

A APPENDIX

A.1 PROOFS AND TECHNICAL DETAILS

Preliminaries. All encoders are deterministic functions of their inputs. Mutual information identities use the chain rule and the (strong) data processing inequality (SDPI).

Lemma 2 (Sufficiency of \mathcal{S}^* under A1) *If $Y \perp (G \setminus \mathcal{S}^*) \mid \mathcal{S}^*$, then $I(G; Y) = I(\mathcal{S}^*; Y)$.*

$Y - (\mathcal{S}^*) - (G \setminus \mathcal{S}^*)$ is a Markov chain. By DPI, $I(G; Y) \leq I(\mathcal{S}^*; Y)$. The reverse inequality holds since \mathcal{S}^* is a (deterministic) function of G . Hence equality.

Theorem 1 Under **A1–A3**, if $\gamma \geq \eta$ then $I(H(G); Y) \geq I(f_{\text{global}}(G); Y)$, with strict inequality when $\gamma > \eta$. By **A2**, $I(H; Y) \geq \gamma I(\mathcal{S}^*; Y)$. By **A3**, $I(f_{\text{global}}; Y) \leq \eta I(\mathcal{S}^*; Y)$. If $\gamma \geq \eta$ the claim follows; strict when $>$.

Corollary 1 If each MPNN layer obeys SDPI with coefficient $\alpha \in (0, 1)$ relative to Y (composition yields α^L), then

$$I(f_{\text{global}}(G); Y) \leq \alpha^L I(\mathcal{S}^*; Y). \quad (6)$$

If $I(H(G); Y) \geq \gamma I(\mathcal{S}^*; Y)$ with depth-independent $\gamma > 0$, then $I(H; Y) > I(f_{\text{global}}; Y)$ for all $L \geq L_0 := \lceil \log(\gamma) / \log(\alpha) \rceil$. SDPI is multiplicative under composition of Markov kernels; apply to L layers. Compare with the lower bound for H .

Lemma 1 For deterministic $f_{\text{global}}(G)$ and $H(G)$,

$$I([f_{\text{global}}, H]; Y) = I(f_{\text{global}}; Y) + I(H; Y \mid f_{\text{global}}) \geq I(f_{\text{global}}; Y). \quad (7)$$

Chain rule and nonnegativity of conditional mutual information.

Proposition 1 Augmenting by concatenation does not decrease Euclidean pairwise distances. $\| [x \oplus u] - [y \oplus v] \|_2^2 = \|x - y\|_2^2 + \|u - v\|_2^2$.

Proposition 2 With i.i.d. α edits and flip-probabilities $p > q$ inside vs. outside \mathcal{S}^* ,

$$\Pr[\phi(G') \neq \phi(G)] - \Pr[\phi(G'') \neq \phi(G)] = (1 - q)^\alpha - (1 - p)^\alpha \geq (p - q)\alpha(1 - p)^{\alpha-1} > 0. \quad (8)$$

Independence gives $1 - (1 - p)^\alpha$ and $1 - (1 - q)^\alpha$ for flip events; subtract. Apply the mean-value theorem to $t \mapsto (1 - t)^\alpha$ on $[q, p]$.

Remarks on A1, A3 **A1** can be read as a graph-theoretic sufficiency statement: labels are determined by structures concentrated around high-centrality seed regions. **A3** abstracts the well-documented spectral low-pass/oversmoothing behavior in deep MPNNs; see Li et al. (2018); Oono & Suzuki (2019). SDPI provides a canonical way to turn such contraction into information bounds used in Cor. 1.

A.2 DETAILED EXPLANATION OF NODE CENTRALITY

For each node $v \in V$, we define three centrality measures that quantify different aspects of v 's importance in the graph:

- **Degree centrality:**

$$C_D(v) = \deg(v),$$

where $\deg(v)$ is the number of immediate neighbors of node v . High-degree nodes function as local hubs, receiving and propagating messages from a larger portion of the graph. In the context of MP networks (e.g., GNNs), such nodes tend to aggregate feature information from many neighbors, often leading to a richer local representation.

- **Betweenness centrality:**

$$C_B(v) = \sum_{s \neq t, s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the total number of shortest paths between nodes s and t , and $\sigma_{st}(v)$ is the number of those paths passing through v . Nodes with high C_B lie on many pivotal paths, allowing them to bridge otherwise distant regions of the graph. In GNNs, such bridging nodes

can facilitate long-range information flow across subgraphs, thereby encoding structurally disparate patterns crucial for downstream tasks (e.g., classification or link prediction).

- **Closeness centrality:**

$$C_C(v) = \frac{1}{\sum_{u \in V} d(u, v)},$$

where $d(u, v)$ is the shortest-path distance between nodes u and v . A node with high C_C maintains a small average distance to all other nodes, enabling efficient gathering or dissemination of signals throughout the graph. In many real-world networks, such nodes are situated in topologically central positions, thus capturing diverse local features with fewer hops compared to peripheral nodes.

Why These Nodes Encode Key Information. Each centrality measure highlights a unique structural role in the graph:

- *High-degree* (degree centrality) nodes often receive or transmit the majority of messages in a localized sub-region, thus naturally encoding dense local interactions.
- *High-betweenness* nodes serve as “gateways,” connecting different communities or components and enabling critical inter-region communication.
- *High-closeness* nodes retain short path distances to a large portion of the graph, allowing them to capture global patterns with minimal propagation steps.

By focusing on such nodes in a graph partitioning or feature aggregation algorithm, one can preserve crucial structural signals that might otherwise be smoothed out or lost in purely global embeddings.

B EXPERIMENTAL SETTING

B.1 DATASET DESCRIPTIONS

We evaluate on eight benchmark graph classification and regression datasets spanning social networks, collaboration networks, and molecular graphs. Key statistics are given where available.

IMDB-BINARY and IMDB-MULTI Yanardag & Vishwanathan (2015). These social network datasets center on movie co-appearances. Nodes represent actors and an undirected edge indicates that two actors co-starred in the same film. IMDB-BINARY contains 1,000 graphs split evenly across two classes; IMDB-MULTI contains 1,500 graphs across three classes. The goal is to distinguish genre differences, for example action versus romance.

COLLAB Yanardag & Vishwanathan (2015). This collaboration network comprises 5,000 ego-networks of computer science authors. Nodes are authors, edges connect co-authors, and the task is to classify each ego-network into one of three research fields.

MUTAG Debnath et al. (1991). A small chemical dataset of 188 graphs. Nodes are atoms of seven possible types, edges are chemical bonds. Node features encode atom type; the binary classification target is mutagenicity (toxic or non-toxic).

PTC Helma et al. (2001). Four related molecular toxicity datasets (PTC_{MR}, PTC_{FR}, PTC_{MM}, PTC_{FM}) with 344–438 graphs each. Atom types and bond types vary slightly across variants. The objective is to predict carcinogenicity or related biological activity.

NCI1/NCI109 Wale et al. (2008). These large chemical activity datasets contain 4,100 and 4,127 graphs respectively. Nodes denote atoms, edges denote bonds. Both require binary classification of compounds as active or inactive against certain cancer cell lines; node feature dimensions differ by one element between NCI1 and NCI109.

Table 7: Dataset specifications, including the number of graphs, classes, average nodes, edges, and node feature dimensionality.

NAME	# GRAPH	# CLASS	AVG. NODES	AVG. EDGES	# NODE FEAT.
IMDB-B	1,000	2	19.77	96.53	136
IMDB-M	1,500	3	13.00	65.94	89
COLLAB	5,000	3	74.49	2457.78	492
MUTAG	188	2	17.93	19.79	7
PTC_MR	344	2	14.29	14.69	18
PTC_FR	351	2	14.56	15.00	19
PTC_MM	336	2	13.97	14.32	20
PTC_FM	349	2	14.11	14.48	18
NCII	4,110	2	29.87	32.30	37
NCI109	4,127	2	29.68	32.13	38
PROTEINS	1,113	2	39.06	72.82	3
OGBG-MolHIV	41,127	1 (ROC-AUC)	25.5	27.5	9 (<i>emd_dim</i> = 100)
ZINC	249,456	1 (MAE)	23.2	49.8	1 (CONTINUOUS)

PROTEINS Borgwardt et al. (2005). A protein structure dataset of 1,113 graphs. Nodes represent amino acids, edges indicate spatial contacts. Node features capture secondary structure; labels denote protein functional classes.

OGBG-MolHIV Hu et al. (2020). A molecular property prediction dataset drawn from MoleculeNet. It comprises 41,127 graphs each representing a drug-like molecule. Nodes are atoms with a nine-dimensional RDKit featurization and edges are chemical bonds with four-dimensional bond features. The single binary task is to predict HIV replication inhibition, evaluated via ROC-AUC on the official splits.

ZINC Irwin et al. (2012); Dwivedi et al. (2020). A graph regression benchmark on a subset of 12,000 molecular graphs sampled from the ZINC15 database (full set 250,000). Nodes denote heavy atoms with categorical type features; edges denote bonds. The target is constrained solubility (log P penalized by synthetic accessibility and cycle count), measured by mean absolute error under an 80/10/10 train/validation/test split.

More detailed quantitative statics are presented in Table 7.

B.2 EXTENDED EXPERIMENTAL DETAILS

Cross-Validation. We use a 10-fold CV split for each dataset, ensuring that every fold preserves class balance as much as possible. The reported accuracy (or other metric) is the mean over all 10 folds.

Hyperparameter Search. We systematically search layer depths 1, 2, 3, 4, 5 and hidden dimensions 16, 32, 64, 128, 256, selecting the best performing combination on a validation fold (within each CV split). We then retrain on the combined training + validation data. More detailed hyperparameter settings are presented in Table 8.

Training Regimen. *Epochs:* up to 200 *Batch Size:* 4096 *Optimizer:* Adam, initial LR = 0.01, LR halved every 50 steps *Graph Partitioning:* After the GNN encoder produces node embeddings, the proposed partitioning algorithm is applied, then subgraph embeddings are fused with the global representation (see Section 4 of the main text).

Implementation. Framework: PyTorch-Geometric Fey & Lenssen (2019), Hardware: 6×A100 GPUs (80GB each)

Table 8: Best performance configurations for each method on different datasets, with specified layer (L) counts and hidden dimension (H) choices. Here, “1” corresponds to 16, “2” to 32, “3” to 64, and “4” to 128 hidden units.

	MUTAG		NCI1		NCI109		PROTEINS		SPTC _{MR}		PTC _{FR}		PTC _{MM}		PTC _{FM}		IMDB-B		BIMDB-M		COLLAB	
METHOD	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
GCN	5	4	4	5	5	2	4	4	5	2	5	4	5	4	5	4	5	4	4	4	1	3
GIN	2	1	2	3	2	3	1	5	2	4	1	3	3	1	1	2	1	3	1	2	2	1
GIN0	3	1	2	4	2	4	1	3	1	4	1	3	4	1	3	2	1	2	1	2	1	3
TopK	5	4	3	5	4	3	4	3	5	3	4	3	5	5	5	2	5	4	4	3	1	4
SAGPOOL	5	3	5	3	5	3	4	2	5	5	3	3	5	4	5	4	5	2	4	4	1	2
EDGEPOOL	5	2	5	3	5	2	5	4	4	1	3	3	4	3	4	4	1	4	1	4	1	4
GRACUS	5	2	5	3	5	3	5	4	5	2	5	4	3	5	5	4	5	4	2	4	2	3
GAT	5	4	5	4	5	3	4	4	2	2	4	4	1	4	3	1	4	4	2	4	1	4
SET2SET	5	3	4	4	3	3	2	3	5	3	4	5	1	3	5	4	4	4	1	4	2	3
SAGE	5	4	4	3	3	3	4	4	2	2	5	4	3	4	4	1	2	4	2	4	1	4

For OGBG-MolHIV, graph node features X were encoded using the molecular encoder provided on the official website, and all experiments were conducted under the same protocol as the other benchmarks.

B.3 DESCRIPTION OF BASELINE

B.3.1 GRAPH NEURAL NETWORK BASELINE

GCN (Graph Convolutional Network) Kipf & Welling (2016) Aggregates neighbor features via adjacency-based averaging and has served as a foundational model for various graph-level tasks. However, when many layers are stacked, node embeddings may become oversmoothed, losing local distinctions.

GraphSAGE (SAGE) Hamilton et al. (2017) Samples a fixed number of neighbors per node at each layer, reducing computational costs, especially in large graphs. By constraining the breadth of feature propagation, GraphSAGE offers faster training but can still be susceptible to deep-layer over-smoothing.

GAT (Graph Attention Network) Veličković et al. (2017) Applies learnable attention coefficients to each node’s neighbors, prioritizing the most relevant edges. While this aids interpretability and dynamic weighting of connections, deeper GATs can still converge to near-uniform attention distributions under certain conditions.

GIN (Graph Isomorphism Network) Xu et al. (2018a) Incorporates an $(1 + \epsilon)$ -based neighbor aggregation function, improving the capacity to distinguish graph structures. GIN typically exhibits strong expressivity but, like other multi-layer GNNs, is prone to rank-1 feature collapse in deeper configurations.

TopK Cangea et al. (2018) Selects a subset of the highest-scoring nodes at each pooling stage, effectively reducing graph size. Although it preserves top “importance” signals, smaller subgraphs may omit certain local patterns needed for accurate classification.

SAGPool Lee et al. (2019) Uses a self-attention mechanism to retain top-relevance nodes, facilitating hierarchical pooling. By focusing on salient substructures, SAGPool reduces over-representation of less-informative areas but, in very deep architectures, may still blur fine-grained details.

EdgePool Cangea et al. (2018) Contracts edges with learned importance scores to produce coarser versions of the graph. This approach can optionally revert to the original structure (unpooling),

making it suitable for node-level tasks, but edges with moderate importance might be prematurely pruned.

Graculus Dhillon et al. (2007) Clusters nodes using a kernel-based approximation to spectral cuts, avoiding explicit eigenvector computation. Although efficient for large graphs, it can merge nodes that share superficial similarity, risking the loss of critical local cues.

Set2Set Vinyals et al. (2015) Extends sequence-to-sequence models to unordered node sets, using a learnable attention-based process to gather graph-wide features. This global pooling can capture important patterns but may underemphasize localized details if the graph is extremely large or heterogeneous.

GCNII Chen et al. (2020) GCNII extends the vanilla GCN by incorporating initial residual connections and identity mapping to alleviate over-smoothing and enable stable training of much deeper architectures. However, high-degree nodes still risk representation collapse because identity mapping only slows but does not fully prevent convergence to a stationary distribution. The model also introduces two additional hyperparameters, which increase tuning complexity and computational cost.

GRAPHORMER Ying et al. (2021) GRAPHORMER decouples local message passing from global self-attention to form a modular graph transformer with linear complexity ($O(N + E)$) that approximates universal functions on graphs. It combines positional and structural encodings with a local MPNN layer and linear attention mechanisms to achieve state-of-the-art performance on 16 benchmarks. Nevertheless, the absence of a standardized encoding scheme can lead to inconsistent generalization across diverse graph types.

Union Subgraph GNNs Xu et al. (2024) Union-style SGNNs merge extracted subgraphs into a larger “union graph” to enhance expressivity (e.g., beyond 1-WL limitations) before downstream readout. This can capture richer relational patterns, but the merged structure raises memory/compute and shifts the objective toward expressivity rather than directly countering depth-induced contraction. In our experiments, SP’s partition–encode–concatenate design yielded stronger accuracy–overhead trade-offs under identical backbones.

B.3.2 ADDITIONAL METHOD

PairNorm Zhao & Akoglu (2019) PairNorm introduces a normalization layer that enforces a constant total pairwise squared distance among node embeddings across layers, preventing collapse into a single subspace and mitigating over-smoothing without altering the architecture. It significantly improves robustness for deep GCN, GAT, and SGC models. Despite this, PairNorm does not inherently enhance representational power and may deliver diminishing accuracy gains as network depth increases.

Jumping Knowledge Networks (JK) Xu et al. (2018b) JKNet adaptively aggregates representations from multiple neighborhood ranges by learning layer-wise attention weights, allowing each node to select the most relevant receptive field. Combined with various backbones, it mitigates both over-smoothing and over-squashing to improve performance on social and bioinformatics networks. The need to store and aggregate multiple intermediate embeddings incurs high memory usage and computational overhead in deep configurations.

DropEdge Rong et al. (2019) DropEdge randomly removes a fraction of edges at each training epoch, acting as data augmentation to slow down over-smoothing and reduce over-fitting. It integrates seamlessly with many GNN backbones and consistently boosts node classification accuracy. However, random edge removal can unintentionally eliminate critical connections while preserving uninformative ones, and it requires careful tuning of the drop rate to avoid performance degradation.

B.3.3 SUBGRAPH GNNs

Policy-Learn Bevilacqua et al. (2023b) These methods learn a policy (e.g., reinforcement- or gradient-based) to select informative subgraphs, then encode the chosen set with a shared subgraph

encoder before fusing with the global representation. Learning the selector improves adaptivity but introduces an extra trainable module and non-trivial training overhead (policy optimization and exploration). In our standardized GIN comparisons, policy-learned SGNNs improved over the backbone yet were consistently outperformed by SP under the same bag sizes, while also incurring higher training cost.

HyMNSouthern et al. (2025) HyMN scores nodes with walk-based centralities (via random-walk/matrix-power operations), forms many local subgraphs, and aggregates their embeddings to balance expressiveness and efficiency. It is flexible and architecture-agnostic, but processing large subgraph bags and computing walk-based scores increase preprocessing and per-epoch cost (empirically higher than SP on multiple benchmarks. In our matched GIN setting, HyMN delivered solid gains yet trailed SP in both accuracy and runtime.

MAG-GNN Kong et al. (2023) MAG-GNN casts subgraph selection as a policy-learning problem: a trainable selector (e.g., RL) chooses informative subgraphs, which are then encoded and fused with the backbone’s global representation. The learned policy offers adaptivity but adds a nontrivial module, exploration noise, and extra hyperparameters, increasing training time and variance. Under our standardized GIN setup and matched bag sizes, MAG-GNN improved over the backbone yet was consistently outperformed by SP, while incurring higher computational overhead.

CS-GNN Bar-Shalom et al. (2024) CS-GNN provides an equivariant subgraph framework via graph products and coarsening, building a structured subgraph bag and processing it with shared encoders to enhance expressiveness. This design preserves permutation symmetry but introduces additional product/coarsening steps and heavier tensor operations, which raise preprocessing and per-epoch costs. In our matched GIN comparisons, CS-GNN yielded competitive gains but trailed SP on both accuracy and runtime, whereas SP’s partition–encode–concatenate path achieved a stronger accuracy–overhead trade-off without architectural redesign.

C ADDITIONAL EXMPERIMENTS

C.1 ERROR ANALYSIS

We analyzed the few configurations where SP underperformed and identified three non-fundamental causes, each with a straightforward remedy.

(1) Overfitting on small datasets. On MUTAG with SET2SET (LSTM+attention readout), the fusion MLP adds parameters and overfits. Enabling early stopping under the same training settings recovers performance: baseline 73.7% versus SP 74.5% (+0.8 pp). A similar pattern appears on the PTC variants (approximately 300 graphs). The effect is summarized in Table 9, where early stopping consistently closes or reverses the gap.

(2) Underfitting of EdgePool on NCI1. For NCI1+EDGEPOOL, the original training length was insufficient. Increasing the number of epochs by 50% for both baseline and SP improves both models and restores SP’s advantage: baseline 73.55% versus SP 74.20%.

Table 9: **Small datasets: early stopping mitigates overfitting.** Accuracy (%) after enabling early stopping under the same settings.

Dataset–Backbone	Baseline	SP	Δ (pp)
PTC_MR–Set2Set	55.9	57.5	+1.6
PTC_FR–Graclus	65.9	66.8	+0.9
PTC_FR–GraphSAGE	67.5	69.1	+1.6
PTC_MM–GIN0	66.8	68.1	+1.3
PTC_MM–TopK	67.6	68.9	+1.3
PTC_FM–GIN0	60.4	62.5	+2.1
PTC_FM–GraphSAGE	61.2	62.0	+0.8

Table 10: **Unlabeled/social graphs: modest (κ, τ) tuning helps.** Reported as tuned SP accuracy and Δ over the corresponding baseline.

Dataset	(κ, τ)	Backbone	SP (%)	Δ (pp)
IMDB-B	(1,3)	EdgePool	74.0	+0.5
IMDB-B	(1,3)	GAT	74.5	+1.5
IMDB-B	(1,3)	Set2Set	73.5	+0.6
IMDB-M	(1,2)	GCN	51.0	+0.1
COLLAB	(4,4)	Graculus	81.6	+2.2
COLLAB	(4,4)	Set2Set	82.1	+4.5
COLLAB	(4,4)	GraphSAGE	81.9	+2.3

MUTAG (n=188, avg_nodes=17.93, naive GCN=74.0) NCI1 (n=4110, avg_nodes=29.87, naive GCN=69.2) PROTEINS (n=1113, avg_nodes=39.06, naive GCN=71.6)

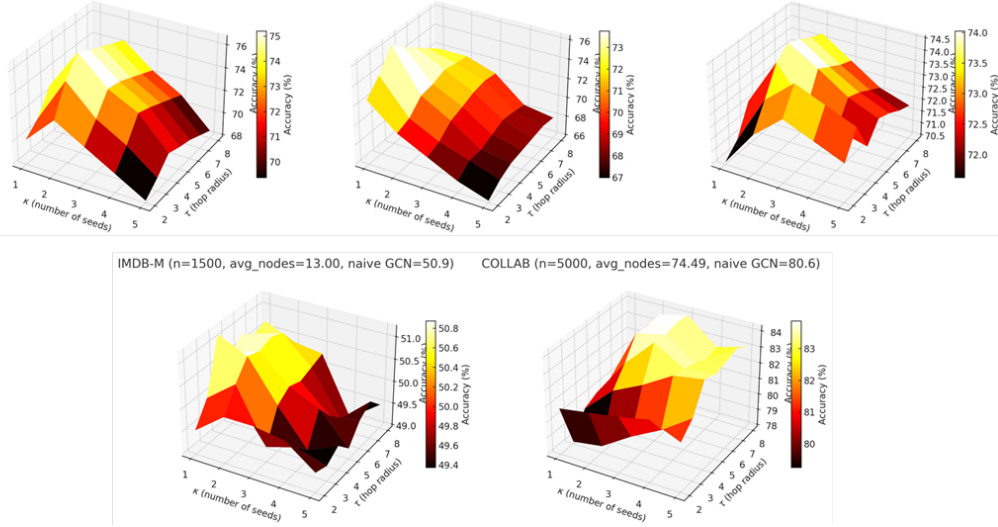


Figure 2: **Accuracy over (τ, κ) .** Peaks at $(7, 3)$ for COLLAB, $(5, 2)$ for PROTEINS, $(4, 1)$ for NCI1, $(4, 2)$ for MUTAG, and a flat surface with a mild optimum near $(4, 1)$ for IMDB-M. These patterns highlight dataset-specific receptive fields and motivate per-dataset tuning of τ and κ .

(3) Suboptimal (κ, τ) on graphs without node features. For fairness, the main paper fixed (κ, τ) globally, which is not optimal for all datasets. Light tuning by average node count per graph yields the improvements reported in Table 10.

Overall, observed drops are attributable to optimization choices (early stopping and training length) and globally fixed (κ, τ) , rather than to the SP mechanism itself. With early stopping on small datasets, sufficient epochs for EDGEPOOL, and modest per-dataset tuning of (κ, τ) , SP consistently regains and extends its advantage.

C.2 SENSITIVITY TO τ AND κ

Figure 2 visualizes the accuracy surfaces over the subgraph radius τ and the number of seeds κ , making the dataset-specific optima explicit. The surfaces are *non-monotonic* and dataset specific. On COLLAB, performance peaks at $\tau=7$, $\kappa=3$ (84.2%, +3.6 pp over a naive GCN), whereas PROTEINS peaks at $\tau=5$, $\kappa=2$ (74.5%, +2.9 pp). NCI1 attains its best at $\tau=4$, $\kappa=1$ (76.2%, +7.0 pp) and then degrades as κ increases. MUTAG peaks at $\tau=4$, $\kappa=2$ (76.6%, +2.6 pp). IMDB-M is largely flat with a mild optimum near $\tau=4$, $\kappa=1$ (51.2%, +0.3 pp).

These shifts indicate that SP exploits *dataset-dependent receptive fields*: molecular graphs favor smaller κ and moderate τ to capture localized chemistry, while social graphs benefit from larger

Table 11: **ZINC (MAE ↓)**. $T=1$ (GIN) baseline MAE: 0.103.

T	Random	Mag-GNN	OSAN	Policy-Learn	CS-GNN	HyMN	SP (ours)
2	0.110±0.005	0.111±0.010	0.195±0.008	0.090±0.008	0.107±0.014	0.089±0.009	0.089±0.004
3	0.112±0.006	0.110±0.012	0.194±0.006	0.091±0.006	0.093±0.007	0.090±0.006	0.086±0.005
5	0.113±0.012	0.107±0.008	0.188±0.007	0.089±0.008	0.089±0.015	0.086±0.003	0.080±0.006
6	0.120±0.013	0.105±0.006	0.182±0.008	0.095±0.005	0.088±0.008	0.092±0.014	0.084±0.011

Table 12: **OGBG-MolHIV (ROC-AUC ↑)**. $T=1$ (GIN) baseline AUC: 80.5.

T	Random	Mag-GNN	OSAN	Policy-Learn	CS-GNN	HyMN	SP (ours)
2	75.9±1.6	80.3±1.1	76.3±2.1	83.5±2.2	78.6±1.5	83.4±2.2	83.5±1.9
3	76.3±1.0	80.6±1.9	77.4±2.1	83.5±1.5	79.6±1.9	83.7±2.1	83.8±2.1
5	77.2±0.9	81.0±2.3	77.8±1.2	83.8±1.5	83.6±1.4	83.7±1.2	84.0±1.0
6	77.7±1.6	81.1±2.2	78.7±2.1	82.4±1.9	84.0±1.7	82.2±2.5	84.1±2.0

neighborhoods and multiple seeds to summarize multi-community structure. Practically, this supports exposing τ and κ as controllable knobs and tuning them per dataset rather than fixing a single setting. Together with the locality and contraction assumptions (Sec. 3.3), these empirical surfaces substantiate that concentrating capacity on high-centrality regions is beneficial, while the optimal spatial scale (radius and seed count) is task dependent.

C.3 HEAD-TO-HEAD COMPARISON WITH SUBGRAPH GNNs

Protocol. We compare SP to recent subgraph GNNs under a unified setup: a 6-layer GIN backbone for all methods and a controlled “bag size” $T \in \{2, 3, 5, 6\}$. For SP, T is aligned with the number of disjoint subgraphs formed per centrality (degree, betweenness, closeness):

$$T=2 : \kappa=1 \text{ (drop one subgraph at random),} \quad T=3 : \kappa=1, \\ T=5 : \kappa=2 \text{ (drop one at random),} \quad T=6 : \kappa=2.$$

All other training details (optimizer, schedule, readout) follow the shared configuration in the main text; we report mean±std over multiple runs.

Results Across four benchmarks (Tables 11–14), SP is best or tied-best at every T , and generally improves as T increases (minor non-monotonicity on ZINC within one standard deviation). The advantage tracks two design differences from other subgraph methods: (i) *deterministic, disjoint, high-centrality* selection (vs. stochastic/learned selection over large candidate pools), and (ii) *readout-time concatenation* that preserves each subgraph’s signal (vs. union-graph construction or processing many overlapping subgraphs). Practically, SP yields stronger and more stable accuracy–overhead trade-offs while keeping the base GNN unchanged.

C.4 COMPLEXITY ANALYSIS

Let n be the number of nodes, m the number of edges, d the embedding width, L the number of GNN layers, and let \mathcal{S} denote the selected subgraphs with $|\mathcal{S}| = \kappa$. Write (n_S, m_S) for the size of subgraph S and \bar{d} for the average degree.

Table 13: **COLLAB (Accuracy ↑)**. $T=1$ (GIN) baseline Acc: 78.2.

T	Random	Mag-GNN	OSAN	Policy-Learn	CS-GNN	HyMN	SP (ours)
2	67.3±2.4	74.7±1.9	67.7±2.6	78.4±2.9	75.6±2.8	78.8±1.5	79.3±1.4
3	67.9±1.2	75.0±1.0	68.7±2.8	80.3±1.9	76.1±2.0	78.7±1.7	80.4±2.4
5	68.5±1.8	75.8±2.7	68.9±2.6	78.9±1.7	77.0±1.5	79.5±1.3	82.3±1.5
6	69.2±2.8	76.5±2.8	70.1±2.5	77.6±1.9	77.8±1.6	77.9±1.2	83.1±3.0

Table 14: **NC11 (Accuracy \uparrow).** $T=1$ (GIN) baseline Acc: 72.8.

T	Random	Mag-GNN	OSAN	Policy-Learn	CS-GNN	HyMN	SP (ours)
2	65.4 \pm 3.1	74.4 \pm 2.8	66.3 \pm 1.0	76.2 \pm 3.5	74.3 \pm 1.2	76.5 \pm 2.9	76.8\pm2.8
3	66.1 \pm 3.0	74.7 \pm 1.5	67.1 \pm 2.8	75.6 \pm 2.5	75.2 \pm 3.3	77.1 \pm 1.2	77.1\pm3.5
5	66.9 \pm 1.6	75.1 \pm 2.3	67.5 \pm 2.8	75.0 \pm 2.0	75.3 \pm 1.0	76.1 \pm 1.6	77.0\pm1.2
6	67.4 \pm 2.8	75.6 \pm 1.6	68.3 \pm 3.1	75.2 \pm 3.2	75.8 \pm 1.8	75.7 \pm 2.8	76.9\pm2.3

Preprocessing (one-time).

Degree centrality: $\mathcal{O}(n+m)$,

Betweenness centrality (Brandes, unweighted): $\mathcal{O}(nm)$ (weighted: $\mathcal{O}(nm+n^2 \log n)$),

τ -hop extraction around κ seeds: $\mathcal{O}\left(\sum_{S \in \mathcal{S}} (n_S + m_S)\right) \subseteq \mathcal{O}(\min\{\kappa(n+m), \kappa \bar{d}^\tau\})$.

These costs are depth-agnostic and can be cached per graph.

Per-epoch training/inference. For a message-passing encoder f , a standard sparse bound is

$$\text{cost}(f, G) = \mathcal{O}(L(m+n)d).$$

With SP, one epoch processes the full graph and all selected subgraphs:

$$\text{cost}_{\text{epoch}} = \text{cost}(f, G) + \sum_{S \in \mathcal{S}} \text{cost}(f, G[S]) = \mathcal{O}\left(Ld[(m+n) + \sum_S (m_S + n_S)]\right).$$

Hence the relative overhead factor

$$\rho := \frac{\sum_S (m_S + n_S)}{m+n}$$

is approximately *depth-invariant*: adding layers multiplies both terms by the same L . In practice, bounded τ and small κ (e.g., $1 \sim 5$) keep ρ modest.

Fusion (readout-time). Concatenation is $\mathcal{O}(|\mathcal{S}|d_s)$; an MLP with hidden width h over the fused vector of size $(d_b + |\mathcal{S}|d_s)$ costs

$$\mathcal{O}((d_b + |\mathcal{S}|d_s)h + hc),$$

typically negligible relative to message passing.

SP’s *one-time* preprocessing is independent of L ; the *per-epoch* cost increases linearly with the total subgraph volume $\sum_S (m_S + n_S)$ and remains roughly constant in relative terms as depth grows.

C.5 COMPUTATIONAL COMPLEXITY AND RUNTIME OVERHEAD

Preprocessing cost (one time). Table 15 reports the wall-clock time required to compute centralities and extract τ -hop subgraphs. SP adds only a small overhead on small node-classification graphs (Cora, CiteSeer), but on a large graph like PubMed the cost grows with $|V|$. For graph-classification benchmarks, the overhead is negligible on MUTAG and rises for DD and COLLAB. This cost is incurred once per dataset and centrality scores are reusable across (κ, τ) sweeps.

Asymptotics and method-level comparison (preprocessing). Table 16 summarizes preprocessing complexity and memory characteristics for SP and representative SGNN families. Empirically, all compared pipelines exhibit approximately cubic scaling in $|V|$ due to global scoring or clustering.

Observed extraction times (normalized to SP) in Table 17 further show SP’s practical advantage as graphs grow.

Per-epoch training overhead. Beyond preprocessing, SP adds a small, depth-independent cost from subgraph passes computed in parallel with the base pass. Table 18 lists representative measurements.

Table 15: One-time preprocessing time (seconds). $|\mathcal{G}|$: number of graphs, $\overline{|V|}$: average nodes per graph. “No-SP” includes dataset loading and any baseline preprocessing.

DATASET	$ \mathcal{G} $	$\overline{ V }$	SP (s)	No-SP (s)
CORA (NODE CLASSIFICATION)	1	2,708	25.67	5.87
CITESEER (NODE CLASSIFICATION)	1	3,327	26.02	6.24
PUBMED (NODE CLASSIFICATION)	1	19,717	1793.61	7.80
MUTAG (GRAPH CLASSIFICATION)	188	17.9	2.70	2.22
DD (GRAPH CLASSIFICATION)	1,178	284.3	523.86	8.05
COLLAB (GRAPH CLASSIFICATION)	5,000	74.5	705.38	86.29
OGBG-MOLHIV	41,127	25.5	113.30	119.07
ZINC	220,011	23.15	701.62	49.85

Table 16: Preprocessing characteristics by method (informal summary).

Method	Preproc. Complexity	Core Ops	Extra Memory
SP (ours)	$O(n^3)$	τ -hop extraction + pooling	baseline size
Eff. SubGNN	$O(n^3)$	spectral clustering + APSP	$n_{\text{cluster}} \times$ baseline
HyMN	$O(k n^3)$	k -step walk + matrix power	baseline + masks

Table 17: Normalized extraction time (SP = $1.0 \times$). Node/edge counts shown in parentheses.

Dataset	SP	Product Graph	HyMN
ZINC12k (23/50)	$1.0 \times$	$1.9 \times$	$2.2 \times$
NCI1 (30/32)	$1.0 \times$	$2.7 \times$	$3.4 \times$
COLLAB (74/2458)	$1.0 \times$	$6.8 \times$	$15.6 \times$
REDDIT (429/497)	$1.0 \times$	$30.5 \times$	$60.8 \times$

Table 18: Per-epoch overhead (representative runs).

Dataset	Wall-clock Δ	Peak GPU mem. Δ
ZINC	$\approx +11\%$	$+5\text{--}10\%$
MolHIV	$\approx +8\%$	$+5\text{--}10\%$

In terms of operations, messages increase by only 20–30% because $\kappa \ll n$ and subgraphs are small. On large graphs (COLLAB), each +1 in κ adds about +2.7% runtime; on small graphs (MUTAG), about +0.07%. Memory overhead from storing subgraph tensors remained modest (about +1.3% on COLLAB when varying $\kappa \times \tau$). Preprocessing is executed once and amortized over repeated training and tuning.

Implementation notes. We batch subgraphs with the full-graph pass; on multi-GPU, subgraphs can run concurrently with the main pass, hiding much of the additional latency.

D CODE

The code is available at <https://anonymous.4open.science/r/SP-3877/>