Forget with Machine Unlearning. In *IEEE Symposium on Security and Privacy*, pp. 463–480, 2015. doi: 10.1109/SP.2015.35. URL https://www.cs.columbia.edu/~junfeng/papers/unlearning-sp15.pdf.

Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph Unlearning. In *ACM SIGSAC Conference on Computer and Communications Security*, pp. 499–513, 2022. URL https://arxiv.org/abs/2103.14991.

Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. GNNDelete: A General Unlearning Strategy for Graph Neural Networks. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=X9yCkmT5Qrl.

Eli Chien, Chao Pan, and Olgica Milenkovic. Efficient Model Updates for Approximate Unlearning of Graph-Structured Data. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=fhcu4FBLciL.

Youngmin Cho and Lawrence K. Saul. Kernel Methods for Deep Learning. In *Advances in Neural Information Processing Systems*, volume 22, 2009. URL https://proceedings.neurips.cc/paper/2009/hash/5751ec3e9a4feab575962e78e006250d-Abstract.html.

Weilin Cong and Mehrdad Mahdavi. Efficiently Forgetting What You Have Learned in Graph Representation Learning via Projection. In *International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6674–6703, 2023a. URL https://proceedings.mlr.press/v206/cong23a.html.

Weilin Cong and Mehrdad Mahdavi. GraphEditor: An Efficient Graph Representation Learning and Unlearning Approach, 2023b. URL https://openreview.net/pdf?id=tyvshLxFUtP. Unpublished OpenReview submission (ICLR 2023 cycle).

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-Expressive Graph Transformer in Linear Time. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2403.01232.

Simon S. Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.13192.

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.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable Inception Graph Neural Networks, 2020. URL https://arxiv.org/abs/2004.11198. arXiv:2004.11198.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Zou. Making AI Forget You: Data Deletion in Machine Learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/cb79f8fa58b91d3af6c9c991f63962d3-Abstract.html.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/guo20c.html.

William W. Hager. Updating the Inverse of a Matrix. *SIAM Review*, 31(2):221–239, 1989. doi: 10.1137/1031049.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30, 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, 2020. URL https://arxiv.org/abs/2005.00687.

Zhaolin Hu, Kun Li, Hehe Fan, and Yi Yang. GraphTARIF: Linear Graph Transformer with Augmented Rank and Improved Focus. In *Proceedings of the ACM Web Conference*, 2026. doi: 10.1145/3774904.3792219. URL https://arxiv.org/abs/2510.10631v2.

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://arxiv.org/abs/2010.13993.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.02907.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1711.00165.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.08993v2. Datasets and Benchmarks Track.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1-nGgWC-.

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks, 2020. URL https://arxiv.org/abs/2007.02901. arXiv:2007.02901.

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, 1996. doi: 10.1007/978-1-4612-0745-0. URL https://glizen.com/radfordneal/bnn.book.html.

Zehao Niu, Mihai Anitescu, and Jie Chen. Graph Neural Network-Inspired Kernels for Gaussian Processes in Semi-Supervised Learning. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.05828.

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://arxiv.org/abs/2302.11640.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, volume 20, 2007. URL https://proceedings.neurips.cc/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2205.12454.

André Ribeiro, Ana Luiza Tenório, Juan Belieni, Amauri H. Souza, and Diego Mesquita. Cooperative Sheaf Neural Networks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.00647v3.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale Attributed Node Embedding. *Journal of Complex Networks*, 9(2), 2021. doi: 10.1093/comnet/cnab014. URL https://arxiv.org/abs/1909.13021.

Ryoma Sato. Training-free Graph Neural Networks and the Power of Labels as Features. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2404.19288.

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.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the Necessity of Auditable Algorithmic Definitions for Machine Unlearning. In *31st USENIX Security Symposium*, pp. 4007–4022, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/thudi.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1710.10903.

Hongwei Wang and Jure Leskovec. Combining Graph Convolutional Neural Networks and Label Propagation. *ACM Transactions on Information Systems*, 40(4):73:1–73:27, 2021. doi: 10.1145/3490478. URL https://cs.stanford.edu/people/jure/pubs/gcnlpa-tois21.pdf.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems*, volume 13, 2000. URL https://proceedings.neurips.cc/paper/2000/hash/19de10adbaa1b2ee13f77f679fa1483a-Abstract.html.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*, 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. GIF: A General Graph Unlearning Strategy via Influence Function. In *Proceedings of the ACM Web Conference*, pp. 651–661, 2023a. URL https://arxiv.org/abs/2304.02835.

Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. Certified Edge Unlearning for Graph Neural Networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2606–2617, 2023b. doi: 10.1145/3580305.3599271. URL https://doi.org/10.1145/3580305.3599271.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and Empowering Transformers for Large-Graph Representations. In *Advances in Neural Information Processing Systems*, volume 36, 2023c. URL https://arxiv.org/abs/2306.10759.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. In *International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, 2016. URL https://proceedings.mlr.press/v48/yanga16.html.

Lu Yi and Zhewei Wei. Scalable and Certifiable Graph Unlearning: Overcoming the Approximation Error Barrier. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f43567631f608dd1317c227c69dae441-Abstract-Conference.html.

Jiahao Zhang, Yilong Wang, Zhiwei Zhang, Xiaorui Liu, and Suhang Wang. Unlearning Inversion Attacks for Graph Neural Networks. In *Proceedings of the 19th ACM International Conference on Web Search and Data Mining*, pp. 934–945, 2026. doi: 10.1145/3773966.3777929. URL https://arxiv.org/abs/2506.00808.

Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael Bronstein, and Jian Tang. Fully-inductive Node Classification on Arbitrary Graphs. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2405.20445.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems*, volume 16, 2003. URL https://proceedings.neurips.cc/paper/2003/hash/87682805257e619d49b8e0dfdc14affa-Abstract.html.

Hao Zhu and Piotr Koniusz. Simple Spectral Graph Convolution. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/pdf?id=CYO5T-YjWZV.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *International Conference on Machine Learning*, pp. 912–919, 2003. URL https://mlg.eng.cam.ac.uk/pub/pdf/ZhuGhaLaf03a.pdf.

A Frozen procedures and development provenance

A.1 Candidate grids

NNGP kernels arise from infinite-width limits (Neal, 1996; Lee et al., 2018; Matthews et al., 2018); neural tangent kernels describe gradient-training dynamics (Jacot et al., 2018). GNTK addresses graph-level prediction (Du et al., 2019), whereas our covariance recursion follows the node-level construction of Niu et al. (2023). The normalized ReLU covariance transform in Equation 3 is

$$\Phi(\rho) = \{\sqrt{1-\rho^2} + (\pi - \arccos\rho)\rho\}/\pi, \quad -1 \le \rho \le 1.$$

Input covariances are normalized to unit diagonal; implementation clips correlations to this interval before evaluating the transform.

The initial generic procedure, recorded as U, contains:

- Linear S^{2}GC-style features with $q \in \{2, 16, 48\}$ and $\beta = 0.1$.
- GCN-form kernels with self-loop coefficient $c \in \{0.5, 1\}$, residual $r \in \{0, 1\}$, and depth $L \in \{1, 2, 3, 4, 6\}$.
- GraphSAGE-form kernels with $r = w_e = w_n = 1$ and $L \in \{2, 4, 6, 8\}$.
- Standardized inputs or rank-Gaussian inputs with log-degree, a normalized ReLU covariance head, and $\lambda \in \{0.003, 0.01, 0.03, 0.1, 0.3\}$.

The initial pool contains 255 readout candidates on the small-graph branch. Its optional second stage adds a Gaussian label-context kernel with weight $\eta \in \{0.5, 1, 2\}$, or concatenated context features when the selected base is linear. Candidate factorization or resource failures are logged and handled without reference to test performance.

The extended protocol U2, used for the generic results, adds multiscale linear features on all datasets and landmark GCN-/SAGE-form kernels above 50,000 nodes, with depths 1–3 and 8,192 landmarks. The multiscale linear block uses the top 256 label-free principal components when its input has more than 512 dimensions. Landmark eigendirections are retained above 10^{-6} times the largest eigenvalue. The added regularization grid is

$$\Lambda = \{0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000\}.$$

Stage one compares the stored initial label-free winner with the new candidates. Stage two evaluates the normalized label-context-plus-multiscale block at weights $\{0.5, 1, 2\}$ and reselects regularization. The initial Gaussian context variant remains eligible only if its base also wins stage one. Thus the extended label-free pool is larger, but the complete staged procedure is not a literal union of every earlier final predictor.

This distinction explains why an earlier final test score can exceed the extended procedure's test score. We report the extended procedure's selected result, rather than switching procedures after seeing that difference.

Dense covariance recursion uses $O(n^2)$ storage and $O(|E|n + n^2)$ work per layer, with a direct $O(|T|^3)$ readout factorization. A straightforward m-landmark layer uses $O(nm + m^2)$ storage and $O(nm^2 + |E|m + m^3)$ work before spectral truncation; the rank-dependent work is not eliminated by the approximation.

A.2 Development and numerical policy

The initial kernel design used validation exploration on Amazon-Ratings, Tolokers, WikiCS, and PubMed. Roman-empire and Minesweeper had prior task-specific development; the other established benchmarks had been evaluated with earlier predictors. The extensions were motivated in part by observed initial large-graph test outcomes. Candidate grids and selection rules were fixed before scoring the resulting selected candidates, but these are not claims of test-unseen research development.

Reported-suite selection also occurred during development. The sixteen-dataset headline is restricted to the named reporting suite and is not a success count over every development run. Exploratory procedures are retained separately and are not substituted into Table 1.

Exact-kernel recursion uses fp64 up to 25,000 nodes and fp32 at larger sizes; readout factorizations use fp64, with output evaluation following the recorded representation dtype. Questions uses host-resident covariance storage. Landmark construction uses a fixed seed. Deterministic verification changes the sparse execution policy, not the validation-selected configuration. Its re-executed test scores differ from the stored default-GPU scores by at most 0.077 points among the completed generic label-deletion checks.

A.3 Baseline distinctions

Local *Final Test* is the test score at the validation-selected epoch. Local *Highest Test* is a maximum over test checkpoints and is not the primary metric. Published Luo results are also validation-selected; they must not be identified with the local Highest statistic merely because their numerical values are similar.

The local proteins comparison uses only two completed GAT-L5H4 seeds. The published edge-aware GAT reference is therefore stated separately, using the tabulated 87.82 value. We do not claim superiority over this stronger reference. Small single-split margins and heterogeneous sources of variation are reported as point-estimate differences.

B Task-aware specifications

B.1 Analytic Constraint Reconstruction (ACR)

ACR is the Minesweeper construction that converts visible clues into linear constraints on unknown mine indicators. Feature column 0 indicates a hidden clue; columns 1–5 encode visible neighbor counts 0–4; column 6 indicates a count of at least five. The candidate grid uses $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ and either omits censored clues or treats them as equality-five constraints. Ties are broken deterministically using validation scores, the omission rule and regularization. The selected primary fits omit censored clues. Training-label deletion changes the set of unknown variables, residual constraints and training prior.

B.2 Typed LCF-Net (TLCF)

Definition. TLCF denotes *Typed LCF-Net*, a typed extension of the *Layer-wise Closed-Form Deep Network* (LCF-Net). It stacks analytically fitted predictors: the predictions from one layer help construct the inputs to the next. *Typed* means that different sequence and graph relations have separate aggregation operators, rather than being pooled into one undifferentiated neighborhood. It is the Roman-empire task-aware model, not the generic kernel-selection protocol.

Inputs and declared structure. Only the supplied fastText node features (Bojanowski et al., 2017), graph edges, node order and training labels are used; no external text or parser is added. Node indices are interpreted as token positions. Sequence offsets $\pm 1, \ldots, \pm 5$ and non-adjacent edges use distinct operators. The latter distinguish direction and distance ranges 2, 3–4, 5–8 and 9+, with further summaries of spanning-arc endpoints, arc interiors and self-excluded second-order neighborhoods. Equal feature rows after rounding to six decimals define word types. These are explicit input assumptions of this case study.

Four fitting steps.

1. **Construct label statistics.** Build training-label count tables for word-type n-grams, typed arc pairs and class-backoff contexts. Count features are leave-one-out for training queries. A Gaussian-kernel lexicon smoother supplies cross-fitted word-type predictions.
2. **Fit layer 1.** Ridge regression combines those statistics with typed embedding aggregates and known-label context.
3. **Refine in layers 2–4.** Gaussian KRR fits typed context from the previous predictions, retaining known training labels in neighbor slots. Per-operator ridge summaries are additional inputs. Training-row predictions are produced out of fold using five folds with a fixed node-identity map.

4. **Select the output.** Average corresponding outputs from three fold-map runs, then use validation to choose layer 2, 3, 4, their average, or a forward–backward chain readout. Chain transitions are estimated from adjacent training-label pairs. Test labels are used only to score the selected output.

The frozen runner uses four layers, first-layer ridge penalty 10^4, KRR penalty 0.01, prediction softmax temperature 0.1, and fold seeds 12345, 777 and 2024. Forward–backward candidates use temperatures $\{0.1, 0.2\}$ and transition powers $\{0.3, 0.5\}$. Validation ties use the rounded validation score and output name. The primary outputs use forward–backward variants: the reported result therefore belongs to the complete typed-feature, refinement and decoding construction, not to the closed-form layers in isolation.

All ten official splits give 91.36 ± 0.38 accuracy; development-unused splits 3–9 average 91.25. This tests the frozen design on additional splits of the same graph, not on a new dataset.

C Unlearning derivations and locality conditions

C.1 Proof of dependency-complete recomputation

Represent fitting as a directed acyclic computation with raw retained data and fixed configuration at its roots. Proceed in topological order. A cached node whose inputs are unaffected equals its fresh-retraining value and may be reused. A changed node recomputed from fresh-equivalent parent values also equals its fresh-retraining value. Induction establishes the output claim. If configuration selection is included, compute all selection-relevant quantities from the retained data and use the same deterministic tie rule before applying the same argument.

This proof covers deterministic approximate representations as well: retrain-equivalence is to the stated approximation procedure, not to an unapproximated full kernel. Finite-precision byte reproducibility additionally requires the same numerical operations and execution policy; real-arithmetic identities alone do not ensure it.

C.2 Proof of the fixed-kernel deletion identity

Partition $K_{TT} + \lambda I$ into retained and deleted rows and let its inverse be M. Standard block inversion gives

$$(K_{RR} + \lambda I)^{-1} = M_{RR} - M_{RF} M_{FF}^{-1} M_{FR}.$$

Using $\alpha_R = M_{RR} Y_R + M_{RF} Y_F$ and $\alpha_F = M_{FR} Y_R + M_{FF} Y_F$ yields

$$\alpha_R - M_{RF} M_{FF}^{-1} \alpha_F = (M_{RR} - M_{RF} M_{FF}^{-1} M_{FR}) Y_R.$$

The dependence on Y_F cancels exactly in real arithmetic. Regularization makes the required principal systems positive definite. Implementations must discard obsolete label-dependent coefficients and update or rebuild maintained state if it is retained for later requests.

For $t = |T|$, $f = |F|$, and C outputs, the coefficient correction involves an f-dimensional solve and retained/deleted block operations; inverse maintenance costs $O(t^2 f + f^3)$ in a straightforward implementation. Refreshing all predictions has its own cost. A prepared inverse is an $O(t^2)$ cache, not free preprocessing.

C.3 Conditional locality of ridge statistics

Suppose an edit changes a fixed local feature map only on rows S_H. Once old and new affected rows are available, updating $H_T^\top H_T$ and $H_T^\top Y_T$ costs $O(|S_H \cap T| D^2 + |S_H \cap T| DC)$, apart from factor updates, representation refresh, and prediction refresh. A rank-update implementation can exploit this structure; a fresh factorization still incurs its own cost.

For an L-step propagation map with unchanged preprocessing, S_H is contained in the reverse-reachable L-hop support of the changed feature or operator entries. With degree-normalized propagation, graph edits can change operator entries beyond the edited endpoints; the support must include those changes and use the relevant old and new graph structure. Global standardization, rank transforms, PCA, landmark refactorization, or fitted label feedback can invalidate this local-support premise.

Local representation changes also need not force a cubic kernel re-solve. If a symmetric kernel change is supported only on rows and columns S, then $\mathrm{rank}(\Delta K) \le 2|S|$. A corresponding low-rank factor or inverse update is possible when its numerical conditions hold. This statement is not an implemented structural-update result for the current winners: establishing S for their complete preprocessing and kernel construction is a separate requirement.

Measured diffusion locality. Thirty-five controls match full refitting within 4.6×10^{-12} on arxiv and 5×10^{-14} elsewhere, with identical predicted classes (Table 3). At $q = 2$, single arxiv edits touch 0.19–3.38% of rows and give 3.2–3.9× speedups; its batched edits give 1.0–1.3×. At PubMed's selected $q = 48$ and arxiv's tested $q = 16$, all five edits reach every node and the local path is slower. The small-graph $q = 2$ ratios span 0.7–3.6× under shared-GPU execution. These measurements support the conditional locality result, not a local-speedup claim for all selected predictors. Appendix O.5 specifies the update construction, model scores and timing protocol.

Table 3: Diffusion locality controls: ranges across five edits per dataset/radius. Ratio is full-refit time divided by local-update time; values below one favor refitting. Both include prediction refresh but exclude prepared caches and edited-array construction. Score discrepancies are rounded upward. These 35 controls are separate from the 280 selected-predictor checks.

Dataset	Radius q	Affected rows (%)	Refit/local	Max. $\lvert\Delta\rvert$
Cora	2	0.55–70.13	0.82–3.57	2.2×10^{-15}
Cora	48	91.77–93.43	0.09–0.45	6.9×10^{-15}
PubMed	2	0.10–47.56	0.72–1.28	2.8×10^{-15}
PubMed	48	100.00–100.00	0.11–0.22	5.9×10^{-15}
Amazon-Ratings	2	0.04–20.00	0.90–1.43	4.9×10^{-14}
ogbn-arxiv	2	0.19–50.37	1.02–3.94	4.6×10^{-12}
ogbn-arxiv	16	100.00–100.00	0.04–0.10	5.6×10^{-13}

D EXACTNESS AND TIMING DETAIL

Completed deterministic label-deletion coverage. Generic counts are Cora 3, CiteSeer 3, PubMed 3, Amazon-Photo 5, Amazon-Computer 5, Coauthor-CS 5, Coauthor-Physics 5, WikiCS 15, Amazon-Ratings 15, Roman-empire 15, Minesweeper 15, Tolokers 15, Squirrel 12, ogbn-arxiv 5, and ogbn-proteins 5. ACR adds 15 and TLCF adds 2. Questions adds fifteen deterministic cases, giving 158 label-deletion cases: 93 generic recomputations, 48 generic downdates, and 17 task-aware checks. The additional structural suite is reported in Appendix O.

What is compared. Generic checks compare all-node scores after updating with a fresh fixed-configuration pipeline rebuild. Recompute cases also test that changing overwritten deleted-label values leaves retained-mask predictions unchanged. Fresh-repeat checks are performed at a designated deletion size per generic split, not at every size. ACR repeats retained-data fits at every tested size. TLCF checks original-fit repeatability and deleted-label-value invariance at its two deletion sizes; it does not run a separate retained-data repeat at both sizes.

Questions precision check. Ten recomputation cases match exactly. The five split-1 downdates have maximum score discrepancy 2.4338×10^{-4} from the full pipeline and 100% argmax agreement. Its host-resident kernel is fp32. The reference casts solved coefficients to the kernel dtype before prediction; the downdate evaluates the same kernel entries in fp64. A cached fp64 full re-solve has almost the same maximum discrepancy (the reported maxima differ by less than 2×10^{-12}). This diagnoses the output-dtype comparison; it is not a direct saved-vector certificate that downdate and re-solve outputs are byte-identical. After 100 sequential deletions the discrepancy is 2.3273×10^{-4},

with unchanged argmax predictions. The discarded first run failed during a CPU sparse product beyond a 32-bit indexing size; the completed run uses output-column blocking.

Cache and state boundary. Kernels, inverse caches, and label-free features are prepared before update timing. Deletion masks are also prepared outside the timing window. Sequential tests maintain inverse and coefficient buffers but do not constitute a secure storage-erasure implementation. Results certify the stated predictor computation at its measured precision, not arbitrary saved state.

Case-study cost. ACR deletion tests rerun all twelve validation candidates and the selected solve in 0.17–0.45 s. TLCF repeats its full three-bag procedure in 214–215 s. These include selection, unlike generic fixed-configuration timings, and are therefore reported separately.

Default-GPU repeatability. Two same-seed GNN retrains under default kernels change up to 23.67% of test argmax predictions in the measured recipes, with up to a 2.88-point test-score difference. This is an execution observation, not evidence that deterministic GNN retraining is impossible. Our own sparse GPU paths also exhibit numerical variation; deterministic-mode verification and faster default-GPU timings must not be conflated.

The largest default-GPU generic unlearn/refit score discrepancy is approximately 6.5×10^{-2} on Tolokers, with minimum class agreement 99.55%. Deterministic recomputation removes this discrepancy in the checked cases. On proteins, default-GPU update error is below 9.4×10^{-5}: the 23.1 s update compares with 79–112 s fresh fits. Arxiv's corresponding times are 7.83 s and 29.05 s. Deterministic one-label AR downdates take 29–49 ms. Complete generic selection is separate, ranging from 16 s on Cora to 6,252 s across three Questions splits.

Deterministic GNN controls. Two runs each of Amazon-Ratings GAT and Roman GCN under deterministic algorithms produce byte-identical saved test-output arrays. Their mean costs are:

Table 4: Default and deterministic GNN execution costs.

	Full recipe			Training steps	
Recipe	Default (s)	Determ. (s)	Ratio	Default (s)	Determ. (s)
Amazon-Ratings GAT	127.57	1,358.15	10.65×	93.17	712.35
Roman-empire GCN	165.85	179.49	1.08×	114.85	126.06

These two architectures illustrate variable reproducibility overhead, not a universal GNN slowdown. Their training masks are unchanged; the cost comparison to a one-label CF update is a fixed-recipe runtime reference rather than a paired retrain on each deletion set.

Deterministic CF timing observations. Median one-record label-deletion times from verification mode are listed below. They include CPU-CSR sparse operations and were collected during verification workloads, so they are reported separately from the controlled default-GPU timing table.

Table 5: Deterministic-mode closed-form update observations.

Dataset	Deterministic-mode update (s)	Dataset	Deterministic-mode update (s)
Cora	0.0694	Amazon-Ratings	0.0293
CiteSeer	0.0707	Roman-empire	2.5369
PubMed	0.0751	Minesweeper	0.1311
Amazon-Photo	0.0210	Tolokers	0.4655
Amazon-Computer	1.9149	Questions	14.1966
Coauthor-CS	0.0439	Squirrel	0.1593
Coauthor-Physics	3.9890	ogbn-arxiv	13.6110
WikiCS	0.0016	ogbn-proteins	118.4014

Default-GPU proteins check. The completed five-case measurement gives update/fresh-fit score differences at most 9.34×10^{-5} and test-ROC-AUC differences below 0.0001 points. These quantify the fast numerical mode; the deterministic-mode score difference is zero.

E Retained closed-form controls

These experiments are retained from the earlier study with their original models, splits, and scope. They support specific component comparisons; they are not reruns of the current generic protocol.

E.1 LCF-Net versus a core diffusion family

A validation-selected control compares the core SGC+Ridge+Correct-and-Smooth family with layer-wise LCF-Net plus its kernel head. Five predefined splits are used except for Cora and Amazon-Photo, which use one.

Table 6: Historical named-family comparison.

Dataset	Core SGC+Ridge+C&S	LCF-Net family	Difference
Minesweeper	72.13 ± 0.87	90.56 ± 0.43	+18.43
Amazon-Ratings	45.34 ± 0.33	53.26 ± 0.22	+7.92
Roman-empire	40.20 ± 0.31	82.69 ± 0.37	+42.48
WikiCS	79.62 ± 0.28	79.72 ± 0.61	+0.10
Cora	83.40	83.70	+0.30
Amazon-Photo	94.31	96.14	+1.83

This supports the value of nonlinear label-fitted refinement relative to the stated diffusion family. It does not determine whether historical LCF-Net or the new graph kernels are preferable under a matched modern selection budget.

E.2 Controlled Gaussian-kernel scaling

On one frozen 640-dimensional ogbn-arxiv representation and the official split, validation-selected full-kernel CG, Nyström, and random Fourier features (Rahimi & Recht, 2007) attain 71.52, 68.99, and 69.91 accuracy, with peaks of 4.35, 1.09, and 2.69 GiB allocated GPU memory. A dense stored $90{,}941 \times 90{,}941$ fp32 training kernel would require 30.8 GiB.

The selected 120-iteration CG configuration takes 231.32 s for fitting plus validation prediction on an RTX 2080 Ti. The corresponding Nyström window takes 0.148 s, including landmark draw, streamed construction, solve, and validation prediction. Shared representation and bandwidth preparation, the selection grid, and test prediction are outside both windows. RFF wall time is omitted because its random-parameter draw was not recorded inside the construction window. The CG residual is approximately 2.06×10^{-2}; this is a budgeted solve, not a claim of an exact converged kernel optimum.

Matrix-free products reduce stored workspace to $O(b|T|)$ for row-block size b, but retain quadratic kernel work per iteration. This experiment concerns the earlier Gaussian head; the current Nyström graph recursion is a different representation.

E.3 Cross-fitted proteins label propagation

An earlier same-split proteins control compares feature-only ridge with direct and cross-fitted LP-Ridge:

Table 7: Historical cross-fitted proteins control.

Construction	Validation ROC-AUC	Test ROC-AUC
Feature-only ridge	75.93	70.07
Direct LP-Ridge	81.75	74.87
Two-fold cross-fitted LP-Ridge	83.55	77.43
Five-fold cross-fitted LP-Ridge	83.79	77.81

Removing the direct training-row self-label path improves the measured result. This supports cross-fitting as a useful construction choice. The current 85.73 result adds a different representation and context construction, so the difference from 77.81 is not attributed to a single isolated component.

F Retained structural-inversion measurements

The earlier diffusion–ridge experiment uses two TrendAttack variants, two edge-deletion ratios, and five runs per cell. Table 8 gives the recorded combined attack AUC, lower indicating less successful inference under that attack. Its population includes both removed-edge and ordinary-edge comparisons against nonedges; it is not solely a deleted-label membership score.

Table 8: Retained structural-inversion measurements.

Dataset	Attack	Edge deletion	Combined AUC, mean ± recorded SD
Cora	TrendAttack-MIA	5%	0.6428 ± 0.0594
Cora	TrendAttack-MIA	10%	0.7276 ± 0.0414
Cora	TrendAttack-SL	5%	0.8710 ± 0.0142
Cora	TrendAttack-SL	10%	0.9048 ± 0.0132
CiteSeer	TrendAttack-MIA	5%	0.6860 ± 0.0541
CiteSeer	TrendAttack-MIA	10%	0.6869 ± 0.0314
CiteSeer	TrendAttack-SL	5%	0.9006 ± 0.0187
CiteSeer	TrendAttack-SL	10%	0.9300 ± 0.0089
PubMed	TrendAttack-MIA	5%	0.9224 ± 0.0029
PubMed	TrendAttack-MIA	10%	0.9146 ± 0.0049
PubMed	TrendAttack-SL	5%	0.9559 ± 0.0045
PubMed	TrendAttack-SL	10%	0.9565 ± 0.0008

Published attack results on other unlearners use different target models and, for some comparisons, different deletion ratios. They are not used here to attribute a privacy advantage causally to exactness. The retained measurements instead motivate evaluating predictor equivalence and structural-inference risk as separate properties.

G Published graph-transformer references

Table 9 uses the retuned, starred entries throughout Tables 2–4 of Luo et al., arXiv v2 (Luo et al., 2024). Entries are published means, not local reruns. Tolokers is outside those tables. Comparisons summarize point estimates under the respective reported evaluation protocols.

Table 9: Published graph-transformer references.

Dataset	Generic CF	GraphGPS	SGFormer	Polynormer
Cora	84.50	83.87	84.82	83.43
CiteSeer	73.00	72.73	72.72	72.19
PubMed	81.40	79.94	80.60	79.35
Amazon-Computer	94.00	91.79	92.42	93.78
Amazon-Photo	96.54	94.89	95.58	96.57
Coauthor-CS	96.07	94.04	95.71	95.42
Coauthor-Physics	96.68	96.71	96.75	97.18
WikiCS	80.89	78.66	80.05	80.26
Squirrel	43.65	39.81	42.65	41.97
Amazon-Ratings	56.31	53.27	54.14	54.96
Roman-empire	84.20	82.72	80.01	92.66
Minesweeper	92.92	90.75	91.42	97.49
Questions	78.36	72.56	73.81	78.94
ogbn-arxiv	72.22	71.23	72.76	73.40
ogbn-proteins	85.73	77.15	79.92	79.53

Generic closed-form scores exceed 14, 12, and 9 of these 15 means respectively. The separately reported ACR result additionally exceeds the Minesweeper Polynormer reference; task-aware scores remain in their own evaluation track.

H HISTORICAL TWO-LAYER REFERENCE COMPARISON

The generic closed-form protocol numerically exceeds all ten entries in this historical two-layer reference panel (10/10). The panel combines six literature values from Shchur et al. (Shchur et al., 2018), the two-layer GAT result from Wiki-CS (Mernyei & Cangea, 2020), and three historical local two-layer results. It is a cross-protocol reference comparison, not ten matched experiments.

Table 10: Historical two-layer reference comparison.

Dataset	Generic CF	Two-layer reference	Source / protocol
Cora	84.50	81.8	Shchur et al., Table 1, GAT
CiteSeer	73.00	71.9	Shchur et al., Table 1, GCN
PubMed	81.40	78.7	Shchur et al., Table 1, GAT
Amazon-Photo	96.54	91.4	Shchur et al., Table 1, GS-mean
Coauthor-CS	96.07	91.3	Shchur et al., Table 1, GS-mean
Coauthor-Physics	96.68	93.0	Shchur et al., Table 1, GS-mean
WikiCS	80.89	79.63	Mernyei and Cangea, Table 2, GAT
Amazon-Ratings	56.31	54.49	Historical local two-layer SAGE, Highest Test
Roman-empire	84.20	83.73	Historical local two-layer SAGE, Highest Test
Minesweeper	92.92	90.72	Historical local two-layer SAGE, Highest Test

Protocol differences. Shchur et al. use the largest connected component, 20 training labels and 30 validation labels per class, averaging over 100 splits and 20 initializations. Our primary protocol (Section 5) uses substantially larger training sets on Amazon-Photo and the Coauthor datasets. Wiki-CS averages five runs on each of twenty masks; our primary comparison uses three masks. The three historical local entries use best-test checkpoints, not validation-selected Final scores. These differences prevent attributing the ten numerical improvements solely to the predictor. Table 1 remains the primary tuned-GNN evidence.

I COMPLETE UPDATE COSTS

Times are seconds. Update is the median one-record latency across tested splits; fresh fit is the median across tested deletion requests. Ratios use unrounded values. Questions pools two recomputation splits and one downdate split (0.113 s), so its median is not a single-path cost. A proteins record contains 112 task labels. GNN references are two-run original-data fixed-recipe costs on the same hardware class. Default-GPU numerical differences are stated in the main paper.

Table 11: Complete default-GPU label-update costs. Update starts from prepared caches and includes prediction refresh; initial cache creation, mask preparation, and generic reselection are separate.

Dataset	Update	Fresh CF fit	Steps/update	Full/update
Cora	0.0540	0.68	90×	383×
CiteSeer	0.0857	5.75	70×	385×
PubMed	0.0009	0.20	10,429×	45,408×
Amazon-Photo	0.0051	0.55	2,262×	13,172×
Amazon-Computer	2.3408	6.83	19×	33×
Coauthor-CS	0.0300	11.38	560×	872×
Coauthor-Physics	3.9211	117.75	4×	7×
WikiCS	0.0013	0.80	19,984×	26,806×
Amazon-Ratings	0.0289	4.37	3,225×	4,416×
Roman-empire	2.3191	5.05	50×	72×
Minesweeper	0.1236	0.57	115×	221×
Tolokers	0.3849	2.94	111×	194×
Questions	15.0939	69.51	14×	20×
Squirrel	0.1400	1.15	93×	345×
ogbn-arxiv	7.8307	29.05	44×	64×
ogbn-proteins	23.0997	99.63	—	—

Cache construction and memory. Complete split-0 representation and downdate-cache construction takes 0.16 s (PubMed), 0.55 s (WikiCS), 0.64 s (Photo), 7.75 s (Ratings) and 10.57 s (CS). These default-GPU measurements include the training kernel K_{TT}, cross-kernel K_{VT}, Cholesky factorization, explicit inverse M and fitted coefficients. Each reported total sums the separate medians of three synchronized representation and cache-construction windows; data loading and model selection are excluded. With t training nodes, M and K_{TT} each need $8t^2$ bytes in fp64, while K_{VT} needs $8nt$ bytes: at most 1.20 GB per training-sized matrix and 2.40 GB for the cross-kernel. These are individual tensor sizes, not peak memory; other matrices coexist.

For initial closed-form setup cost $F + C$ (fit plus extra cache construction), per-request update U, and GNN recipe cost G, amortization after q requests compares $F + C + qU$ with $(q + 1)G$. If $G > U$, break-even requires $q \geq (F + C - G)/(G - U)$. On all five label-free winners, even full measured cache setup plus one update costs less than one GNN training-step reference. For label-context Physics, Table 11's median fit and update costs are 117.75 s and 3.92 s, versus 16.3 s of GNN steps or a 26.5 s full recipe. Using these component costs in the same model predicts break-even after nine requests against training steps, or five against the full recipe. This is an amortization projection, not a timed sequence benchmark.

J Representation controls and numerical diagnostics

Every control is independently validation-selected within the stated candidate restriction. The restricted nonresidual GCN covariance is an internal control, not an official GNNGP reproduction.

Table 12: Matched candidate-restriction controls from the initial frozen procedure.

Dataset	Linear	GCN_0	Shallow	Full
WikiCS	79.73	80.69	80.67	80.89
Amazon-Ratings	43.04	51.84	54.65	56.31
Roman-empire	51.64	47.62	80.28	80.94
Minesweeper	74.25	82.23	89.48	90.75
Tolokers	74.43	80.64	84.50	84.98

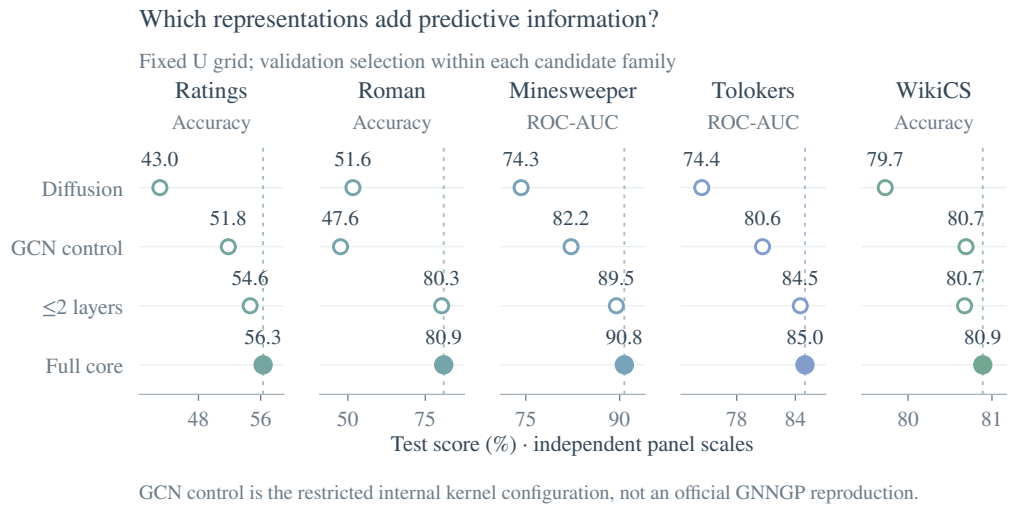

Figure 5: Independently validation-selected representation controls from the initial procedure: linear diffusion, standardized nonresidual GCN covariance, shallow ($\leq$ 2-layer) pool, and full label-free pool. Scales are independent; vertical guides mark the full-core result.



Figure 6: Numerical scope. Left: 43 fp64-kernel downdates; dashed bound 2.6×10^{-11}. The five fp32 Questions cases are in Table 2. Right: deterministic/default GNN wall-time ratios; both deterministic pairs reproduce saved test outputs exactly (Appendix D).

K Dataset scale and selected configurations

Dataset sources are Sen et al. and Planetoid for citation graphs (Sen et al., 2008; Yang et al., 2016), Shchur et al. for Amazon/Coauthor (Shchur et al., 2018), WikiCS (Mernyei & Cangea, 2020), and attributed Wikipedia graphs (Rozemberczki et al., 2021), with heterophily filtering and OGB protocols from Platonov et al. (2023); Hu et al. (2020). Table 13 gives sizes, primary split counts and split-0 configurations; each split selects independently. The numerical snapshot retains all selected tuples and validation/test scores. Squirrel uses 2,223 filtered nodes; OGB uses official temporal (arxiv) and species (proteins) splits.

Table 13: Dataset scale and representative validation selections. q is diffusion depth; L is covariance depth. Context means the extended label-context-plus-multiscale block; Gaussian means the eligible earlier Gaussian-context construction.

Dataset	Nodes	Splits	Base on split 0	Refinement
Cora	2,708	1	Diffusion, $q = 48$	Context
CiteSeer	3,327	1	Diffusion, $q = 16$	Context
PubMed	19,717	1	Diffusion, $q = 48$	None
Amazon-Photo	7,650	1	GCN kernel, $L = 3$	None
Amazon-Computer	13,752	1	GCN kernel, $L = 6$	Gaussian
Coauthor-CS	18,333	1	GCN kernel, $L = 2$	None
Coauthor-Physics	34,493	1	GCN kernel, $L = 6$	Context
WikiCS	11,701	3	GCN kernel, $L = 2$	None
Amazon-Ratings	24,492	3	SAGE kernel, $L = 6$	None
Roman-empire	22,662	3	GCN kernel, $L = 3$	Gaussian
Minesweeper	10,000	3	SAGE kernel, $L = 6$	Context
Tolokers	11,758	3	GCN kernel, $L = 6$	Gaussian
Questions	48,921	3	SAGE kernel, $L = 8$	Gaussian
Squirrel	2,223	10	GCN kernel, $L = 6$	Gaussian
ogbn-arxiv	169,343	1	Landmark GCN, $L = 3$	Context
ogbn-proteins	132,534	1	Landmark SAGE, $L = 2$	Context

L Local baseline reproductions and split variation

Final Test uses the validation-selected checkpoint; Highest Test is the maximum over test checkpoints and is included only for provenance. Table 1 uses Final. Empty cells mean that the stated local record is absent, not that the method is inapplicable.

Table 14: Local architecture-specific reference means (%). Final is validation-selected; Highest is the best test checkpoint. Proteins SAGE and GAT-L5H4 each have two completed seeds. Dashes indicate unavailable local records.

	GCN		SAGE		GAT	
Dataset	Final	Highest	Final	Highest	Final	Highest
Cora	83.50	85.12	83.48	84.28	83.56	83.98
CiteSeer	72.44	73.30	71.58	72.42	71.38	72.74
PubMed	80.54	81.20	78.40	78.86	79.88	80.26
Amazon-Photo	96.05	96.47	96.30	97.02	96.40	96.84
Amazon-Computer	94.13	94.40	93.26	93.56	93.94	94.21
Coauthor-CS	95.88	96.13	96.05	96.50	96.13	96.25
Coauthor-Physics	97.50	97.54	97.03	97.27	97.24	97.34
WikiCS	80.40	80.49	80.45	80.69	80.93	81.26
Amazon-Ratings	53.66	54.15	54.99	55.85	55.30	55.96
Roman-empire	91.14	91.45	90.70	91.02	90.05	90.62
Minesweeper	97.77	97.86	98.08	98.24	97.83	97.91
Tolokers	86.03	86.21	84.48	84.72	85.45	85.71

Table 14 (continued)

	GCN		SAGE		GAT	
Dataset	Final	Highest	Final	Highest	Final	Highest
Questions	78.19	78.34	76.11	76.44	77.31	77.41
Squirrel	44.47	46.88	40.95	41.58	40.12	42.97
ogbn-arxiv	72.93	73.56	72.79	73.15	—	—
ogbn-proteins	75.65	—	79.35	—	83.78	—

Table 15: Generic closed-form split aggregation. SD measures variation across the primary splits, not repeated identical solves. A dash means SD is undefined because only one split was evaluated.

Dataset	Primary splits	Mean (%)	Sample SD
Cora	1	84.50	—
CiteSeer	1	73.00	—
PubMed	1	81.40	—
Amazon-Photo	1	96.54	—
Amazon-Computer	1	94.00	—
Coauthor-CS	1	96.07	—
Coauthor-Physics	1	96.68	—
WikiCS	3	80.89	0.70
Amazon-Ratings	3	56.31	0.20
Roman-empire	3	84.20	0.32
Minesweeper	3	92.92	0.40
Tolokers	3	85.52	0.51
Questions	3	78.36	1.77
Squirrel	10	43.65	1.67
ogbn-arxiv	1	72.22	—
ogbn-proteins	1	85.73	—

M Development controls and alternatives

These diagnostics explain choices made during development. They are validation measurements, not additional test-selected entries in Table 1. The initial and extended protocols and their development history are specified in Appendix A.

Initial frozen protocol. Before the landmark and normalized-context extensions, the initial protocol was above or within one point of the best local Final reference on ten of its thirteen datasets. Its pool and selection rules were fixed before scoring that protocol's selected test outputs. This is a per-protocol freeze, not a claim that all research development was test-unseen: earlier experiments informed the design as described in Appendix A. The initial arxiv branch used linear features only; its 68.87 test accuracy became 72.22 under the extended protocol.

Approximation depends on the architecture. On Amazon-Ratings, the tested Monte-Carlo network features reached validation accuracy 45.89 at width 4,096 and 48.21 at width 8,192 (best over tested depths, both at depth seven). The exact depth-six SAGE-form comparison reached 56.17. Nyström was more accurate but still rank-sensitive on this graph: at depth six, 2,048, 4,096 and 8,192 landmarks gave 46.54, 49.15 and 52.45, respectively. In contrast, the tested WikiCS and Tolokers GCN-form kernels were already close to their exact-kernel validation scores at much smaller ranks:

Table 16: Validation-only landmark checks at representative depths. These are development configurations, not the selected predictors in the main test table.

Dataset	Kernel/depth	Exact	$m = 2{,}048$	4,096	8,192
Amazon-Ratings	SAGE / 6	56.17	46.54	49.15	52.45
WikiCS	GCN / 2	82.41	82.49	82.40	82.35
Tolokers	GCN / 6	84.04	84.01	84.01	84.00

The contrasting behavior motivates retaining exact kernels where feasible rather than assuming that one approximation rank works equally well for every graph architecture. The rejected Monte-Carlo construction is one finite-width design, not a general impossibility result for random-feature approximations.

Label-density shift on proteins. In the earlier label-context diagnostic, the mean fraction of labeled neighbors was 0.941 for training nodes but 0.100 for validation nodes. Individual context blocks reached up to 84.19 validation ROC–AUC, while tested joint concatenations ranged from 33.95 to 51.39. This observed covariate difference motivated normalizing by labeled mass. It does not isolate the cause of the final gain: the extended construction also changes multiscale features and regularization.

Alternatives not adopted. On an arxiv development configuration with validation accuracy 73.87, the tested Correct-and-Smooth settings reached 70.05–70.70. One round of prediction-context refinement reached 73.93, a 0.06-point increase; the tested second round reached at most 72.88. Spectral inputs, residual choices and alternative self-loop coefficients were also explored, without a consistent improvement across the compared settings. For proteins, subsequent edge-aware kernel variants reached at most 91.40 validation ROC–AUC and nonlinear Gaussian label-context heads at most 91.44, versus 91.37 for the retained configuration. Neither exploration was promoted to a new test-scored protocol. These observations are local to the tested settings, not universal rankings of these techniques.

N Complexity and computational resources

Let n be the graph size, t the training-set size, D an explicit feature width, m a landmark rank, f a deletion size, and C the output dimension. Table 17 separates construction from fitting and maintenance. Output refresh and selection are additional operations; the same feature construction can have a very different cached-deletion cost from its initial fit cost.

Table 17: Straightforward computational bounds for the stated operations. Bounds are not claims that every implementation materializes every listed object.

Operation	Work	Stored objects
Dense covariance layer	$O(\|E\|n + n^2)$	$O(n^2)$
Landmark covariance layer	$O(nm^2 + \|E\|m + m^3)$ before truncation	$O(nm + m^2)$
Primal ridge fit	$O(tD^2 + tDC + D^3)$	Features plus $D \times D$ statistics
Dense KRR fit	$O(t^3 + t^2C)$ after kernel construction	$t \times t$ training kernel
Inverse-cache maintenance	$O(t^2 f + f^3)$	$t \times t$ inverse
Feature readout refresh	$O(nDC)$	Features and output scores

Dataset handling and baseline implementations use PyTorch Geometric (Fey & Lenssen, 2019). Current experiments use single NVIDIA RTX PRO 5000 Blackwell GPUs; paired reference timings use the same GPU model. Default-GPU measurements synchronize their timing windows. Deterministic label verification uses CPU-CSR sparse products and GPU dense linear algebra, with column blocking for host-resident Questions kernels. The structural suite also verifies fixed-order GPU sparse execution (Appendix O). The label-verification runner caps PyTorch CPU threads at twelve; structural

fixed-order GPU lanes use eight CPU threads and default timing lanes twelve. The GNN timing queue caps OpenMP/MKL threads at eight. These timing measurements are distinct from historical RTX 2080 Ti controls in Appendix E. Initial representation construction, cached updates, prediction refresh and complete validation search retain the separate boundaries reported in Appendix D and Appendix I.

O STRUCTURAL DELETION: COMPLETED CHECKS AND COMPUTATION REUSE

O.1 EDITS AND RETRAINING TARGETS

The completed structural suite contains 122 distinct configurations: seven edits on each of fourteen split-0 generic winners, seven edits on each of three ACR splits, and three TLCF edits. Generic configuration tuples are the same ones used for the reported predictions; preprocessing and features are recomputed where affected, but model selection is fixed. ACR reruns its complete twelve-candidate selection protocol. TLCF reruns its full protocol, including validation selection. Questions remains label-only because the structural run requires approximately 60 GB of free host memory; proteins remains label-only because its endpoint features are sums of incident edge features, requiring a separate coupled edge-feature edit path.

Write $n = |V|$, $t = |T|$, and m for the number of undirected edges. The seven edits are: (i) remove one edge; (ii) remove $\max(1, \mathrm{round}(0.01m))$ edges; (iii) remove one training node; (iv) remove k training nodes; (v) zero one node's feature row; (vi) zero k nodes' feature rows; (vii) remove a connected breadth-first-search prefix targeting $\max(5, \mathrm{round}(0.01n))$ nodes, seeded at a training node. Here $k = \min\{\max(10, \mathrm{round}(0.01t)), \lfloor t/2 \rfloor\}$. Thus the larger feature edit uses a training-size-based count, not 1% of all nodes. Generic sampling uses seed 20260926; ACR uses seed $4000 + s$ for split s.

Edge removal deletes both directions. Node and subgraph removal delete the corresponding feature/label records and incident edges, then reindex nodes and all split masks in order. Degrees and adjacency normalization are recomputed. Subgraph size can be limited by its connected component. ACR applies the edits to the supplied feature matrix and graph; it does not regenerate clue values after graph mutation. These tests verify retrain-equivalence on the specified edited data, not robustness to an alternative regenerated Minesweeper task. TLCF tests removal of 103 non-chain arcs (1%), one training token with its features, label and incident arcs, and one token's feature row. Token deletion preserves the remaining order when reindexing.

O.2 EXACTLY WHICH STATE CAN BE REUSED?

Edge edits reuse feature-only transformations, such as rank transforms and feature projections; graph-dependent propagation, degree features and label context are refreshed. Feature edits reuse unchanged graph operators and graph/label-only context, while feature-dependent context is rebuilt. Node and subgraph edits replay the full generic fixed-configuration pipeline, including global preprocessing and fold construction after reindexing. The reference fit uses the edited inputs with no reuse cache. All output scores are compared, not just test accuracy.

The 98 generic cases have zero maximum score discrepancy and 100% argmax agreement. The edits themselves change scores by 2.88×10^{-4} to 1.016; the largest changed-prediction fraction is 8.14% (rounded bounds). ACR independently rebuilds the edited data and reruns the complete protocol for its reference. All 21 cases have zero discrepancy on defined outputs, identical undefined-score masks and identical selected configurations. Thirteen edits change the selection relative to the unedited model. Some ACR edits have zero effect on predictions; the nonzero-effect statement above pertains to the generic suite. TLCF's three cases also have zero score discrepancy and matching selection; two edits change its selected chain-decoding parameter. Their score changes range from 0.48 to 0.90. These checks concern predictor outputs, not erasure of every obsolete stored object or byte identity of the entire model state.

O.3 Deterministic sparse execution

Deterministic verification uses either CPU-CSR products or a fixed-order GPU sparse product. The latter coalesces and sorts sparse entries, groups rows by nonzero count, and accumulates their neighbor contributions in a fixed order. This avoids unordered accumulation within each output element. All 98 generic structural cases match their fresh-fit scores exactly with this GPU path. Earlier CPU-CSR checks are retained as additional verification, not counted twice. Minesweeper takes 1.45–1.92 s per edited-data recomputation in that mode. Results do not assert bit identity across different sparse implementations, devices, precisions or library versions.

Three direct benchmarks use full $n \times n$ fp64 operands on Computer and Tolokers and an $n \times 8{,}192$ fp32 operand on Physics. The fixed-order product is bit-identical across three calls; maximum relative discrepancies from CPU-CSR are 4.05×10^{-16}, 4.45×10^{-16} and 2.49×10^{-7}. Median multiplication times are 6.0, 6.9 and $18.4\times$ faster than eight-thread CPU-CSR. One-time operator preparation takes 0.759, 0.066 and 0.142 s, respectively, outside those multiplication windows. The tested cuSPARSE COO/CSR paths are faster but differ between identical calls; this is an observation about these workloads and software, not a claim about every sparse implementation.

Table 18: Completed structural suite: seven edits per split-0 generic winner. All deterministic discrepancies are zero. C = CPU-CSR, D = fixed-order GPU; the execution column gives verification/timing modes (G = default GPU). Times are medians within edit families in seconds; the node column includes subgraph removal. GNN wall time is a measured full recipe, not training steps alone.

Dataset	Modes	Edge	Feature	Node/subgraph	GNN wall	Ratio range
Cora	C,D/G	0.76	0.83	1.31	20.6	15.7–27.3
CiteSeer	C,D/G	0.76	6.48	7.65	33.0	4.3–43.3
PubMed	C,D/G	0.47	0.48	0.53	39.5	74.3–83.9
Amazon-Photo	C,D/G	1.43	1.90	1.91	66.6	34.9–46.8
Amazon-Computer	C,D/G	8.24	9.93	10.10	77.9	7.7–9.5
Coauthor-CS	D/G	11.61	13.90	14.10	26.1	1.9–2.3
Coauthor-Physics	D/D	26.03	318.09	321.58	26.5	0.1–1.0
WikiCS	C,D/G	0.99	1.22	1.20	34.0	28.0–34.4
Amazon-Ratings	D/D	10.48	10.48	10.36	127.6	12.2–12.3
Roman-empire	C,D/G	8.02	10.50	9.39	165.8	15.8–20.7
Minesweeper	C,D/G	1.32	1.22	1.36	27.3	20.0–22.3
Tolokers	C,D/G	5.96	5.85	5.96	74.7	12.5–12.8
Squirrel	C,D/G	0.52	1.52	1.63	48.3	29.7–92.1
ogbn-arxiv	D/D	115.91	55.24	122.94	497.5	4.0–9.0

O.4 Timing scope and same-model refitting

The structural timer starts after preparing and validating the edited graph. It includes affected preprocessing, representation construction, solves and all-node prediction; initial cache construction, mutation/reindexing and generic model reselection are outside the window. Each generic edited-data recompute and fresh fit is timed once with CUDA synchronization at its boundaries. Per-family medians pool two edge edits, two feature edits, or the two node edits and one subgraph edit. Measurements were collected with other workloads sharing the GPU, not under an isolated-throughput protocol. The GNN reference is the mean full-recipe wall time of two existing runs on the same hardware, including evaluation; it is not a paired retrain on each edited graph and is not a training-steps-only denominator. Table 18 identifies the timing mode in every row.

These costs can benefit from reusing unaffected computation even when there is no local structural update: the median edge-edit reuse advantage over a fresh closed-form fit is $11.0\times$ on CiteSeer and $3.0\times$ on Squirrel. ACR's full selection-aware structural fits take 0.16–0.76 s on the CPU and TLCF's full protocol takes 450–983 s on the shared machine; both are kept separate from the generic fixed-configuration GPU cost table. On Physics, rebuilding rank transforms and the 8,415-dimensional feature projection dominates node/feature/subgraph edits (296–340 s). Edge edits reuse these transforms and take 25–27 s. On arxiv, all seven deterministic edits take 51–123 s, versus a 497.5 s full GNN recipe; feature edits reuse graph and label-context objects and take 51–59 s.

A second, direct comparison uses the already-completed default-GPU label suite. For each split-model pair with a one-record deletion, divide that request's fresh full-pipeline fit time by its cached-update time. This includes the cost of rebuilding the representation in the reference; it is not merely a comparison against re-solving a precomputed kernel. The ten downdate pairs range from 107.3× to 1,362.7×, with median 272.4×. The twenty label-dependent recomputation pairs range from 2.18× to 67.09×, with median 5.99×. These are default-GPU timings, with the numerical precision qualifications of Appendix D; the fixed-order GPU structural measurements are a separate execution mode.

O.5 Locality is conditional on propagation radius

An additional 35 diffusion controls test five edits (one or 100 edges, one or ten training nodes, one feature row) at seven dataset/radius settings. These are separate from the 280 selected-predictor deletion checks. For changed operator rows S_1, propagate the affected set by $C_k = S_1 \cup C_{k-1} \cup N'(C_{k-1})$, initialized by changed feature rows. Recompute only these propagation rows, subtract old and add new training-row contributions to both ridge statistics, solve, and refresh all predictions. The reference rebuilds the same diffusion model on the edited graph.

Table 3 summarizes all seven settings and their radius-dependent costs. Small affected sets need not translate into a speedup: fixed execution overheads also matter, especially on the smaller graphs. The $q = 2$ models score 82.30, 77.70, 42.15 and 60.18 on Cora, PubMed, Ratings and arxiv, respectively; these are locality controls, not replacements for the selected predictors. Timings use a warm-up and medians of five synchronized GPU repetitions, excluding initial caches and edited-array preparation.