FROM ART TO ENGINEERING: AN ARCHITECTURE EVALUATION FRAMEWORK FOR EDGE-LEVEL DEEP GRAPH TRANSFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep graph transformation lacks controlled evaluation protocols for comparing architectures across structurally different transformation tasks. We introduce a controlled evaluation framework for edge-level deep graph transformation that places ten architectures and thirteen graph-computation and real-world tasks under a shared evaluation contract, yielding sixteen directly comparable experiments. The framework is open-source, provides reproducible baselines for analysing architecture–task interactions, together with controls over graph topology, feature provision and supervision redaction, and welcomes community contributions of new tasks, architectures, and evaluation settings. Across it, no architecture is consistently strongest, and three ablation studies show that changing the information supplied to the learner can change relative architectural performance while the transformation itself remains fixed. Dense architectures, which do not use the graph's edges to determine which nodes exchange information, can also match or exceed graph-native ones, leading in eight of the sixteen experiments. We define a task's *information regime* as the information conditions under which it is learned: what task-relevant information is supplied, the structural range over which it must be resolved, and whether predictions depend on one another. We use cross-task analysis to motivate the information regime as a fifth factor that extends an existing task categorisation framework.

1 INTRODUCTION

When developing machine learning models, we explore design alternatives and test candidate solutions. This requires general engineering knowledge about which architectures can support which types of tasks, the ability to experiment with different architectures for a specific task or dataset, and the ability to test a specific architecture on a range of tasks or datasets.

Graph transformation covers a broad spectrum of tasks that change a graph's nodes, edges, labels, attributes, or topology. *Deep Graph Transformation (DGT)* is the field of learning transformations from input to output graphs, including node-level, edge-level, and node-edge co-transformations (Guo et al., 2022). Because model performance depends on the architecture, the graph structure being transformed, the complexity of the underlying transformation pattern, and the supervision mechanism used to learn it, all of which are task-specific, existing DGT models tend to be developed around narrow application settings and lack shared evaluation protocols. The Diffusion Convolutional Recurrent Neural Network, for example, is best understood as a fixed-topology, node-level transformation model: it predicts future traffic states on a road-network graph and was originally evaluated on the METR-LA and PEMS-BAY traffic datasets (Li et al., 2018), with later work continuing to evaluate and extend it in the same setting (Shao et al., 2022). This pattern of matching models to a particular graph type, supervision design, and application domain makes it difficult to systematically verify and compare graph transformation models. While such specialisation is appropriate to the problems these methods address, it leaves architectural comparison underdetermined: because each method arrives with its own data, features, and evaluation protocol, reported differences cannot be separated from the conditions that produced them.

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.

These concerns extend beyond procedural variation. Reproducibility work in neighbouring areas shows that inconsistent metrics, hyperparameter optimisation and bespoke evaluation choices create misleading impressions of architectural superiority (Henderson et al., 2018; Lucic et al., 2018), and the same fragility appears in graph learning unless model selection and assessment are standardised.

This leaves an empirical question specific to learned graph transformation: *Which architectures are best suited to which edge-level graph transformation tasks, and which properties of a task govern this suitability*? In particular: *Can dense architectures match or exceed graph-native architectures on tasks traditionally assumed to require explicit relational inductive biases*?

To address this question, we conduct a controlled empirical comparison across ten architectures distributed among five architecture classes on edge-level deep graph transformation—transformations that modify the edge set or edge attributes of a graph while leaving the vertex set fixed—using our evaluation framework which fixes the experimental conditions under which they are compared. We deliberately restrict the scope to these transformations: fixing the vertex set allows us to study architectural suitability without also modelling variable-size graph outputs.

Applying this framework, we find that no architecture is consistently strongest across the suite and that architectural suitability depends on the task and the information available to the learner. Ablation experiments show that performance can change even for the same graph transformation task. We characterise the features supplied, their type, structural range, and output dependence as the task's information regime.

This work makes four key contributions: (i) It provides general engineering knowledge based on an empirical characterisation of architectural suitability for edge-level DGT across ten architectures; (ii) It introduces an evaluation framework for reproducible, interpretable comparisons under a shared experimental contract, allowing both the testing of a range of architectures on a specific task and the evaluation of a candidate architecture on a range of tasks, supported by configurable synthetic graph generation over ten structural graph families, task-level control of feature provision, and controllable supervision redaction; (iii) It defines the *information regime* by the features supplied, structural range and output dependence, and adds it as a fifth dimension to the task categorisation of Machowczyk & Heckel (2026); and (iv) It conducts and analyses sixteen tasks under varying information regimes, demonstrating the framework's capacity to produce controlled, directly comparable baselines, from which future domain-specific models can be developed and tested.

2 Related Work

NEC-DGT (Guo et al., 2019) learns joint node–edge translation through stacked blocks in which node and edge attribute updates are derived from one another, and is evaluated on its own combination of datasets, targets, and metrics. Our work is complementary: we do not propose a new transformation architecture, but instead hold the task, supervision, features, and metrics fixed and vary the architecture, which the comparison in the specialised literature does not supply. NEC-DGT also marks a boundary of the present study: joint node–edge transformation requires predicting and evaluating two output domains consistently, which lies outside the fixed-vertex edge-classification scope used here.

GraphGT (Du et al., 2021), a large multi-domain benchmark for graph generation and transformation with common data-processing and evaluation interfaces, is the closest existing effort. It combines broad dataset coverage with model benchmarking, including comparisons of graph-generation models across sixteen datasets and Interaction Networks (Battaglia et al., 2016) and NEC-DGT (Guo et al., 2019) across seventeen graph-transformation datasets. GraphGT's emphasis is breadth across domains, graph types, and transformation settings. Ours is complementary and deliberately narrower in task scope, focusing on a single output contract—fixed-vertex edge classification—while comparing ten architectures spanning substantially different inductive biases under shared experimental conditions. We standardise task semantics, supervision, feature provision, metrics, model selection, thresholding, checkpointing, and reporting across these comparisons. We further provide controls for graph family, feature provision, and supervision redaction, allowing graph structure and the information supplied to the architectures to be varied while the transformation itself remains fixed. GraphGT establishes common data and evaluation infrastructure across a broad transformation space; our framework uses a constrained prediction setting and controlled interventions to study architecture–task interactions directly.

A parallel line of work establishes that conclusions about graph architectures are sensitive to the protocol under which they are drawn. Standardised model selection and assessment change the ranking of graph neural networks on established benchmarks (Errica et al., 2020); robust hyperparameter search over multiple splits narrows or reverses reported gaps (Shchur et al., 2018); and systematic traversal of the GNN design space shows that architectural choices depend heavily on the task (You et al., 2020). Huang et al. (2021) press the point furthest: combining label propagation with a shallow model that ignores graph structure during training matches or exceeds state-of-the-art GNNs on standard node classification benchmarks at a fraction of the parameter count, indicating that a graph-native inductive bias is not always what accounts for performance. These results concern node- and graph-level predictions on fixed benchmark datasets. We ask the corresponding question for graph transformation, where the object being predicted is the graph structure itself, and the source graph may contain noisy or incomplete relations. To answer it, our suite spans feedforward, convolutional, attention-based, non-parametric, and graph-native architectures.

3 METHODOLOGY

We separate the evaluation design into the shared framework contract, task selection and scope, and the experimental workflow.

3.1 FRAMEWORK DESIGN

Architectural coverage, evaluation contract and method-level divergence. The suite spans ten architectures with distinct inductive biases, grouped into five architecture classes. MLP and Deep MLP (Rumelhart et al., 1986) are feedforward, CNN (LeCun et al., 1998) is convolutional, the Patch Transformer (Vaswani et al., 2017; Dosovitskiy et al., 2021) is attention-based, and the Random Forest (RF) (Breiman, 2001) is non-parametric. The Graph Transformer (Vaswani et al., 2017), GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GIN (Xu et al., 2019), and GPS (Rampášek et al., 2022) are graph-native. Feedforward, convolutional, attention-based, and non-parametric architectures are dense and operate on padded feature tensors. Graph-native architectures instead encode node representations and decode candidate edges, with pairwise features supplied to the edge decoder. A denotes the source graph's adjacency matrix: dense architectures use A to construct the feature channels requested by a task, with raw A supplied only when the adjacency channel is enabled, while graph-native architectures use A to determine which nodes exchange information. The Graph Transformer computes attention only between nodes joined by an edge in A, unlike GPS's global attention. Tables abbreviate the Random Forest as RF, the Patch Transformer as Patch Tx, and the Graph Transformer as Edge Tx. The evaluation contract fixes task semantics, supervision, feature provision, metrics, model selection, checkpointing, and reporting while permitting architecture-specific computation. Full contract and implementation details appear in Appendix A.

Experimental controls and scalable execution. Three controls determine the conditions under which a task is evaluated. A configurable synthetic graph generator sets the topology across ten structural families, explored further in Appendix A.2, and controls structural properties independently of the task definition. Feature provision is set at the task level and applied uniformly to every architecture, so a task can be run with or without structural descriptors such as common neighbour counts and adjacency powers. Supervision redaction determines whether the edges under prediction remain in A before architecture-specific processing, so those edges can be retained or withheld from the structural information available to the architectures. In the reported experiments, feature provision is varied explicitly in the three ablation studies, while topology and supervision redaction remain task-specified conditions. Separately, memory-efficient on-demand pair scoring, which limits the graph-native decoder to a fixed set of structural pair features, and tile-based batching for dense architectures make larger tasks computationally feasible without changing their evaluation conditions.

3.2 TASK SELECTION AND SCOPE

To populate the framework with a diverse, methodologically uniform task set, we adopt the task categories of Machowczyk & Heckel (2026), which categorises graph computational tasks along four dimensions: complexity, input–output relation, graph type, and required structural change. The

present task set retains all thirteen tasks from that pre-existing taxonomy that fit a common edge classification output contract. The architectures were selected separately to span distinct modelling generations and inductive biases. We restrict the scope to edge classification because it is the setting in which the object under prediction is the graph structure itself: the architecture decides which relations hold in a graph whose adjacency matrix A also determines the structural information available to the architecture; this puts an architecture's access to structure directly under test. Within this restriction, a single prediction format still spans a broad range of the remaining taxonomic dimensions, including polynomial through NP-hard complexity classes, deterministic through nondeterministic and optimisation-based input–output relations, and directed and undirected graphs with and without node and edge attributes. This selection covers all thirteen edge classification tasks: *Symmetric Closure*, *Transitive Closure*, *Shortcut Elimination*, *Shape Completion*, *Molecular Bond Inference*, *Intra-Community Edge Prediction*, *Feedback Edge Set Identification*, *Spanning Tree Identification*, *Minimum Chordal Graph Completion*, *Estimated Minimum Chordal Graph Completion*, *Algebraic Connectivity Maximisation*, *Graph Denoising*, and *Retweet Prediction*. Representative applications include molecular reconstruction, social network prediction, circuit testing, distributed communication, Bayesian inference, power grid reinforcement, and knowledge graph denoising (Machowczyk & Heckel, 2026).

There is another reason for restricting the architecture scope to edge-level deep graph transformation and the task scope to edge classification. Tasks that add or delete vertices alter the dimensionality of A, introducing representation-level complexity that would substantially expand the evaluation contract of the framework without contributing to the central question of architectural suitability within a fixed prediction format (Machowczyk & Heckel, 2026; You et al., 2018). By holding the output format constant at edge classification while varying the structural and combinatorial properties of the tasks, we isolate the relation between architecture and task structure. The taxonomic breadth of the selected tasks ensures that conclusions drawn from the framework reflect variation along the dimensions most likely to interact with architectural inductive bias, across a wide region of the DGT task space.

3.3 Experiment Workflow

All experiments follow a uniform protocol that complements the evaluation contract described in Section 3.1 with controlled repetition, seed management, and reporting. We evaluate each task over five independently seeded runs, with each task seed shared by all ten architectures; ablation conditions use five additional, independently drawn task seeds. Random seeding is deliberate: using distinct stochastic runs exposes variance across graph generation, split construction, data ordering, model initialisation, and training trajectories, enabling meaningful reporting of mean $\pm$ standard deviation and guarding against conclusions drawn from a single fortuitous or pathological run.

For real-world datasets, where graph sizes yield substantially longer per-run training times, evaluation is restricted to a single run per architecture, and variance estimates are not reported. Two real-world tasks are exceptions and follow the full five-run protocol: Algebraic Connectivity Maximisation, whose power-grid networks are small enough to permit repetition (Varbella et al., 2024), and Intra-Community Edge Prediction. Graph Denoising and Retweet Prediction remain single-run.

All edge classification tasks report a common set of metrics: F1, AU-PRC, accuracy, precision, and recall. Balanced accuracy serves as a diagnostic for trivial all-positive or all-negative behaviour that conventional accuracy can mask when class distributions are skewed. AU-ROC is recorded but not foregrounded, as AU-PRC is more informative under the class imbalance typical of edge-level tasks, where the number of candidate node pairs can greatly exceed the number of positive edges. Two tasks depart from this metric convention. Algebraic Connectivity Maximisation is trained as an addition-only edge structure completion task, but its objective is graph-level: maximising the second-smallest Laplacian eigenvalue. We therefore evaluate it by comparing base, optimal search, and model-computed λ_2 values. Graph Denoising adopts the experiment-specific metrics of the original GOLD framework (Deng et al., 2023)—Recall@k and AUC—to permit direct comparison with published results.

All experiments were carried out on NVIDIA A100-80GB and A100-40GB GPUs. [1]

[1] To preserve the double-blind peer review process, we are not including the HPC credits yet. They will be added in the camera-ready version.

4 EXPERIMENTAL ANALYSIS

The evaluation contract and workflow described in Section 3 produce a uniform set of results across all ten architectures and sixteen experiments. These comprise the thirteen tasks, three of which we also run a second time with reduced features. Apart from graph denoising, which has published task-specific baselines, none of these tasks had neural baselines under a controlled graph transformation setting before this work. In each case, the removed features carry much of the answer in the full-feature setting, so reducing them makes the same task depend on the structure the architecture derives for itself. In this section, we analyse these results through per-task examinations of how each architecture performs relative to the task's structural demands.

For analysis, we distinguish between synthetic graph-computation tasks and tasks evaluated on real-world datasets. The former comprise nine problems drawn from the graph-theoretic and graph transformation literature (Machowczyk & Heckel, 2026), procedurally generated here apart from molecular bond inference, which uses RDKit. The latter adapt four established public datasets: Intra-Community Edge Prediction using SNAP Facebook Combined (McAuley & Leskovec, 2012), Retweet Prediction using the SNAP Higgs Twitter Retweet Network (De Domenico et al., 2013), Algebraic Connectivity Maximisation using PowerGraph (Varbella et al., 2024), and Graph Denoising using ConceptNet-N20 (Deng et al., 2023), the last replicating the published GOLD protocol. Sections 4.1 and 4.2 present full analyses for one graph computation task, Transitive Closure, and one real-world task, Graph Denoising. Section 4.3 summarises the remaining eleven tasks, with full results available in Appendix B. Section 4.4 introduces the terms used in the subsequent cross-task analysis.

4.1 SYNTHETIC GRAPH COMPUTATION TASK: TRANSITIVE CLOSURE

Transitive closure requires to solve a multi-hop reachability problem on directed graphs (Plump, 2016). The canonical adjacency powers A^2–A^5 provide direct evidence of bounded reachability, since $A^k_{ij} > 0$ indicates a length-k path from node i to node j. We therefore expect them to compress architectural differences in the full-feature setting, while their bounded range may leave an advantage to architectures that can recover longer-range structure.

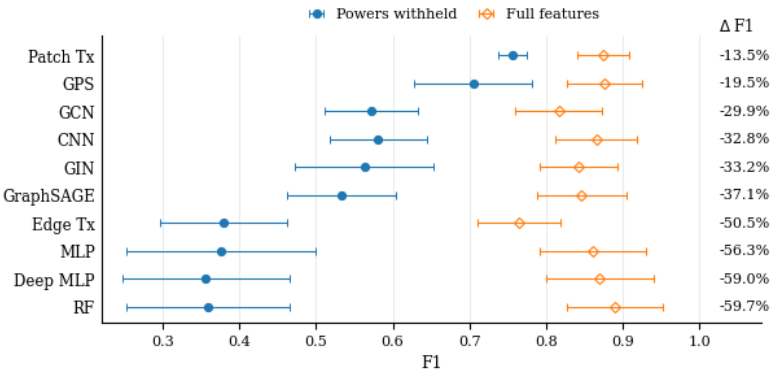

Figure 1: Transitive closure over five runs. Filled circles show F1 with adjacency powers A^2–A^5 withheld and open diamonds the full feature set; error bars show one standard deviation. Δ is the relative F1 change after withholding the powers. Rows are ordered by Δ. Full metric sets appear in the Appendix tables 6, 7 and 8.

The full-feature results in Figure 1 and Appendix Table 8 are consistent with this expectation. Performance is relatively compressed, and most notably the Random Forest achieves the highest F1 (0.8895 ± 0.0624), despite having no mechanism for deriving multi-hop structure beyond the supplied power features. GPS instead leads AU-PRC and the Patch Transformer leads recall, so the ability to recover multi-hop structure gives no consistent advantage if bounded reachability is supplied explicitly.

When we withhold A^2–A^5, requiring models to recover multi-hop structure from the remaining inputs, we expect architectures with broader structural access to degrade less. The ablated results in

Figure 1 show this separation: the Patch Transformer loses only 13.5% F1 and becomes the strongest architecture, with GPS losing 19.5%, whereas the Random Forest loses 59.7% and falls from first to ninth by ablated F1. The contrast confirms that the supplied adjacency powers substantially compress architectural differences in the full-feature regime, while removing them exposes differences in the architectures' ability to recover multi-hop structure.

We attribute the greater robustness of the Patch Transformer and GPS over MLP, Deep MLP and Random Forest to how each architecture can reconstruct the withheld reachability information. These three architectures cannot combine evidence across candidate pairs, so removing the explicit path features leaves them with no direct mechanism for deriving multi-hop reachability from the graph structure. As finite-depth message-passing architectures, GCN, GraphSAGE and GIN can propagate connectivity information across several hops, but remain limited by their local propagation depth. The Patch Transformer, by contrast, can combine information globally across its full $N \times N$ feature representation, while GPS combines local message passing with global attention.

4.2 Real-World Dataset Task: Graph Denoising

Graph denoising targets the ConceptNet commonsense knowledge graph—a medium-sized directed simple graph with 102,400 triples and approximately 20% injected noise—where the objective is to identify erroneous edges, following the denoising setting of GOLD (Deng et al., 2023) without its pre-mined logical rules or specialised architecture. The primary input signal is supplied by pre-trained BERT embeddings (Devlin et al., 2019), while A contains the injected noisy edges. Each candidate triple carries semantic features directly, while graph-native architectures can additionally use neighbourhood information defined by A. Based on GOLD's finding that local neighbour information is its most important structural component, we expect the message-passing architectures—GCN, GraphSAGE, and GIN—to achieve the highest Recall@k, with GPS and the Graph Transformer also benefiting from their graph-local computation. We expect the dense architectures to perform worse because they do not preferentially aggregate information from graph neighbours. Our framework does not reproduce GOLD's specialised local and global encoders or its pre-mined logical rules, so instead we use GOLD's published ablations as the comparison target: they show the performance remaining when individual specialised components are removed, allowing us to ask whether generic architectures recover the same signal under the restricted information and modelling assumptions used here. Table 1 reports this comparison under GOLD's Recall@k and AUC protocol, with $k = 20{,}480$ after preprocessing.

Table 1: Graph denoising on ConceptNet-N20 against the GOLD ablation ladder of Deng et al. (2023). Percentages are relative to the w/o E_{local} reference, in which GOLD's local neighbour encoder is removed. Best result within our framework per metric in **bold**. Full metrics appear in Table 9.

		GOLD AUC		Recall@k	
Study	Model	value	% of ref.	value	% of ref.
GOLD	GOLD (Sent-T5-xxl), full	0.979		0.878	
	w/o E_{global}	0.974		0.862	
	w/o PLM	0.963		0.834	
	w/o E_{local} *(reference)*	0.907	100.0	0.652	100.0
Our framework	GIN	0.8746	96.4	**0.6396**	**98.1**
	CNN	**0.8935**	**98.5**	0.6020	92.3
	⋮	⋮	⋮	⋮	⋮
	Patch Tx	0.8476	93.5	0.5572	85.5

GIN and GraphSAGE take the top two places on Recall@k, at 0.6396 and 0.6358, and the MLP, Deep MLP and Patch Transformer take the bottom three, at 0.5631, 0.5594 and 0.5572. The middle of the ranking runs against expectation. GCN, the third message-passing architecture, reaches only 0.5998, behind GPS at 0.6010 and behind the CNN at 0.6020, which has no graph-local mechanism at all; the Graph Transformer is the weakest of the five graph-native architectures at 0.5878.

Against the comparison target, GIN reaches 98.1% of the Recall@k GOLD retains without its local neighbour encoder. That encoder is the component GOLD's ablations identify as load-bearing, with

removal reducing Recall@k by 25.7%, versus 5.0% for the pre-trained language model. Despite the differences in architectures, losses and features, both our results and GOLD's ablations point to local neighbourhood information as an important source of predictive performance. Our ranking further adds that access to this information is not sufficient: all five graph-native architectures receive the same semantic features and use the same A to define their graph neighbourhoods, yet they separate by more than 0.05 Recall@k. We conclude that the difficulty of this task lies in weighing corrupt relational evidence against reliable per-candidate evidence.

4.3 Remaining Tasks

Five tasks behave as their specifications lead us to expect. Symmetric closure asks, for a candidate edge (i, j), whether the edge (j, i) is present; the supplied $A^\top$ feature provides that answer at the candidate's own position, and the dense architectures solve the task outright at F1 0.986–1.0 while GCN, GraphSAGE, GIN and the Graph Transformer plateau between 0.574 and 0.703. Shortcut elimination asks whether a length-2 alternative path makes a direct edge redundant, which no supplied feature encodes, and the ordering reverses completely: the graph-native architectures reach ≈ 0.999 while the MLP, Deep MLP and Random Forest stall at 0.664–0.677, a gap of more than 0.32. With atom-level descriptors present, molecular bond inference is largely determined by the chemical descriptors of the two atoms at either end of each candidate bond, and every architecture exceeds 0.949. Spanning tree identification admits many valid trees while supervision names one, and all ten architectures fall into a narrow band at 0.721–0.766. Intra-community edge prediction presents a reliable graph and a relational label, and on F1 the graph-native architectures lead at 0.981–0.982. In each case, the observed result is consistent with expectations based on the task specification and supplied features. The remainder of this section reports the results that diverged from those expectations.

Three tasks are limited by their target. The chordal completion pair inverts the expected difficulty ordering: the estimated variant targets a polynomial-time heuristic and should be more learnable than its NP-hard counterpart, yet proves harder for every architecture without exception, with the CNN scoring 0.632 on the exact task against 0.474 on the heuristic, and the full ten spanning 0.595–0.658 against 0.371–0.474. The heuristic's target is produced by an order-dependent procedure, and static structural features approximate a trajectory less closely than a mathematically defined optimum. Shape completion holds all ten architectures between 0.408 and 0.443, which we attribute to its formulation: the generator embeds partial motifs in background structure, so some background configurations resemble targets and confuse the architectures.

Four tasks diverge from the expectations. The Patch Transformer records the widest inversion in the suite, taking the best score of any architecture on transitive closure with the adjacency powers removed (0.7560) and the worst of any architecture on molecular bond inference with the atom descriptors reduced (0.6311, behind the next architecture by 0.063). Global attention over the full grid supports reasoning about long-range paths and does not support discrimination among local structures. GPS is the exception on symmetric closure, reaching 0.794 with the full feature set and 0.701 under ablation: the only graph-native architecture to clear 0.75 in either condition, ahead of GCN, GraphSAGE, GIN and the Graph Transformer by roughly 0.09 with features supplied and 0.22 without, though below every dense architecture in both. Minimum feedback edge set exposes differences in both ranking and selection. The graph-native architectures show cycle-membership signal, with AU-ROC between 0.776 and 0.803, but at their selected thresholds favour recall above 0.88 against precision below 0.66. The Patch Transformer ranks candidates more accurately, reaching AU-ROC 0.906, and makes a narrower selection with precision 0.828 and recall 0.810, yielding F1 0.8182. Minimality constrains the chosen set as a whole, and per-edge thresholding has no mechanism to enforce it.

The real-world dataset tasks diverge in three further ways. Intra-community edge prediction reverses between metrics: the graph-native architectures that lead F1 record balanced accuracy of 0.622–0.649, while the Random Forest leads balanced accuracy (0.846), AU-ROC (0.920) and AU-PRC (0.995). With recall between 0.991 and 0.995, the graph-native architectures accept nearly every candidate in a heavily positive evaluation set, which F1 rewards and balanced accuracy does not. Retweet prediction presents a comparable setting and supports the same expectation, but the margin is narrower: six of the ten architectures fall within 0.009 F1 of one another, the five graph-native architectures and the Patch Transformer, where intra-community separated the leading graph-native architecture from the

best dense one by 0.069. Algebraic connectivity maximisation is supplied the Fiedler vector, the eigenvector associated with the eigenvalue the task maximises, and still reaches only 57.3% of the optimal-search λ_2 gain with the Graph Transformer against 94.3% for a greedy search. With four small power-grid networks the binding constraint is lack of training signal.

4.4 Cross-Task Analysis

The task-level results show that architectural suitability depends on three recurring properties of the evaluated learning problem: (i) the task-relevant information supplied to the learner, (ii) the structural range over which the target relation must be recovered, and (iii) whether correct edge predictions can be made independently or must jointly satisfy a property of the predicted graph. The ablations provide the clearest evidence for supplied information: transitive closure, molecular bond inference, and symmetric closure all change their architectural ordering or the separation between dense and graph-native architectures when task-relevant features are withheld, even though the transformation and its classification under the four-factor task categorisation of Machowczyk & Heckel (2026) remain fixed. Structural range is reflected by the contrast between pair-local symmetric closure, bounded-pattern shortcut elimination, and multi-hop transitive closure, which increasingly require architectures to recover information beyond the candidate edge. Output dependence appears in feedback edge set, spanning tree and chordal completion, where independently predicted edges must jointly satisfy graph-level validity or minimality conditions. These task-level patterns provide the empirical basis for a systematic characterisation of architectural suitability.

We also review the architecture–experiment matrix to summarise cross-task behaviour. Across the sixteen experiments, aggregate performance is relatively close and metric wins are distributed across all ten architectures, with no architecture consistently strongest across the suite. The heatmap and clustering analyses show systematic variation but do not identify which task properties are associated with it, and the task clustering is dominated by overall performance level. Clustering the architectures does, however, recover the dense and graph-native grouping of Section 3.1 as its strongest two-way division. The supporting visualisations and clustering details appear in Appendix D.

5 Information Regimes and Task Characterisation

Table 2: Information regime of each task experiment. A dash marks a regime with a non-distinct feature set. Rows are grouped by range. Where multiple highest-performing architectures achieved the same score, all were enumerated.

Task	Defining Features	Range	Constrained	Leading Architecture(s)
Symmetric closure	transpose	local (pairs)		Deep MLP, RF, CNN
Symmetric closure (abl.)	transpose withheld	local (pairs)		CNN
Molecular bond inference	6 atom descriptors; two-hop	local (pattern)		GPS
Mol. bond inference (abl.)	3 atom descriptors; two-hop	local (pattern)		CNN
Graph denoising	node type; 96 BERT channels	local (pattern)		GIN
Shortcut elimination	A^3–A^5; no A^2	local (pattern)		GIN, GPS
Intra-community	shortest-path distances	local (pattern)		GPS
Retweet prediction	in/out degree; transpose	local (pattern)		GPS
Transitive closure	A^2–A^5	global		RF
Transitive closure (abl.)	A^2–A^5 withheld	global		Patch Tx
Spanning tree	—	global	✓	GPS
Feedback edge set	—	global	✓	Patch Tx
Algebraic connectivity	Fiedler vector	global		Edge Tx
Shape completion	shortest-path distances	local (pattern), global		RF
Minimum chordal	shortest-path distances	local (pattern), global	✓	GPS
Est. chordal	shortest-path distances	local (pattern), global	✓	CNN

We use the three properties identified in Section 4.4 to systematically characterise the experiments. The first property records the task-relevant features supplied to the learner and whether they are attached to nodes or candidate pairs. The second adapts the notion of locality from Battaglia et al. (2018) to describe the structural range over which the target relation is determined: *local (pairs)* when the relation is confined to a candidate pair and its transpose, *local (pattern)* when it depends on

a bounded structural configuration, and *global* when it depends on paths, cycles, spectral structure, or another graph-wide relation. A task may span more than one range. The third property records whether candidate predictions can be made independently or whether the selected edges must jointly satisfy a structural requirement for the output to be valid. We refer to these properties together as the task's *information regime*. This information regime captures distinctions that are not represented by the four-factor task categorisation of Machowczyk & Heckel (2026). Most directly, the three ablation pairs retain the same categorisation criteria while changing the information supplied to the learner and producing different architectural outcomes. Structural range and output dependence likewise describe aspects of the learning problem that the existing factors do not record. To address this gap, we propose information regime as a fifth factor in the categorisation.

Table 2 characterises all sixteen experiments under these properties and reports the best-performing architecture for each experiment under its primary evaluation measure. For the standard edge classification tasks, this is F1; for molecular bond inference, macro-F1; for graph denoising, Recall@k; and for algebraic connectivity maximisation, the percentage of optimal-search gain.

6 Conclusion, Future Work and Call for Contributions

In this work, we introduced a controlled evaluation framework for edge-level deep graph transformation to address the difficulty of comparing architectures across tasks developed under different experimental settings. The framework places ten architectures and thirteen tasks under a shared evaluation contract, producing sixteen directly comparable experiments. Across this suite, no architecture is consistently strongest, and the ablations show that the same graph-theoretic transformation can induce different architectural rankings under different information regimes. Dense architectures can also match or exceed graph-native ones under some information regimes, leading in eight of the sixteen experiments, including three in which no supplied feature carries the answer. To capture this dependency, we propose *information regime* as a fifth factor extending the task categorisation of Machowczyk & Heckel (2026), alongside complexity, input–output relation, graph type, and the required structural change.

The framework is intended to be developed further based on both the limitations identified here and the community needs. The framework can be extended first to edge regression and node attribute classification or regression, followed by joint node–edge and variable-vertex transformations that require broader prediction and evaluation contracts. Architecture extensibility forms a parallel development path. Compatible architectures can be integrated into the framework as dense or graph-native ones, while an evaluation API could allow externally implemented architectures to use the benchmark under the same task and evaluation conditions. At present, adding an architecture requires substantial manual rewiring when it does not already conform to the dense or graph-native interfaces, while adding new tasks is fully supported.

A broader theoretical question also remains open. Discussing graph transformation as computation requires a computational model against which tasks and architectures can be characterised. Graph rewriting is one candidate, since it has been proposed as a semantic foundation that subsumes GNN computation and may extend to other architecture classes (Machowczyk & Heckel, 2023).

The results should also be interpreted with some threats to validity in mind. Node-relabeling robustness is not varied in the reported experiments; node ordering is fixed within each dataset instance. Most importantly, several task formulations make the supervised label an imperfect representation of the underlying graph problem, which can affect what per-edge performance reveals about architectural capability. For example, Spanning Tree Identification supervises one valid solution even though multiple valid outputs may exist; Estimated Minimum Chordal Completion targets an order-dependent heuristic; and Intra-Community Edge Prediction reduces community structure to a binary intra- versus inter-community label without requiring recovery of community identity. These and the remaining task-specific threats to validity are discussed in Appendix C.

We welcome community contributions and proposals involving code, tasks, architectures, graph and feature generation, and methodological insight, which may shape further development alongside these extensions. Proposals may begin as ideas and can be refined during discussion. The specific details on how you can contribute are outlined in our open-source GitHub repository: <*redacted for peer review*>.

AI USE STATEMENT

AI was used to assist developers with both code generation and review. All outputs were evaluated before implementation. We also used AI to assist with writing and typesetting through detailed instructions. We have reviewed all AI-assisted work.

REPRODUCIBILITY STATEMENT

The architecture evaluation framework developed for this research is open-source and welcomes contribution from the community. We have redacted the name and the link to preserve author anonymity in the double-blind peer review process, but will make them public in the camera-ready version. The complete repository was anonymised and the per-run training results, including task seeds, were included in the supplementary material. Task files are submitted in the default mode. To reproduce the Symmetric Closure and Transitive Closure ablation experiments, update the feature set to match that in Table 2. For the other ablation study, the ablation mode is built into the task file and can be directly enabled.

REFERENCES

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4502–4510, 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 3:2980, October 2013. ISSN 2045-2322. doi: 10.1038/srep02980. URL http://dx.doi.org/10.1038/srep02980.

Zheye Deng, Weiqi Wang, Zhaowei Wang, Xin Liu, and Yangqiu Song. Gold: A global and local-aware denoising framework for commonsense knowledge graph noise detection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3591–3608, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.232. URL https://aclanthology.org/2023.findings-emnlp.232/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Yuanqi Du, Shiyu Wang, Xiaojie Guo, Hengning Cao, Shujie Hu, Junji Jiang, Aishwarya Varala, Abhinav Angirekula, and Liang Zhao. GraphGT: Machine learning datasets for graph generation and transformation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on*

Learning Representations (ICLR), 2020. URL https://openreview.net/forum?id=HygDF6NFPB.

Xiaojie Guo, Liang Zhao, Cameron Nowzari, Setareh Rafatirad, Houman Homayoun, and Sai Manoj Pudukotai Dinakarrao. Deep multi-attributed graph translation with node-edge co-evolution. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 250–259. IEEE, 2019. doi: 10.1109/ICDM.2019.00035.

Xiaojie Guo, Shiyu Wang, and Liang Zhao. Graph neural networks: Graph transformation. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao (eds.), *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 251–275. Springer Singapore, Singapore, 2022.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3207–3214. AAAI Press, 2018. doi: 10.1609/aaai.v32i1.11694.

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 (ICLR)*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. In *Advances in Neural Information Processing Systems*, volume 31, pp. 700–709, 2018.

Adam Machowczyk and Reiko Heckel. Graph rewriting for graph neural networks. In Maribel Fernández and Christopher M. Poskitt (eds.), *Graph Transformation*, volume 13961 of *Lecture Notes in Computer Science*, pp. 292–301, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-36709-0. doi: 10.1007/978-3-031-36709-0_16.

Adam Machowczyk and Reiko Heckel. Benchmark first: Defining tasks for graph transformation learning. In Blair Archibald and Oszkár Semeráth (eds.), *Graph Transformation*, volume 16624 of *Lecture Notes in Computer Science*, pp. 223–234, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-29730-3. doi: 10.1007/978-3-032-29730-3_13.

Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25, pp. 548–556. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf.

Detlef Plump. Reasoning about graph programs. In Andrea Corradini and Hans Zantema (eds.), *Proceedings of the 9th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2016)*, volume 225 of *Electronic Proceedings in Theoretical Computer Science*, pp. 35–44. Open Publishing Association, 2016. doi: 10.4204/EPTCS.225.6. URL https://doi.org/10.4204/EPTCS.225.6.

Ladislav Rampášek, Michael 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, pp. 14501–14515, 2022.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 1567–1577. ACM, August 2022. doi: 10.1145/3534678.3539396. URL http://dx.doi.org/10.1145/3534678.3539396.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

Anna Varbella, Kenza Amara, Blazhe Gjorgiev, Mennatallah El-Assady, and Giovanni Sansavini. PowerGraph: A power grid benchmark dataset for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 110784–110804, 2024. doi: 10.52202/079017-3517. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c7caf017cbbca1f4b368ffdc7bb8f319-Abstract-Datasets_and_Benchmarks_Track.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024. OpenReview.net. URL https://openreview.net/forum?id=sNFLN3itAd.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717. PMLR, 2018. URL https://proceedings.mlr.press/v80/you18a.html.

Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17009–17021, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c5c3d4fe6b2cc463c7d7ecba17cc9de7-Abstract.html.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9061–9073. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4be49c79f233b4f4070794825c323733-Paper.pdf.

A Framework and Evaluation Contract

This appendix summarises the implementation-level conditions that define the shared evaluation contract outlined in Section 3.1. The contract ensures that architectures are compared under documented task semantics, supervision, metrics, model-selection policies, checkpointing conventions, and reporting outputs, while leaving each free to follow its own computational pathway.

A.1 Contract elements and permitted divergence

The framework evaluates dense architectures, which operate on padded feature tensors, and graph-native architectures, which encode node representations and decode candidate edges. Dense architectures consume task-requested features as multi-channel $N \times N$ tensors, while graph-native architectures consume node and edge features and use A to determine which nodes exchange information. This difference is intentional: imposing a single representation on both would make the comparison less faithful to the architectures being evaluated.

Table 3 lists elements fixed across dense and graph-native architectures by the shared contract.

Table 3: Evaluation contract elements fixed across dense and graph-native architectures.

Contract element	**Shared condition**
Task semantics	Each task defines a source graph, a supervision matrix, directedness, and the supervised edge positions to be evaluated.
Label semantics	Binary and multiclass labels have the same interpretation across the framework. Padding is represented using masks.
Supervision masks	The same effective supervision mask determines which edge positions contribute to loss and metrics. Undirected tasks evaluate each unordered off-diagonal pair once; directed tasks evaluate ordered off-diagonal pairs according to the task mask, with diagonal entries included only when A contains a self-loop.
Feature provision	Canonical structural features are requested at the task level and made available consistently across the framework, including on-demand edge scoring. Feature ablations are task-level interventions that apply uniformly to dense and graph-native architectures and do not affect the pair scores built into the graph-native edge decoder.
Metric computation	Binary and multiclass tasks use a common metric set, with task-specific departures explicitly stated in the experiment description.
Model selection	Validation performance determines thresholding and checkpoint selection according to the rules specified by the framework. Where these rules differ, the difference is fixed across all runs of the framework.
Checkpointing	Model checkpoints are saved with task, configuration, and model metadata sufficient to identify the corresponding run.
Reporting	Results for dense and graph-native architectures are written into a common result structure and final summary, allowing a single comparison table across all architectures.
Run lifecycle	A combined run of dense and graph-native architectures on the same task shares the same task object, split construction, and run finalisation behaviour.

Permitted divergence is restricted to points where unification would distort dense or graph-native architectures. Dense architectures use padded feature tensors and dataset-level channel statistics fitted on the training split. Graph-native architectures use node and edge tensors, A, and per-graph normalisation. Their edge decoder always includes common neighbour, Jaccard and Adamic–Adar features for each candidate pair, independently of the features a task requests: an encoder that represents each node separately cannot capture the relation between the two endpoints of a candidate, and combining such structural features with message passing is standard in link prediction with graph neural networks (Zhang et al., 2021; Wang et al., 2024). On directed graphs, the common neighbour feature counts the length-2 paths between the endpoints, so it remains available to graph-native architectures when a task withholds the adjacency powers. For dense architectures, redaction of A occurs during channel assembly before requested A-derived channels are constructed, while graph-native architectures use the redacted A to determine which nodes exchange information. These divergences define how each architecture receives the task.

A.2 SYNTHETIC GRAPH GENERATOR

The synthetic graph generator is used to separate task structure from dataset idiosyncrasy. Each generated dataset is defined by one or more graph families, a graph size range, directedness, a feature request, a label function, train–validation–test ratios, and a task seed. Unless otherwise stated, synthetic experiments use datasets containing multiple generated graphs with task-specified train–validation–test ratios and one independently drawn task seed per run. The generator supports the ten structural families in Table 4. These families are intended to vary degree distribution, clustering, community structure, geometry, tree structure, and cyclic structure while preserving a common edge classification interface.

Table 4: Synthetic graph families used by the framework. N denotes the number of nodes in the generated graph.

Family	**Primary structural property**	**Directed support**
`erdos_renyi`	Uniform random connectivity. Edge probability is set to $3.5/N$, so expected degree is held constant as graph size varies.	Native
`barabasi_albert`	Preferential attachment and scale-free degree structure.	Directed analogue
`watts_strogatz`	Small-world structure with local neighbourhoods and random rewiring. The undirected form draws $k \in [2, \min(N-1, 6)]$; the directed form fixes $k = \min(2, N-1)$ and is correspondingly sparser.	Directed analogue
`random_regular`	Strict 4-regular structure, falling back to a cycle for $N \leq 4$.	Orientation-based
`stochastic_block`	Community structure with higher within-block than between-block density.	Native
`powerlaw_cluster`	Hubs combined with triadic closure and clustering.	Directed analogue
`random_geometric`	Spatial proximity induced by node positions. The undirected form connects within radius $\sqrt{3.5/(N\pi)}$, again fixing expected degree; the directed form links each node to its three nearest neighbours, giving fixed out-degree and variable in-degree.	Directed analogue
`balanced_tree`	Hierarchical tree with branching factor drawn from $[2, 4]$, grown until it admits N nodes and then truncated, so the final level is generally partial.	Root-oriented
`tree_plus_chords`	Tree backbone with additional non-tree edges.	Directed analogue
`shape_cycle`	One motif type per graph, from triangle to hexagon, placed over roughly half the nodes. One edge of each motif is always removed, and each remaining motif edge is removed with probability 0.3; the other nodes form a cycle-free background forest attached to the motifs. The completed motifs are retained as the supervision target.	Native

Topology is controlled through the synthetic graph generator. For a fixed task definition and feature request, changing the graph family changes the structural setting under which the same transformation must be learned. The framework supports comparisons of whether architectural suitability is stable across graph families or depends on specific topological conditions, although graph family is not varied independently in the reported experiments.

A.3 FEATURE ABLATION AND SUPERVISION REDACTION

The framework distinguishes feature ablation from supervision redaction. Feature ablation controls whether canonical structural features are supplied to the architectures. Table 5 lists the canonical set.

Canonical features are of two kinds. A *node* feature assigns one value to each vertex, giving a vector of length N. A *pairwise* feature assigns one value to each ordered vertex pair, giving an $N \times N$ matrix aligned with A. The distinction determines where a feature can be read. A pairwise feature

carries its value at the candidate edge's own position, so a dense architecture that indexes the pairwise feature grid reads it directly, whereas a node feature must be combined across the two endpoints. In graph-native architectures, the two kinds enter at different points: node features are consumed by the encoder as vertex inputs, while pairwise features are consumed by the edge decoder.

Table 5: Canonical features supplied by the framework. Custom, task-registered features are not included. d_{row} and d_{col} denote the row and column degrees of a candidate pair, and d_w the row degree of a shared neighbour w.

Feature	Type	Definition
`transpose`	pairwise	$A^\top$; presence of the reverse of the candidate edge.
`power_2`	pairwise	A^2, the number of length-2 paths between the pair.
`power_3`	pairwise	A^3.
`power_4`	pairwise	A^4.
`power_5`	pairwise	A^5.
`degree`	node	Row degree, that is out-degree under the directed convention.
`deg_row`	pairwise	Source endpoint degree, broadcast along rows.
`deg_col`	pairwise	Target endpoint degree, broadcast along columns; equal to `deg_row` when the graph is undirected.
`deg_diff`	pairwise	$\lvert d_{\text{row}} - d_{\text{col}} \rvert$, supplied explicitly as a non-linear cue for structural symmetry.
`triangles`	node	Closed triples per node from $\mathrm{diag}(A^3)$, halved when the graph is undirected.
`clustering_coeff`	node	Triangles over admissible pairs; the directed case uses $d_{\text{out}} d_{\text{in}}$ minus reciprocal edges.
`cn`	pairwise	Common neighbours, $AA^\top$ when undirected and AA when directed.
`jaccard`	pairwise	`cn` divided by $d_{\text{row}} + d_{\text{col}} - \texttt{cn}$, zero where that union is empty.
`adamic_adar`	pairwise	$\sum_w 1/\log d_w$ over shared neighbours w, restricted to $d_w > 1$.
`shortest_path`	pairwise	All-pairs shortest-path distance, with unreachable pairs encoded as -1.
`twohop`	node	Unique nodes at exactly distance two, excluding the source and its direct neighbours; outward reachability when directed.

The reported experiments request task-specific subsets of these canonical features explicitly. Two macros abbreviate common groups, with `powers` expanding to `power_2`–`power_5` and `endpoint_degree` to `deg_row` and `deg_col`. Removing requested features in an ablation tests whether an architecture can recover task-relevant structure without relying on those precomputed structural descriptors.

Supervision redaction controls whether supervised edges remain in A. In dense architectures, the framework redacts A before constructing requested A-derived channels, with raw A included only when the adjacency channel is enabled. In graph-native architectures, the redacted A determines which nodes exchange information. These controls are especially important for edge-level DGT because an edge being predicted may also affect the structural information available to the architecture.

A.4 Scalability and reproducibility mechanisms

The framework includes memory-efficient execution mechanisms for tasks where materialising all candidate-pair features is impractical. Dense architectures operate on padded feature tensors; MLP, Deep MLP and RF score candidate edges independently, while CNN and the Patch Transformer operate on the full padded feature grid. For graph-native architectures, the framework includes on-demand scoring of supervised pairs without materialising every $N \times N$ edge feature matrix. Intra-Community Edge Prediction and Retweet Prediction additionally use tile-based batching to process supervised edges in smaller groups. These mechanisms enable the reported experiments to run at their required graph sizes; runtime and memory scaling are not evaluated as experimental outcomes in this study.

Reproducibility is controlled through task-level seeds. A seeded task fixes graph generation, split construction, dataloader shuffling, and model initialisation. We use repeated runs with different seeds to estimate stability under the same task definition while keeping the task fixed. We report

hardware details, epoch counts, batch sizes, learning rates, and task-specific metric differences where appropriate, including in the task files and other supplementary material attached to the submission outside the appendices.

B Full Experimental Results

This appendix reports the full numerical results supporting the compact tables in the main experimental analysis. Unless otherwise stated, values are reported as mean ± standard deviation over five independent runs, with the standard deviation computed using divisor n.

B.1 Transitive Closure

Table 6: Transitive closure: full feature set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.8607 ± 0.0694	0.8716 ± 0.0634	0.9791 ± 0.0106	0.9835 ± 0.0076	0.8810 ± 0.0530	1.0000 ± 0.0000	0.7619 ± 0.1059
Deep MLP	0.8698 ± 0.0695	0.8778 ± 0.0647	0.9804 ± 0.0105	0.9845 ± 0.0076	0.8883 ± 0.0537	0.9995 ± 0.0010	0.7765 ± 0.1074
GraphSAGE	0.8464 ± 0.0591	0.8976 ± 0.0474	0.9844 ± 0.0083	0.9817 ± 0.0068	0.8719 ± 0.0450	0.9894 ± 0.0017	0.7442 ± 0.0901
RF	0.8895 ± 0.0624	0.8983 ± 0.0599	0.9839 ± 0.0101	0.9866 ± 0.0070	0.9033 ± 0.0500	1.0000 ± 0.0000	0.8067 ± 0.1001
GIN	0.8423 ± 0.0506	0.9053 ± 0.0398	0.9862 ± 0.0066	0.9811 ± 0.0061	0.8721 ± 0.0387	0.9740 ± 0.0040	0.7456 ± 0.0775
CNN	0.8653 ± 0.0530	0.9504 ± 0.0287	0.9950 ± 0.0030	0.9824 ± 0.0067	0.9137 ± 0.0387	0.9021 ± 0.0355	0.8340 ± 0.0764
GCN	0.8162 ± 0.0566	0.8945 ± 0.0419	0.9865 ± 0.0064	0.9778 ± 0.0072	0.8611 ± 0.0340	0.9339 ± 0.0423	0.7259 ± 0.0660
Patch Tx	0.8738 ± 0.0339	0.9509 ± 0.0208	0.9957 ± 0.0018	0.9823 ± 0.0056	0.9423 ± 0.0143	0.8530 ± 0.0424	0.8960 ± 0.0249
Edge Tx	0.7650 ± 0.0540	0.7536 ± 0.0828	0.8816 ± 0.0508	0.9737 ± 0.0055	0.8200 ± 0.0367	0.9544 ± 0.0177	0.6421 ± 0.0737
GPS	0.8756 ± 0.0494	0.9558 ± 0.0268	0.9955 ± 0.0030	0.9837 ± 0.0064	0.9161 ± 0.0375	0.9208 ± 0.0307	0.8375 ± 0.0743

Table 7: Transitive closure: adjacency powers A^2–A^5 ablated over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.3760 ± 0.1230	0.4024 ± 0.0967	0.9090 ± 0.0242	0.9363 ± 0.0161	0.6675 ± 0.0831	0.5033 ± 0.0969	0.3612 ± 0.1807
Deep MLP	0.3562 ± 0.1092	0.4010 ± 0.0932	0.9095 ± 0.0246	0.9314 ± 0.0141	0.6700 ± 0.0935	0.4768 ± 0.1130	0.3721 ± 0.2099
GraphSAGE	0.5328 ± 0.0704	0.6296 ± 0.0771	0.9508 ± 0.0133	0.9498 ± 0.0145	0.7322 ± 0.0491	0.6193 ± 0.1025	0.4853 ± 0.1011
RF	0.3587 ± 0.1067	0.4068 ± 0.0849	0.9156 ± 0.0189	0.9314 ± 0.0139	0.6708 ± 0.0922	0.4730 ± 0.1086	0.3737 ± 0.2066
GIN	0.5628 ± 0.0906	0.6841 ± 0.0903	0.9608 ± 0.0132	0.9542 ± 0.0120	0.7473 ± 0.0682	0.6558 ± 0.0741	0.5116 ± 0.1388
CNN	0.5811 ± 0.0633	0.6774 ± 0.0671	0.9668 ± 0.0080	0.9535 ± 0.0121	0.7640 ± 0.0494	0.6369 ± 0.0810	0.5490 ± 0.1020
GCN	0.5720 ± 0.0614	0.6854 ± 0.0639	0.9665 ± 0.0101	0.9494 ± 0.0110	0.7807 ± 0.0660	0.5905 ± 0.0758	0.5888 ± 0.1391
Patch Tx	0.7560 ± 0.0184	0.8126 ± 0.0327	0.9844 ± 0.0030	0.9702 ± 0.0067	0.8797 ± 0.0221	0.7385 ± 0.0264	0.7776 ± 0.0498
Edge Tx	0.3790 ± 0.0825	0.3963 ± 0.0743	0.7824 ± 0.0424	0.9464 ± 0.0130	0.6331 ± 0.0358	0.6268 ± 0.1302	0.2772 ± 0.0697
GPS	0.7049 ± 0.0771	0.8145 ± 0.0721	0.9832 ± 0.0082	0.9647 ± 0.0133	0.8404 ± 0.0451	0.7127 ± 0.0733	0.6994 ± 0.0868

Table 8: Transitive closure over five runs, with the full feature set and with adjacency powers A^2–A^5 withheld. Δ is the ablated F1 as a relative change from the full-feature F1. Rows are ordered by Δ. Full metric sets for both conditions appear in Tables 6 and 7.

	Full features				Powers withheld	
Model	F1	AU-PRC	Precision	Recall	F1	Δ
Patch Tx	0.8738 ± 0.0339	0.9509 ± 0.0208	0.8530 ± 0.0424	$\mathbf{0.8960 \pm 0.0249}$	$\mathbf{0.7560 \pm 0.0184}$	-13.5%
GPS	0.8756 ± 0.0494	$\mathbf{0.9558 \pm 0.0268}$	0.9208 ± 0.0307	0.8375 ± 0.0743	0.7049 ± 0.0771	-19.5%
⋮	⋮	⋮	⋮	⋮	⋮	⋮
MLP	0.8607 ± 0.0694	0.8716 ± 0.0634	$\mathbf{1.0000 \pm 0.0000}$	0.7619 ± 0.1059	0.3760 ± 0.1230	-56.3%
Deep MLP	0.8698 ± 0.0695	0.8778 ± 0.0647	0.9995 ± 0.0010	0.7765 ± 0.1074	0.3562 ± 0.1092	-59.0%
RF	$\mathbf{0.8895 \pm 0.0624}$	0.8983 ± 0.0599	$\mathbf{1.0000 \pm 0.0000}$	0.8067 ± 0.1001	0.3587 ± 0.1067	-59.7%

B.2 Graph Denoising

Graph denoising is evaluated on ConceptNet-N20 using the GOLD ranking protocol (Deng et al., 2023). Because this is a real-world dataset task and the results are reported for a single run per architecture, no mean ± std is available.

Table 9: Graph denoising on ConceptNet-N20: full single-run results for all architectures. AU-ROC represents both the framework-level metric and the noisy-over-clean ranking AUC used by the GOLD protocol (Deng et al., 2023).

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall	Recall@k
MLP	0.8995	0.9661	0.8574	0.8364	0.7522	0.9215	0.8785	0.5631
Deep MLP	0.8909	0.9679	0.8601	0.8244	0.7526	0.9238	0.8603	0.5594
GraphSAGE	0.9211	0.9703	0.8835	0.8707	0.8017	0.9375	0.9052	0.6358
RF	0.9091	0.9758	0.8905	0.8333	0.5000	0.8333	1.0000	0.5716
GIN	0.9058	0.9639	0.8746	0.8497	0.8151	0.9482	0.8671	0.6396
CNN	0.9139	0.9764	0.8935	0.8588	0.7780	0.9291	0.8992	0.6020
GCN	0.9151	0.9373	0.8169	0.8602	0.7717	0.9261	0.9044	0.5998
Patch Tx	0.9098	0.9639	0.8476	0.8501	0.7357	0.9123	0.9073	0.5572
Edge Tx	0.8822	0.9550	0.8452	0.8141	0.7716	0.9346	0.8353	0.5878
GPS	0.8741	0.9517	0.8435	0.8038	0.7774	0.9397	0.8170	0.6010

B.3 Additional Graph Computation Tasks

Table 10: Symmetric closure: full feature set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.9999 ± 0.0001	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9999 ± 0.0001	1.0000 ± 0.0000	0.9999 ± 0.0001
Deep MLP	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9999 ± 0.0001
GraphSAGE	0.7003 ± 0.0159	0.7462 ± 0.0297	0.9443 ± 0.0046	0.9728 ± 0.0056	0.7919 ± 0.0114	0.8688 ± 0.0541	0.5888 ± 0.0246
RF	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
GIN	0.5740 ± 0.0477	0.5766 ± 0.0669	0.8816 ± 0.0194	0.9597 ± 0.0065	0.7471 ± 0.0186	0.6626 ± 0.0799	0.5090 ± 0.0390
CNN	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
GCN	0.7017 ± 0.0164	0.7473 ± 0.0298	0.9449 ± 0.0060	0.9727 ± 0.0066	0.7935 ± 0.0144	0.8661 ± 0.0566	0.5928 ± 0.0315
Patch Tx	0.9864 ± 0.0083	0.9989 ± 0.0014	0.9999 ± 0.0001	0.9984 ± 0.0014	0.9919 ± 0.0054	0.9882 ± 0.0073	0.9846 ± 0.0104
Edge Tx	0.7034 ± 0.0153	0.7452 ± 0.0310	0.9371 ± 0.0065	0.9727 ± 0.0063	0.7960 ± 0.0095	0.8570 ± 0.0428	0.5978 ± 0.0206
GPS	0.7940 ± 0.0183	0.8839 ± 0.0174	0.9912 ± 0.0034	0.9771 ± 0.0062	0.8980 ± 0.0251	0.7825 ± 0.0184	0.8088 ± 0.0499

Table 11: Symmetric closure: transposed adjacency ablated over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.7162 ± 0.0147	0.7369 ± 0.0470	0.9016 ± 0.0160	0.9715 ± 0.0068	0.7875 ± 0.0200	0.9568 ± 0.0693	0.5769 ± 0.0431
Deep MLP	0.7175 ± 0.0154	0.7443 ± 0.0504	0.9066 ± 0.0170	0.9714 ± 0.0068	0.7899 ± 0.0207	0.9476 ± 0.0697	0.5822 ± 0.0447
GraphSAGE	0.4785 ± 0.0899	0.4372 ± 0.0801	0.8899 ± 0.0202	0.9309 ± 0.0225	0.7345 ± 0.0571	0.4582 ± 0.0819	0.5127 ± 0.1270
RF	0.7245 ± 0.0187	0.7656 ± 0.0437	0.9173 ± 0.0139	0.9713 ± 0.0078	0.7989 ± 0.0224	0.9191 ± 0.0512	0.6023 ± 0.0485
GIN	0.4197 ± 0.1162	0.3391 ± 0.1029	0.8211 ± 0.0334	0.9103 ± 0.0543	0.7137 ± 0.0686	0.3963 ± 0.1186	0.4929 ± 0.1725
CNN	0.7260 ± 0.0143	0.7757 ± 0.0386	0.9256 ± 0.0112	0.9711 ± 0.0080	0.8023 ± 0.0256	0.9150 ± 0.0800	0.6096 ± 0.0565
GCN	0.4745 ± 0.0870	0.4316 ± 0.0798	0.8872 ± 0.0187	0.9292 ± 0.0256	0.7338 ± 0.0571	0.4537 ± 0.0775	0.5131 ± 0.1301
Patch Tx	0.7066 ± 0.0178	0.7539 ± 0.0319	0.9231 ± 0.0157	0.9677 ± 0.0101	0.8011 ± 0.0145	0.8426 ± 0.0463	0.6105 ± 0.0300
Edge Tx	0.4785 ± 0.0928	0.4277 ± 0.0782	0.8681 ± 0.0148	0.9332 ± 0.0196	0.7295 ± 0.0567	0.4671 ± 0.0803	0.4988 ± 0.1214
GPS	0.7006 ± 0.0373	0.7414 ± 0.0120	0.9805 ± 0.0054	0.9604 ± 0.0099	0.8626 ± 0.0452	0.6651 ± 0.0243	0.7507 ± 0.0962

Table 12: Shortcut elimination: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.6637 ± 0.0846	0.7518 ± 0.1021	0.9019 ± 0.0301	0.8690 ± 0.0160	0.7764 ± 0.0454	0.7191 ± 0.0762	0.6227 ± 0.1100
Deep MLP	0.6669 ± 0.0859	0.7524 ± 0.0979	0.9035 ± 0.0250	0.8711 ± 0.0167	0.7778 ± 0.0475	0.7280 ± 0.0714	0.6228 ± 0.1152
GraphSAGE	0.9985 ± 0.0008	0.9999 ± 0.0001	1.0000 ± 0.0000	0.9993 ± 0.0004	0.9992 ± 0.0005	0.9979 ± 0.0013	0.9990 ± 0.0008
RF	0.6773 ± 0.0798	0.7600 ± 0.0893	0.9123 ± 0.0183	0.8740 ± 0.0119	0.7851 ± 0.0417	0.7270 ± 0.0652	0.6353 ± 0.0918
GIN	0.9993 ± 0.0006	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9997 ± 0.0002	0.9993 ± 0.0005	0.9999 ± 0.0002	0.9986 ± 0.0010
CNN	0.7674 ± 0.0662	0.8647 ± 0.0616	0.9565 ± 0.0124	0.9034 ± 0.0135	0.8524 ± 0.0415	0.7683 ± 0.0408	0.7704 ± 0.0960
GCN	0.9990 ± 0.0008	0.9999 ± 0.0001	1.0000 ± 0.0000	0.9996 ± 0.0004	0.9994 ± 0.0005	0.9988 ± 0.0014	0.9991 ± 0.0009
Patch Tx	0.7698 ± 0.0575	0.8520 ± 0.0609	0.9529 ± 0.0124	0.8996 ± 0.0136	0.8627 ± 0.0298	0.7402 ± 0.0548	0.8026 ± 0.0644
Edge Tx	0.9987 ± 0.0017	0.9999 ± 0.0001	1.0000 ± 0.0000	0.9993 ± 0.0010	0.9992 ± 0.0007	0.9981 ± 0.0035	0.9993 ± 0.0010
GPS	0.9993 ± 0.0005	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9997 ± 0.0003	0.9995 ± 0.0004	0.9992 ± 0.0010	0.9994 ± 0.0009

Table 13: Shape completion: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.4141 ± 0.0054	0.2420 ± 0.0225	0.8341 ± 0.0231	0.9842 ± 0.0005	0.7669 ± 0.0023	0.3340 ± 0.0066	0.5449 ± 0.0046
Deep MLP	0.4166 ± 0.0025	0.2549 ± 0.0266	0.8082 ± 0.0259	0.9844 ± 0.0004	0.7668 ± 0.0024	0.3374 ± 0.0017	0.5446 ± 0.0046
GraphSAGE	0.4186 ± 0.0037	0.2973 ± 0.0072	0.9140 ± 0.0032	0.9866 ± 0.0003	0.7312 ± 0.0055	0.3773 ± 0.0078	0.4705 ± 0.0112
RF	0.4427 ± 0.0056	0.3114 ± 0.0059	0.8578 ± 0.0042	0.9892 ± 0.0008	0.7060 ± 0.0109	0.4753 ± 0.0308	0.4170 ± 0.0227
GIN	0.4095 ± 0.0031	0.2836 ± 0.0056	0.9080 ± 0.0034	0.9860 ± 0.0003	0.7318 ± 0.0040	0.3616 ± 0.0059	0.4723 ± 0.0081
CNN	0.4161 ± 0.0029	0.2141 ± 0.0016	0.8607 ± 0.0026	0.9844 ± 0.0004	0.7658 ± 0.0026	0.3373 ± 0.0018	0.5427 ± 0.0052
GCN	0.4250 ± 0.0044	0.3003 ± 0.0040	0.9140 ± 0.0032	0.9868 ± 0.0004	0.7332 ± 0.0025	0.3850 ± 0.0059	0.4742 ± 0.0049
Patch Tx	0.4082 ± 0.0026	0.2146 ± 0.0028	0.8613 ± 0.0026	0.9843 ± 0.0003	0.7592 ± 0.0014	0.3321 ± 0.0025	0.5294 ± 0.0028
Edge Tx	0.4177 ± 0.0072	0.3021 ± 0.0073	0.9154 ± 0.0036	0.9868 ± 0.0007	0.7265 ± 0.0057	0.3825 ± 0.0145	0.4608 ± 0.0118
GPS	0.4334 ± 0.0037	0.3237 ± 0.0051	0.9231 ± 0.0025	0.9874 ± 0.0006	0.7306 ± 0.0066	0.4041 ± 0.0146	0.4685 ± 0.0138

Table 14: Molecular bond inference: full atom-level feature set over five runs. Macro-averaged metrics are reported.

Model	F1 macro	AU-PRC macro	AU-ROC macro	Accuracy	Balanced Acc.	Precision macro	Recall macro
MLP	0.9742 ± 0.0035	0.9971 ± 0.0006	0.9992 ± 0.0001	0.9756 ± 0.0030	0.9796 ± 0.0038	0.9692 ± 0.0052	0.9796 ± 0.0038
Deep MLP	0.9738 ± 0.0032	0.9971 ± 0.0005	0.9992 ± 0.0001	0.9756 ± 0.0024	0.9820 ± 0.0045	0.9663 ± 0.0050	0.9820 ± 0.0045
GraphSAGE	0.9976 ± 0.0014	0.9998 ± 0.0002	1.0000 ± 0.0000	0.9987 ± 0.0006	0.9978 ± 0.0015	0.9974 ± 0.0017	0.9978 ± 0.0015
RF	0.9750 ± 0.0023	0.9967 ± 0.0008	0.9992 ± 0.0001	0.9762 ± 0.0024	0.9800 ± 0.0020	0.9702 ± 0.0030	0.9800 ± 0.0020
GIN	0.9497 ± 0.0062	0.9863 ± 0.0036	0.9976 ± 0.0008	0.9708 ± 0.0037	0.9449 ± 0.0069	0.9550 ± 0.0070	0.9449 ± 0.0069
CNN	0.9977 ± 0.0011	0.9999 ± 0.0001	1.0000 ± 0.0000	0.9988 ± 0.0006	0.9978 ± 0.0012	0.9975 ± 0.0013	0.9978 ± 0.0012
GCN	0.9980 ± 0.0014	0.9999 ± 0.0002	0.9999 ± 0.0002	0.9990 ± 0.0005	0.9982 ± 0.0017	0.9978 ± 0.0016	0.9982 ± 0.0017
Patch Tx	0.9845 ± 0.0031	0.9955 ± 0.0020	0.9990 ± 0.0007	0.9912 ± 0.0015	0.9807 ± 0.0030	0.9886 ± 0.0036	0.9807 ± 0.0030
Edge Tx	0.9983 ± 0.0014	0.9999 ± 0.0002	1.0000 ± 0.0000	0.9992 ± 0.0006	0.9985 ± 0.0016	0.9981 ± 0.0017	0.9985 ± 0.0016
GPS	0.9984 ± 0.0012	0.9998 ± 0.0004	0.9999 ± 0.0002	0.9993 ± 0.0005	0.9981 ± 0.0015	0.9987 ± 0.0010	0.9981 ± 0.0015

Table 15: Molecular bond inference: atom-level features partially ablated over five runs. Macro-averaged metrics are reported.

Model	F1 macro	AU-PRC macro	AU-ROC macro	Accuracy	Balanced Acc.	Precision macro	Recall macro
MLP	0.6953 ± 0.0044	0.7156 ± 0.0106	0.9116 ± 0.0038	0.7518 ± 0.0064	0.7011 ± 0.0105	0.7079 ± 0.0077	0.7011 ± 0.0105
Deep MLP	0.6971 ± 0.0055	0.7153 ± 0.0072	0.9122 ± 0.0026	0.7504 ± 0.0059	0.7105 ± 0.0061	0.6994 ± 0.0072	0.7105 ± 0.0061
GraphSAGE	0.7458 ± 0.0056	0.8086 ± 0.0037	0.9543 ± 0.0007	0.8249 ± 0.0042	0.7522 ± 0.0103	0.7472 ± 0.0055	0.7522 ± 0.0103
RF	0.6952 ± 0.0045	0.7235 ± 0.0071	0.9133 ± 0.0027	0.7516 ± 0.0053	0.7031 ± 0.0084	0.7031 ± 0.0109	0.7031 ± 0.0084
GIN	0.6944 ± 0.0048	0.7519 ± 0.0027	0.9321 ± 0.0025	0.7863 ± 0.0076	0.6919 ± 0.0058	0.7049 ± 0.0082	0.6919 ± 0.0058
CNN	0.7894 ± 0.0094	0.8496 ± 0.0062	0.9674 ± 0.0015	0.8623 ± 0.0015	0.7895 ± 0.0182	0.7954 ± 0.0063	0.7895 ± 0.0182
GCN	0.7472 ± 0.0050	0.8139 ± 0.0077	0.9574 ± 0.0019	0.8322 ± 0.0048	0.7531 ± 0.0086	0.7481 ± 0.0074	0.7531 ± 0.0086
Patch Tx	0.6311 ± 0.0077	0.6620 ± 0.0108	0.8908 ± 0.0118	0.7468 ± 0.0068	0.6149 ± 0.0066	0.6545 ± 0.0107	0.6149 ± 0.0066
Edge Tx	0.7556 ± 0.0065	0.8203 ± 0.0069	0.9619 ± 0.0017	0.8441 ± 0.0055	0.7611 ± 0.0096	0.7572 ± 0.0065	0.7611 ± 0.0096
GPS	0.7736 ± 0.0033	0.8345 ± 0.0043	0.9692 ± 0.0019	0.8642 ± 0.0068	0.7860 ± 0.0069	0.7678 ± 0.0047	0.7860 ± 0.0069

Table 16: Minimum feedback edge set: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.6840 ± 0.0686	0.7744 ± 0.0563	0.7756 ± 0.0233	0.7026 ± 0.0216	0.6978 ± 0.0269	0.7339 ± 0.0733	0.6486 ± 0.1012
Deep MLP	0.6880 ± 0.0705	0.7736 ± 0.0528	0.7771 ± 0.0210	0.7055 ± 0.0219	0.6998 ± 0.0275	0.7326 ± 0.0675	0.6552 ± 0.1016
GraphSAGE	0.7493 ± 0.0404	0.7844 ± 0.0418	0.7940 ± 0.0235	0.7009 ± 0.0269	0.6929 ± 0.0217	0.6532 ± 0.0550	0.8823 ± 0.0331
RF	0.7062 ± 0.0434	0.7514 ± 0.0400	0.7670 ± 0.0184	0.6949 ± 0.0223	0.6918 ± 0.0213	0.6906 ± 0.0551	0.7237 ± 0.0356
GIN	0.7409 ± 0.0440	0.7561 ± 0.0582	0.7757 ± 0.0271	0.6872 ± 0.0276	0.6777 ± 0.0211	0.6392 ± 0.0543	0.8843 ± 0.0367
CNN	0.7743 ± 0.0410	0.8677 ± 0.0387	0.8673 ± 0.0119	0.7719 ± 0.0166	0.7698 ± 0.0138	0.7755 ± 0.0634	0.7789 ± 0.0614
GCN	0.7498 ± 0.0402	0.7804 ± 0.0470	0.7944 ± 0.0234	0.6946 ± 0.0271	0.6856 ± 0.0228	0.6428 ± 0.0539	0.9035 ± 0.0316
Patch Tx	0.8182 ± 0.0313	0.9010 ± 0.0406	0.9057 ± 0.0164	0.8181 ± 0.0179	0.8193 ± 0.0179	0.8284 ± 0.0529	0.8099 ± 0.0226
Edge Tx	0.7532 ± 0.0389	0.7960 ± 0.0413	0.8034 ± 0.0226	0.7026 ± 0.0246	0.6944 ± 0.0197	0.6515 ± 0.0529	0.8959 ± 0.0305
GPS	0.7541 ± 0.0345	0.7825 ± 0.0468	0.7968 ± 0.0266	0.7066 ± 0.0200	0.6993 ± 0.0194	0.6577 ± 0.0501	0.8874 ± 0.0297

Table 17: Spanning tree identification: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.7342 ± 0.0059	0.7542 ± 0.0083	0.8058 ± 0.0341	0.7279 ± 0.0331	0.7348 ± 0.0436	0.6474 ± 0.0120	0.8488 ± 0.0248
Deep MLP	0.7346 ± 0.0060	0.7559 ± 0.0098	0.8066 ± 0.0344	0.7280 ± 0.0338	0.7349 ± 0.0440	0.6474 ± 0.0119	0.8499 ± 0.0249
GraphSAGE	0.7527 ± 0.0048	0.8161 ± 0.0028	0.8404 ± 0.0245	0.7511 ± 0.0301	0.7562 ± 0.0386	0.6726 ± 0.0098	0.8554 ± 0.0268
RF	0.7207 ± 0.0045	0.7572 ± 0.0064	0.7914 ± 0.0336	0.7187 ± 0.0308	0.7254 ± 0.0381	0.6430 ± 0.0107	0.8204 ± 0.0210
GIN	0.7506 ± 0.0097	0.8242 ± 0.0065	0.8435 ± 0.0214	0.7484 ± 0.0234	0.7559 ± 0.0327	0.6675 ± 0.0093	0.8572 ± 0.0119
CNN	0.7507 ± 0.0045	0.8062 ± 0.0113	0.8369 ± 0.0277	0.7489 ± 0.0259	0.7567 ± 0.0335	0.6692 ± 0.0084	0.8552 ± 0.0139
GCN	0.7447 ± 0.0069	0.8070 ± 0.0061	0.8316 ± 0.0241	0.7362 ± 0.0313	0.7438 ± 0.0425	0.6520 ± 0.0090	0.8694 ± 0.0310
Patch Tx	0.7420 ± 0.0051	0.7979 ± 0.0147	0.8312 ± 0.0297	0.7441 ± 0.0263	0.7504 ± 0.0306	0.6704 ± 0.0169	0.8329 ± 0.0361
Edge Tx	0.7590 ± 0.0083	0.8345 ± 0.0069	0.8512 ± 0.0173	0.7613 ± 0.0224	0.7665 ± 0.0298	0.6851 ± 0.0075	0.8510 ± 0.0193
GPS	0.7655 ± 0.0069	0.8480 ± 0.0055	0.8591 ± 0.0176	0.7699 ± 0.0253	0.7733 ± 0.0310	0.6971 ± 0.0050	0.8493 ± 0.0221

Table 18: Minimum chordal graph completion: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.6416 ± 0.0065	0.6845 ± 0.0060	0.9524 ± 0.0011	0.9415 ± 0.0034	0.8316 ± 0.0080	0.5919 ± 0.0254	0.7026 ± 0.0215
Deep MLP	0.6332 ± 0.0054	0.6804 ± 0.0068	0.9523 ± 0.0012	0.9407 ± 0.0029	0.8238 ± 0.0071	0.5887 ± 0.0208	0.6864 ± 0.0185
GraphSAGE	0.6195 ± 0.0111	0.6762 ± 0.0074	0.9579 ± 0.0016	0.9382 ± 0.0042	0.8174 ± 0.0180	0.5758 ± 0.0335	0.6754 ± 0.0426
RF	0.6033 ± 0.0051	0.6176 ± 0.0154	0.9420 ± 0.0020	0.9343 ± 0.0048	0.8129 ± 0.0165	0.5515 ± 0.0284	0.6704 ± 0.0414
GIN	0.6156 ± 0.0062	0.6741 ± 0.0075	0.9602 ± 0.0021	0.9381 ± 0.0023	0.8130 ± 0.0148	0.5742 ± 0.0212	0.6661 ± 0.0345
CNN	0.6322 ± 0.0171	0.7154 ± 0.0176	0.9555 ± 0.0031	0.9425 ± 0.0020	0.8146 ± 0.0156	0.6038 ± 0.0134	0.6644 ± 0.0328
GCN	0.6048 ± 0.0094	0.6646 ± 0.0103	0.9564 ± 0.0020	0.9372 ± 0.0050	0.8028 ± 0.0192	0.5745 ± 0.0398	0.6449 ± 0.0469
Patch Tx	0.5954 ± 0.0206	0.6486 ± 0.0213	0.9441 ± 0.0054	0.9355 ± 0.0023	0.7992 ± 0.0251	0.5601 ± 0.0180	0.6390 ± 0.0548
Edge Tx	0.6363 ± 0.0122	0.7043 ± 0.0109	0.9657 ± 0.0022	0.9405 ± 0.0049	0.8303 ± 0.0302	0.5908 ± 0.0430	0.7008 ± 0.0707
GPS	0.6582 ± 0.0068	0.7577 ± 0.0120	0.9732 ± 0.0022	0.9364 ± 0.0019	0.8842 ± 0.0127	0.5491 ± 0.0107	0.8229 ± 0.0290

Table 19: Estimated minimum chordal graph completion: full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.3799 ± 0.0042	0.3330 ± 0.0031	0.8580 ± 0.0108	0.8702 ± 0.0207	0.7019 ± 0.0091	0.3068 ± 0.0118	0.5027 ± 0.0387
Deep MLP	0.3831 ± 0.0041	0.3381 ± 0.0025	0.8599 ± 0.0112	0.8746 ± 0.0241	0.6988 ± 0.0126	0.3181 ± 0.0233	0.4906 ± 0.0514
GraphSAGE	0.4037 ± 0.0043	0.3574 ± 0.0087	0.8764 ± 0.0133	0.8754 ± 0.0139	0.7198 ± 0.0061	0.3240 ± 0.0045	0.5354 ± 0.0087
RF	0.3714 ± 0.0049	0.2921 ± 0.0054	0.8486 ± 0.0114	0.8550 ± 0.0153	0.7127 ± 0.0041	0.2821 ± 0.0069	0.5440 ± 0.0112
GIN	0.3901 ± 0.0055	0.3357 ± 0.0032	0.8633 ± 0.0112	0.8699 ± 0.0139	0.7137 ± 0.0055	0.3093 ± 0.0054	0.5284 ± 0.0160
CNN	0.4737 ± 0.0123	0.4910 ± 0.0194	0.9141 ± 0.0125	0.8945 ± 0.0155	0.7597 ± 0.0083	0.3918 ± 0.0164	0.5999 ± 0.0137
GCN	0.4028 ± 0.0039	0.3554 ± 0.0056	0.8777 ± 0.0133	0.8751 ± 0.0156	0.7191 ± 0.0081	0.3239 ± 0.0087	0.5340 ± 0.0213
Patch Tx	0.4399 ± 0.0151	0.4330 ± 0.0221	0.8946 ± 0.0148	0.8861 ± 0.0169	0.7393 ± 0.0096	0.3605 ± 0.0176	0.5653 ± 0.0163
Edge Tx	0.4103 ± 0.0034	0.3778 ± 0.0070	0.8836 ± 0.0126	0.8734 ± 0.0134	0.7298 ± 0.0072	0.3241 ± 0.0063	0.5594 ± 0.0083
GPS	0.4305 ± 0.0060	0.3974 ± 0.0124	0.8951 ± 0.0123	0.8797 ± 0.0132	0.7414 ± 0.0089	0.3432 ± 0.0057	0.5773 ± 0.0074

B.4 Additional Real-World Dataset Tasks

Table 20: Intra-community edge prediction on the SNAP Facebook Combined dataset (McAuley & Leskovec, 2012): full result set over five runs.

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.9060 ± 0.0106	0.9926 ± 0.0022	0.8756 ± 0.0346	0.8337 ± 0.0167	0.8048 ± 0.0381	0.9889 ± 0.0043	0.8362 ± 0.0206
Deep MLP	0.8959 ± 0.0116	0.9925 ± 0.0028	0.8768 ± 0.0363	0.8178 ± 0.0185	0.8135 ± 0.0337	0.9904 ± 0.0034	0.8182 ± 0.0201
GraphSAGE	0.9813 ± 0.0014	0.9900 ± 0.0028	0.8583 ± 0.0356	0.9638 ± 0.0026	0.6494 ± 0.0333	0.9717 ± 0.0027	0.9912 ± 0.0029
RF	0.9130 ± 0.0068	0.9953 ± 0.0015	0.9201 ± 0.0201	0.8454 ± 0.0113	0.8463 ± 0.0207	0.9925 ± 0.0023	0.8453 ± 0.0128
GIN	0.9818 ± 0.0011	0.9903 ± 0.0019	0.8453 ± 0.0294	0.9647 ± 0.0022	0.6235 ± 0.0470	0.9696 ± 0.0040	0.9945 ± 0.0025
CNN	0.9082 ± 0.0072	0.9928 ± 0.0022	0.8743 ± 0.0299	0.8371 ± 0.0118	0.8012 ± 0.0329	0.9883 ± 0.0033	0.8402 ± 0.0121
GCN	0.9814 ± 0.0009	0.9911 ± 0.0019	0.8687 ± 0.0264	0.9639 ± 0.0018	0.6221 ± 0.0305	0.9695 ± 0.0028	0.9937 ± 0.0033
Patch Tx	0.8201 ± 0.0563	0.9724 ± 0.0078	0.6241 ± 0.0615	0.7045 ± 0.0749	0.5836 ± 0.0381	0.9693 ± 0.0053	0.7152 ± 0.0821
Edge Tx	0.9814 ± 0.0014	0.9933 ± 0.0017	0.8935 ± 0.0145	0.9638 ± 0.0026	0.6323 ± 0.0368	0.9703 ± 0.0028	0.9927 ± 0.0036
GPS	0.9821 ± 0.0015	0.9901 ± 0.0045	0.8564 ± 0.0534	0.9654 ± 0.0030	0.6476 ± 0.0357	0.9715 ± 0.0028	0.9930 ± 0.0020

Table 21: Retweet prediction: full single-run results for all architectures on the SNAP Higgs Twitter Retweet Network dataset (De Domenico et al., 2013).

Model	F1	AU-PRC	AU-ROC	Accuracy	Balanced Acc.	Precision	Recall
MLP	0.8067	0.9149	0.9692	0.9288	0.9139	0.7365	0.8917
Deep MLP	0.8126	0.9155	0.9691	0.9317	0.9143	0.7490	0.8880
GraphSAGE	0.8525	0.9162	0.9676	0.9532	0.8963	0.8984	0.8110
RF	0.7837	0.8723	0.9544	0.9202	0.8992	0.7146	0.8676
GIN	0.8521	0.9107	0.9650	0.9525	0.9002	0.8847	0.8219
CNN	0.8109	0.9206	0.9692	0.9309	0.9140	0.7458	0.8886
GCN	0.8531	0.9160	0.9672	0.9529	0.8997	0.8892	0.8197
Patch Tx	0.8470	0.9054	0.9596	0.9514	0.8940	0.8901	0.8079
Edge Tx	0.8507	0.8903	0.9556	0.9518	0.9007	0.8789	0.8242
GPS	0.8552	0.9159	0.9678	0.9536	0.9010	0.8914	0.8219

Table 22: Algebraic connectivity maximisation: λ_2 values and percentage of optimal-search gain over five runs on the PowerGraph dataset (Varbella et al., 2024). The greedy search is included as a non-neural reference.

Model	Base λ_2	Final λ_2	Optimal Search λ_2	Model Gain
MLP	0.0956	0.2398 ± 0.0816	0.6295 ± 0.0134	$27.0 \pm 15.3\%$
Deep MLP	0.0956	0.2073 ± 0.1113	0.6295 ± 0.0134	$21.4 \pm 21.9\%$
GraphSAGE	0.0956	0.2794 ± 0.0290	0.6295 ± 0.0134	$34.5 \pm 6.0\%$
RF	0.0956	0.2509 ± 0.0732	0.6295 ± 0.0134	$29.0 \pm 13.2\%$
GIN	0.0956	0.2931 ± 0.0524	0.6295 ± 0.0134	$37.1 \pm 10.3\%$
CNN	0.0956	0.2242 ± 0.0890	0.6295 ± 0.0134	$24.0 \pm 16.5\%$
GCN	0.0956	0.2885 ± 0.0216	0.6295 ± 0.0134	$36.2 \pm 4.1\%$
Patch Tx	0.0956	0.3764 ± 0.1138	0.6295 ± 0.0134	$52.9 \pm 22.6\%$
Edge Tx	0.0956	0.4011 ± 0.0293	0.6295 ± 0.0134	$57.3 \pm 6.3\%$
GPS	0.0956	0.2897 ± 0.1287	0.6295 ± 0.0134	$36.6 \pm 24.3\%$
Greedy Search	0.0956	0.5989 ± 0.0202	0.6295 ± 0.0134	$94.3 \pm 2.9\%$

C THREATS TO VALIDITY AND TASK-SPECIFIC CAVEATS

This appendix summarises task-specific caveats that affect the interpretation of the experimental results. The purpose is to identify cases where task formulation, feature availability, supervision design, or dataset scale influence the conclusions drawn from the framework. Where a graph-computation task has a defining algorithm, that algorithm generates the supervision labels.

C.1 FEATURE AVAILABILITY AND TASK FORMULATION

Symmetric closure. Symmetric closure is intentionally simple: it requires recognising that an ordered pair is positive whenever either direction of the edge is present. Its simplicity is a limitation if interpreted as a realistic application task, but it is useful as a diagnostic probe because it tests whether architectures can exploit a direct per-edge structural rule. The full-feature setting supplies $A^\top$ as a pairwise feature, which makes the pattern directly available to architectures that can use it. The corresponding ablation provides the more conservative view of performance when this direct structural cue is removed.

Transitive closure and shortcut elimination. In the full-feature transitive closure setting, bounded adjacency powers A^2–A^5 are provided as canonical structural features. These features provide direct evidence of finite-length reachability. The dataset generator used in the study generates paths of length up to 140, while the power features only provide the answer for paths of length up to 5. The ablation results in Section 4.1 measure how strongly each architecture depends on this bounded reachability evidence. Shortcut elimination is evaluated in its length-2 form. The common neighbour feature of the graph-native edge decoder is independent of feature provision; on these directed tasks it counts the length-2 paths between the endpoints of a candidate and evaluates to the same values as A^2, which the transitive closure ablation withholds and shortcut elimination does not request. Graph-native architectures keep this information in both experiments, and in shortcut elimination a non-zero count coincides exactly with a positive label.

Shape completion. Shape completion evaluates both existing edges and missing edges used to complete local motifs. This makes the task partly auxiliary: architectures are rewarded for recognising already-present support structures as well as predicting absent completion edges. Because existing edges are directly observable, this formulation may inflate conventional edge classification metrics relative to an evaluation restricted only to non-edges. The generator also embeds partial cycle motifs in background structure, so some background configurations resemble target shapes without being them, which contributes to the low scores recorded across the suite.

Molecular bond inference. Molecular bond inference extends the framework with RDKit-generated molecular graphs, domain-specific atom descriptors, and bond type labels. Generated molecules are sanitised with RDKit before conversion to graph tensors. These choices make the task more realistic as a chemistry-inspired edge typing problem, but they also introduce domain-specific design decisions that may affect generality. In particular, some atom-level features are highly informative. For this reason, Section 4.3 reports both the full-feature setting and an ablation in which the rich atom descriptor set is reduced, making the feature dependence explicit.

C.2 OPTIMISATION TASKS AND NON-UNIQUE TARGETS

Minimum feedback edge set. The feedback edge set experiment uses the undirected formulation. This is important because complexity and algorithmic properties differ between directed and undirected variants. For an undirected graph, deleting the non-tree edges of a spanning forest gives a minimum cardinality feedback edge set, but the optimum is generally non-unique. The reported task uses one deterministic spanning-tree-derived target set per graph.

Spanning tree identification. Spanning tree identification is non-unique: a connected graph may admit many valid spanning trees. The reported experiments train on a single selected target tree per graph. This creates a target-selection bias, because an architecture may identify structurally valid alternatives that are penalised if they differ from the chosen target. The narrow performance band observed across architectures is plausibly partly attributable to target non-uniqueness, which limits

what the band reveals about architectural capacity. Training attempts based on arborescence, golden trees, structure-based objective functions, and output distribution all performed worse than this.

Minimum chordal graph completion. The exact minimum chordal completion task relies on precomputed minimum fill-ins generated outside the main task file. This introduces a replication burden because the task loader expects the specific output format produced by the generator. In addition, graphs are sampled from Erdős–Rényi families, so the result may not generalise to other graph families. As with spanning tree identification, a graph can admit multiple minimum chordal completions, and training on a single completion may penalise alternative valid solutions.

Estimated minimum chordal graph completion. The estimated chordal completion task uses the output of NetworkX's greedy `complete_to_chordal_graph` procedure as the learning target. As such, the task evaluates whether architectures can imitate a particular order-dependent heuristic. The canonical mathematical optimum is the target of the exact task. This distinction is central to interpreting the result in Section 4.3: the estimated task may be harder to learn than the exact task because the target reflects a sequential algorithmic process, while the exact task targets a static optimum.

C.3 Real-world dataset tasks

Intra-community edge prediction. The intra-community task labels whether an edge lies within a detected community without requiring prediction of the specific community identity. Results should be interpreted as measuring whether architectures can distinguish intra-community from inter-community ties under the supplied graph representation.

Retweet prediction. The reported retweet experiment constructs A from training positives only, so validation and test edges are absent from A. Global node degree features are derived from this A, and supervised positive targets are removed from both the induced adjacency constructed from A and the degree counts of the tiles that supervise them. Negative candidates are sampled against the full positive set at a fixed five-to-one ratio and split alongside the positives, so all architectures are scored against the same candidate distribution.

Algebraic connectivity maximisation. Algebraic connectivity maximisation differs from the other edge classification tasks because its objective is graph-level: improving λ_2, the second-smallest Laplacian eigenvalue. The reported model-gain score uses final graph quality in place of per-edge classification metrics. The task is also data-limited: it uses four small power-grid networks (Varbella et al., 2024), which leads to high variance across runs, and several architectures report standard deviations exceeding forty percent of their own means. Architectures are trained against one-step greedy edge-gain targets because direct training against globally optimal multi-edge targets was empirically less stable.

Graph denoising. Graph denoising differs from the other tasks in both data modality and evaluation protocol. It follows the ConceptNet-N20 setting from GOLD and uses pre-trained language model embeddings as the primary semantic signal. The comparison to GOLD is the same in meaning but not identical to a reimplementation: the loss, architecture, and available features differ from the original implementation. RF converged to an all-positive solution (balanced accuracy 0.500), the only degenerate outcome among the 160 model–experiment combinations; broader tuning could test whether this failure is configuration-specific, while its detection illustrates the role of balanced accuracy as a diagnostic.

D Additional Cross-Task Analysis

This appendix provides the supporting visualisations and clustering details for the cross-task analysis in Section 4.4. All heatmaps use a common 0–5 score: the sum of the common metric set of Section 3.3, except for Algebraic Connectivity Maximisation, which uses five times the model-gain fraction. The figures label the Patch Transformer as Transformer.

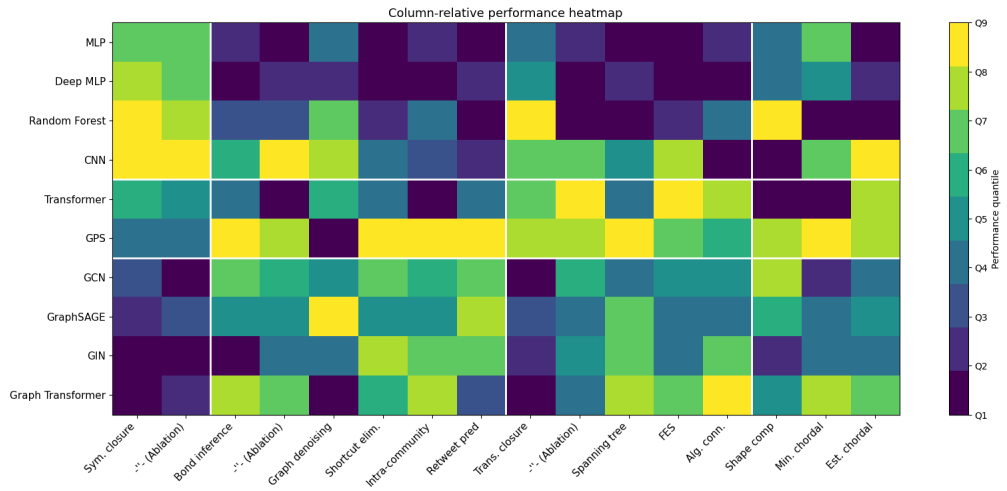

Figure 2: Column-relative heatmap of the architecture–task matrix. Each cell sums five metrics into a 0–5 score and is assigned to one of nine within-task quantile bins, with Q9 the highest within a task. Rows are grouped by the architecture clusters of Figure 4 and columns by structural range.

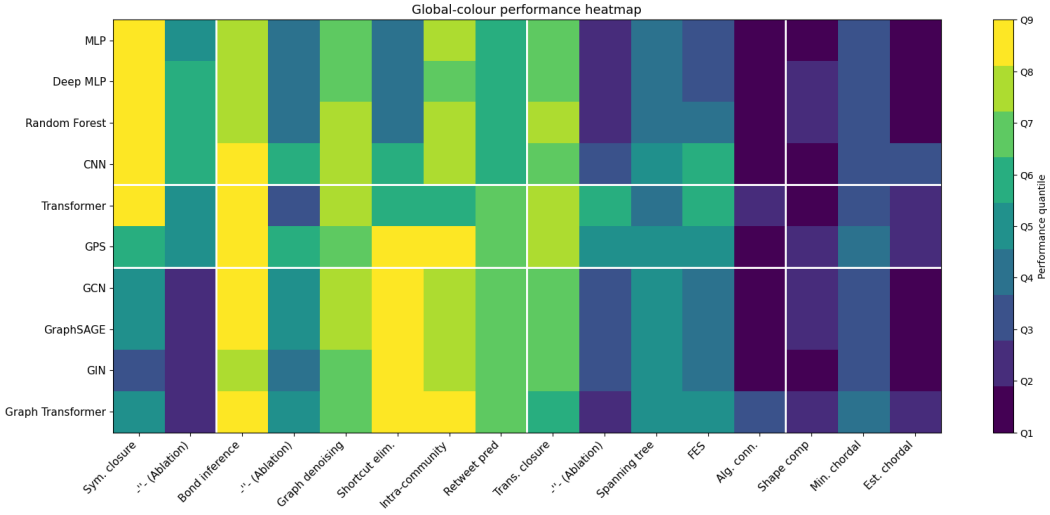

Figure 3: Global-binned heatmap of the full architecture–task result matrix. Each cell shows the same summed score across five evaluation metrics shown in Figure 2, ranging from 0 to 5. Cells are assigned to nine quantile bins over the full matrix, with Q9 denoting the highest scores globally. Rows and columns follow the same ordering and grouping as Figure 2. Where the column-relative panel shows which architectures a task favours, this panel shows how far any architecture gets.

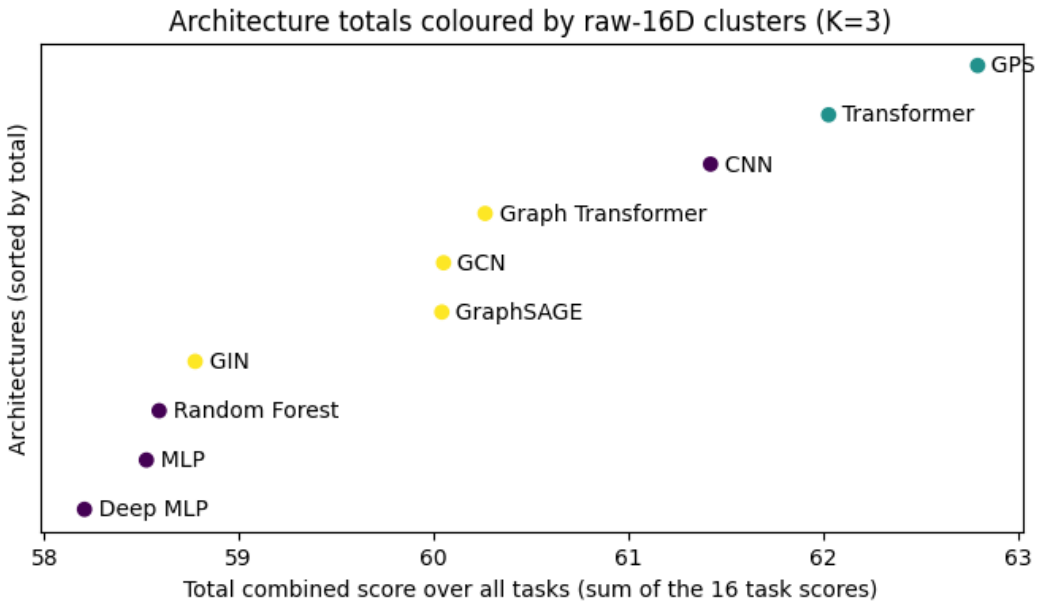

Figure 4: Architecture clustering in task-performance space. Each architecture is represented by its vector of scores across the sixteen experiments. The three clusters are: {Deep MLP, MLP, Random Forest, CNN}, {GIN, GraphSAGE, GCN, Graph Transformer}, and {Patch Transformer, GPS}. Colour provides a redundant visual encoding of these memberships.

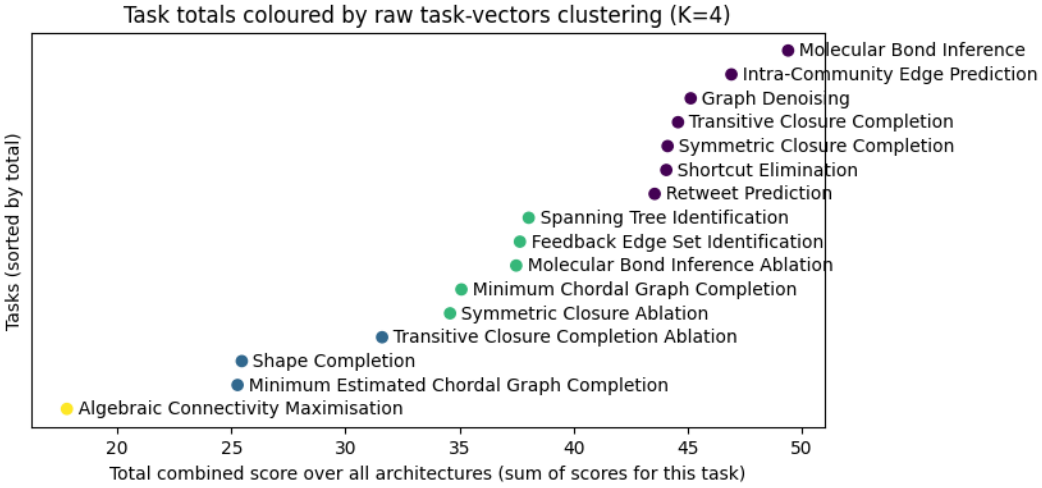

Figure 5: Task clustering from architecture-performance vectors. The four clusters contain: {Algebraic Connectivity Maximisation}; {Shape Completion, Estimated Minimum Chordal Completion, Transitive Closure Ablation}; {Symmetric Closure Ablation, Minimum Chordal Completion, Molecular Bond Inference Ablation, Feedback Edge Set Identification, Spanning Tree Identification}; and {Retweet Prediction, Shortcut Elimination, Symmetric Closure, Transitive Closure, Graph Denoising, Intra-Community Edge Prediction, Molecular Bond Inference}. Colour provides a redundant visual encoding of these memberships.

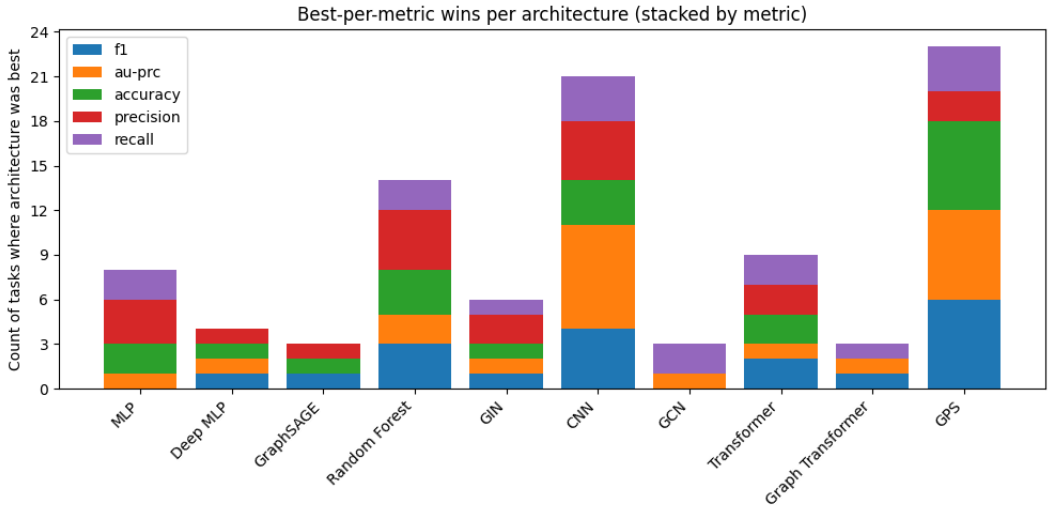

Figure 6: Per-metric win distribution across architectures. A win is counted when an architecture achieves the best score for a given task–metric slot. Tied architectures each receive a win. Independently of colour, stacked segments are ordered from bottom to top as F1, AU-PRC, accuracy, precision, and recall. Wins are distributed across all ten architectures, with no single globally dominant architecture.

Table 23: Silhouette scores for the K-means clusterings of Figures 4 and 5, computed over the same 0–5 score matrix. A dash marks a value of K outside the reported search range.

Clustering	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$
Architectures (16-D score vectors)	0.460	0.446	0.383	0.402	—
Experiments (10-D score vectors)	0.518	0.476	0.413	0.369	0.387

Figures 4 and 5 are reported at $K = 3$ and $K = 4$, which are the partitions we found most interpretable; Table 23 records the silhouette scores across the search range so that this choice is visible. The two-cluster partition of the architectures is informative in its own right: it separates {MLP, Deep MLP, Random Forest, CNN, Patch Transformer} from {GCN, GraphSAGE, GIN, Graph Transformer, GPS}, which is exactly the dense and graph-native grouping defined in Section 3.1, and it records the highest silhouette of any architecture clustering in the table.