Translating Subgraphs to Nodes Makes Simple GNNs Strong and Efficient for Subgraph Representation Learning

Dongkwan Kim [1] **Alice Oh** [1]

Abstract

Subgraph representation learning has emerged as an important problem, but it is by default approached with specialized graph neural networks on a large global graph. These models demand extensive memory and computational resources but challenge modeling hierarchical structures of subgraphs. In this paper, we propose Subgraph-To-Node (S2N) translation, a novel formulation for learning representations of subgraphs. Specifically, given a set of subgraphs in the global graph, we construct a new graph by coarsely transforming subgraphs into nodes. Demonstrating both theoretical and empirical evidence, S2N not only significantly reduces memory and computational costs compared to state-of-the-art models but also outperforms them by capturing both local and global structures of the subgraph. By leveraging graph coarsening methods, our method outperforms baselines even in a data-scarce setting with insufficient subgraphs. Our experiments on eight benchmarks demonstrate that fined-tuned models with S2N translation can process 183 – 711 times more subgraph samples than state-of-the-art models at a better or similar performance level.

1. Introduction

Subgraph representation learning has been shown to be useful for various real-world problems (Bordes et al., 2014; Luo, 2022; Hamidi Rad et al., 2022; Maheshwari et al., 2024). Current research uses the default data structures for graph-level tasks, treating the subgraph as just a subset of the global graph. Existing studies on subgraph representation learning focus on developing graph neural networks specialized for subgraphs (Alsentzer et al., 2020; Wang & Zhang, 2022). However, specialized models suffer from large memory and computational requirements from a large global graph. Their complex operations also lead to limitations in learning both local and global interactions of subgraphs. Here, we address a fundamental yet underexplored question in subgraph representation learning before model design: **How can we efficiently and effectively process subgraphs as data for representation learning?**

In this paper, we propose 'Subgraph-To-Node (S2N)', a novel data structure to solve subgraph-level prediction tasks efficiently. This data structure is a new graph translated from the original global graph and subgraphs, where its nodes are the original subgraphs, and its edges are the relations among the original subgraphs. Then, we can get the results of the subgraph-level tasks by performing node-level tasks from these node representations.

For example, Alsentzer et al. (2020) introduces a fitness social network where subgraphs are users, nodes are workouts, and edges indicate if multiple users complete workouts. By using S2N translation, a new graph is created; users become nodes, and edges express relations between them. This graph directly shows the relationships between users, following the conventional approach of describing social networks where nodes are users. As seen in this example, S2N can provide a more precise description of real-world problems than a form of subgraphs.

The S2N translation enables efficient subgraph representation learning. The number of nodes in the S2N graph is decreased to the number of original subgraphs. The edges of the S2N graph are also significantly reduced, which we theoretically prove and empirically confirm in real-world datasets. We can load large batches of subgraphs on the GPU memory and parallelize the training and inference. Since S2N translation does not interfere with model selection, even simple GNNs without complex operations can encode node representations in the S2N graph.

There can be various implementations of S2N translation, and here, we create new edges as the number of shared edges across a pair of subgraphs. Then, we normalize and sparsify edges based on weights to approximate the structure of the global graph. This process makes a coarse S2N graph carry

[1]School of Computing, KAIST, South Korea. Correspondence to: Dongkwan Kim <dongkwan.kim@kaist.ac.kr>, Alice Oh <alice.oh@kaist.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

sufficient information across subgraphs for the task. We can additionally preserve structural information during S2N translation using two strategies. First, we propose S2N that retains internal structures in subgraphs. Second, we incorporate structural encoding into input node features (Dwivedi et al., 2022). These enhancements require negligible resources, as they are performed only once before training and are efficiently performed with low complexity of computation and space.

Furthermore, we address S2N's challenge when there are not sufficient samples available, specifically, representing parts of the global graph not covered by existing subgraphs. We introduce Coarsened S2N (CoS2N), which uses graph coarsening to create 'virtual' subgraphs that summarize the global structure. The CoS2N allows message-passing between distant subgraphs with labels without compromising efficiency. We also theoretically show that CoS2N can reduce the approximation error in S2N's representations.

We conduct experiments with four real-world and four synthetic datasets to evaluate the classification performance as well as efficiency, including throughput, latency, parameters, and memory usage. We demonstrate that models using S2N transformation are more efficient than existing approaches, with similar or even better performance. Specifically, while best-tuned models with S2N can process 183 – 711 times as many samples, their classification performance shows 99.9 – 102.9% of the state-of-the-art model.

The rest of the paper is organized as follows. First, we present a Subgraph-To-Node (S2N) translation, a novel way to generate an efficient data structure for subgraph representation learning (§3). This section includes Coarsened S2N (CoS2N), the combination with graph coarsening to tackle a data-scarce setting. Second, we theoretically show that S2N reduces the computational complexity and approximates subgraph representations from the original global graph (§4). Third, we demonstrate the efficiency of S2N compared to the state-of-the-art approaches, specifically enabling up to 711 times the throughput while maintaining the performance of at least 99.9% (§5, §6).

2. Related Work

Our S2N translation tackles representation learning of subgraphs, and this is closely linked to graph coarsening. We introduce these fields and their connection with our study. We discuss more related work in Appendix B.

Subgraph Representation Learning Learning subgraph representations has been beneficial across various real-world problems. Researchers model higher-order interactions by subgraphs that nodes, edges, and graphs cannot. For example, diseases and patients in gene networks (Luo, 2022), teams in collaboration networks (Hamidi Rad et al., 2022), and communities in mobile game user networks (Zhang et al., 2023) are represented by subgraphs. However, they are specialized for each domain (Zhang et al., 2023; Li et al., 2023; Trümper et al., 2023; Ouyang et al., 2024; Maheshwari et al., 2024) or have strong assumptions about the subgraph (Meng et al., 2018; Hamidi Rad et al., 2022; Kim et al., 2022; Luo, 2022), making them difficult to generalize. The Subgraph Neural Network (SubGNN) (Alsentzer et al., 2020) is the first general approach to subgraph representation learning using topology, positions, and connectivity. The GNN with LAbeling trickS for Subgraph (GLASS) (Wang & Zhang, 2022) uses a labeling trick to distinguish nodes inside and outside the subgraph and enhance the expressive power of representations. However, both SubGNN and GLASS perform complex operations on a large global graph, demanding high memory and computation but not reflecting the layered structures of subgraphs. Our method allows efficient learning of subgraph representations without a complex model design.

Graph Coarsening Our S2N translation summarizes subgraphs into nodes, and in that sense, it is related to graph coarsening (or summarization) methods (Loukas & Vandergheynst, 2018; Loukas, 2019; Deng et al., 2020; Cai et al., 2021; Huang et al., 2021; Zhou et al., 2021; Jin et al., 2022). These methods are similar to our work that aims to handle large-scale graphs efficiently, but they have not been applied to subgraph-level tasks. Moreover, the super-nodes in coarse graphs are not given to existing graph coarsening methods; thus, algorithms to decide on super-nodes are required. In S2N translation, we treat subgraphs as super-nodes and can create coarse graphs with nominal costs.

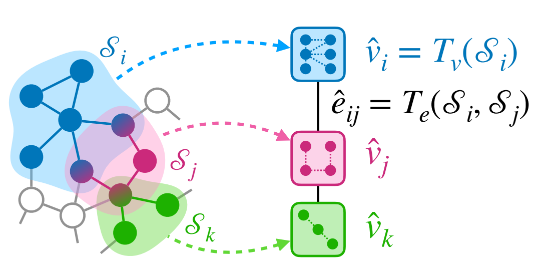
3. Data Structures for Subgraph Representation Learning

We introduce three data structures for subgraph representation learning including our proposed Subgraph-To-Node (S2N) translation.

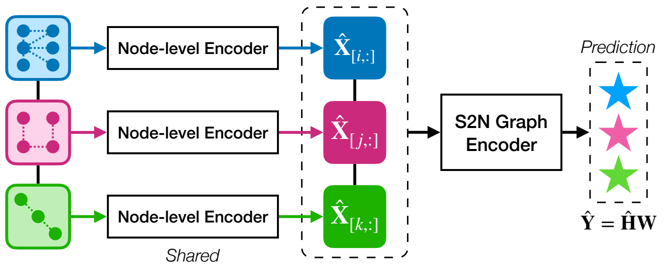
Notations We first summarize the notations in the subgraph representation learning for classification. Let $\mathcal{G} = (\mathbb{V}, \boldsymbol{A}, \boldsymbol{X})$ be a global graph where $\mathbb{V}$ is a set of nodes ($|\mathbb{V}| = N$), $\boldsymbol{A} \in \{0, 1\}^{N \times N}$ is an adjacency matrix, and $\boldsymbol{X} \in \mathbb{R}^{N \times F_0}$ is a node feature matrix. A subgraph $\mathcal{S} = (\mathbb{V}^{\text{sub}}, \boldsymbol{A}^{\text{sub}})$ is a graph formed by subsets of nodes and edges in the global graph $\mathcal{G}$. For the subgraph classification task, there is a set of $M (< N)$ subgraphs $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_M\}$, and for $\mathcal{S}_i = (\mathbb{V}_i^{\text{sub}}, \boldsymbol{A}_i^{\text{sub}})$, the goal is to learn its representation $\boldsymbol{h}_i \in \mathbb{R}^F$ and the logit vector $\boldsymbol{y}_i \in \mathbb{R}^C$ where F and C are the numbers of hidden features and classes, respectively.

3.1. Conventional Data Structures

The existing GNN-based approach employs two types of data structures when solving subgraph-level tasks. This pa-


$\mathcal{S}_i$
$\hat{v}_i = T_v(\mathcal{S}_i)$
$\hat{e}_{ij} = T_e(\mathcal{S}_i, \mathcal{S}_j)$
$\mathcal{S}_j$
$\hat{v}_j$
$\mathcal{S}_k$
$\hat{v}_k$

(a) The S2N translation. Subgraphs $\mathcal{S}_i$ and $\mathcal{S}_j$ are transformed into nodes $\hat{v}_i$ and $\hat{v}_j$ by T_v, and an edge $\hat{e}_{ij}$ between them is formed by T_e.


Node-level Encoder
Node-level Encoder
Node-level Encoder
Shared
$\hat{\mathbf{X}}_{[i,:]}$
$\hat{\mathbf{X}}_{[j,:]}$
$\hat{\mathbf{X}}_{[k,:]}$
S2N Graph Encoder
Prediction
$\hat{\mathbf{Y}} = \hat{\mathbf{H}}\mathbf{W}$

(b) Models for graphs translated by S2N. We apply a node-level encoder first (weighted sum for S2N+0 and GNN plus readout for S2N+A), then an S2N graph encoder (GNN) to their outputs for the prediction.

Figure 1: Overview of the Subgraph-To-Node (S2N) translation and models for translated graphs.

per refers to these two as Separated and Connected forms. The *Separated* form treats each subgraph as a separate graph, applying the GNN instance-wise for each graph. Existing studies express these separated graphs as *standalone* or *segregated* graphs and use this separated form as the main baseline. The *Connected* form represents subgraphs by applying the GNN on the whole global graph and pooling node representations. The separated form preserves only internal structures, and the connected form retains all information in the global graph. For this reason, using the connected form requires more memory and computational resources. Since incorporating the structures in the global graph is essential in learning subgraphs, we design a new data structure that can approximate the global graph without significant costs.

3.2. Subgraph-To-Node (S2N) Translation

The S2N translation reduces memory and computational costs by constructing a new coarse graph that summarizes the original subgraph into a node. As in Figure 1a, for each subgraph $\mathcal{S}_i \in \mathbb{S}$ in the global graph $\mathcal{G}$, we create a node $\hat{v}_i = T_v(\mathcal{S}_i)$ in the translated graph $\hat{\mathcal{G}}$; for all pairs $(\mathcal{S}_i, \mathcal{S}_j)$ of subgraphs in $\mathcal{G}$, we assign an edge $\hat{e}_{ij} = T_e(\mathcal{S}_i, \mathcal{S}_j)$ between corresponding nodes in $\hat{\mathcal{G}}$. Here, T_v and T_e are translation functions for nodes and edges in $\hat{\mathcal{G}}$, respectively. Formally, the S2N translated graph $\hat{\mathcal{G}} = (\hat{\mathbb{V}}, \hat{\boldsymbol{A}})$ where $|\hat{\mathbb{V}}| = M$ and $\hat{\boldsymbol{A}} \in \mathbb{R}^{M \times M}$, is defined by

$$\hat{\mathbb{V}} = \{T_v(\mathcal{S}_i) \mid \mathcal{S}_i \in \mathbb{S}\}, \; \hat{\boldsymbol{A}}_{[i,j]} = T_e(\mathcal{S}_i, \mathcal{S}_j). \quad (1)$$

We can choose any function for T_v and T_e. For example, T_e can be simple heuristics (e.g., the distance between subgraphs) or modeled with neural networks to learn the graph structure (Franceschi et al., 2019; Kim & Oh, 2021; Fatemi et al., 2021).

In this paper, we choose two versions of S2N functions with negligible costs: **S2N+0** and **S2N+A**. For both versions, we use the same T_e to make an edge and its weight as the number of edges between two subgraphs $\mathcal{S}_i$ and $\mathcal{S}_j$, which is defined as follows:

$$T_e(\mathcal{S}_i, \mathcal{S}_j) = \textstyle\sum_{v_i \in \mathbb{V}_i^{\text{sub}}} \sum_{v_j \in \mathbb{V}_j^{\text{sub}}} \boldsymbol{A}_{[v_i, v_j]}. \quad (2)$$

When using edge weights as input, if the range of the values is too wide, learning may be unstable. So, we normalize and clamp the edge weights to between 0 to 1 by selecting edges in a specific range of standard scores ($a - b$ where a, b are hyperparameters).

$$\text{normalize}(\hat{\boldsymbol{A}}) = \text{clamp}\left(\frac{(\hat{\boldsymbol{A}} - \text{mean}(\hat{\boldsymbol{A}}))/\text{std}(\hat{\boldsymbol{A}}) - a}{b - a}\right) \quad (3)$$
$$\text{where } \; \text{clamp}(x) = \max(0, \min(1, x)).$$

For T_v, we use different functions for S2N+0 and S2N+A. The difference between the two is whether it maintains the internal structures $\boldsymbol{A}_i^{\text{sub}}$ of the subgraph $\mathcal{S}_i = (\mathbb{V}_i^{\text{sub}}, \boldsymbol{A}_i^{\text{sub}})$. S2N+0 uses T_v that ignores $\boldsymbol{A}_i^{\text{sub}}$ and treats the node as a set (i.e., $\mathbb{V}_i^{\text{sub}}$). In contrast, S2N+A's T_v retains all information of nodes and edges in the subgraph:

$$\textbf{S2N+0: } T_v(\mathcal{S}_i) = \mathbb{V}_i^{\text{sub}}, \; \textbf{S2N+A: } T_v(\mathcal{S}_i) = (\mathbb{V}_i^{\text{sub}}, \boldsymbol{A}_i^{\text{sub}}). \quad (4)$$

Note that their names originated from whether the adjacency matrix is a zero matrix (0) or not (A).

We can enhance the input features $\boldsymbol{X}$ by incorporating structural encoding, thereby preserving more information when S2N summarizes global structures. In this paper, we adopt Random Walk Positional Encoding (RWPE) (Dwivedi et al., 2022) that encodes the k-hop topology of the global graph for each node. The efficiency of S2N is maintained since RWPE is computed once before training and only requires the space complexity of $O(N)$.

3.3. Models for S2N Translated Graphs

We propose simple but strong models for S2N (Figure 1b): node-level encoder $\texttt{ENC}_{\text{node}}$ and S2N graph encoder $\texttt{ENC}_{\texttt{S2N}}$. First, $\texttt{ENC}_{\text{node}}$ takes $T_v(\mathcal{S})$ as an input and produces $\hat{\boldsymbol{x}}_i \in \mathbb{R}^F$, input vector for the node in the S2N graph. Then, $\texttt{ENC}_{\texttt{S2N}}$ takes $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_M]^\top \in \mathbb{R}^{M \times F}$ and $\hat{\boldsymbol{A}}$ as inputs, and produces representations $\hat{\boldsymbol{H}} = [\hat{\boldsymbol{h}}_1, ..., \hat{\boldsymbol{h}}_M]^\top \in \mathbb{R}^{M \times F}$ and logits $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_M]^\top \in \mathbb{R}^{M \times C}$.

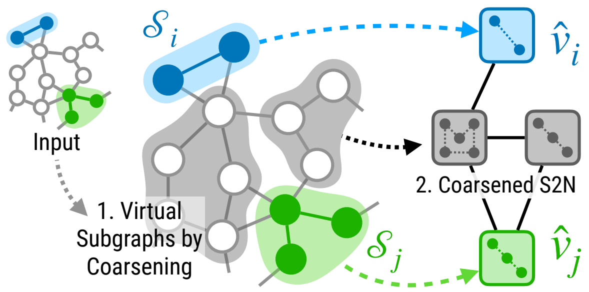


Figure 2: Overview of Coarsened Subgraph-To-Node (CoS2N) Translation with virtual subgraphs generated by graph coarsening.

For $\texttt{ENC}_{\text{node}}$, we use different models for S2N+0 and S2N+A. Since the node in S2N+0 is a set of original nodes in $\mathcal{S}_i$, we take a set of node features in $\mathbb{V}_i$ as an input and generate a weighted sum of them. For S2N+A, we apply a GNN model to each subgraph as an individual graph, then apply a weighted sum for readout. Formally,

$$\begin{aligned}&\textbf{S2N+0: } \hat{\boldsymbol{x}}_i = \textstyle\sum_{v \in \mathbb{V}_i^{\text{sub}}} \omega_{vi} \cdot \boldsymbol{X}_{[v,:]}, \\ &\textbf{S2N+A: } \hat{\boldsymbol{x}}_i = \textstyle\sum_{v \in \mathbb{V}_i^{\text{sub}}} \omega_{vi} \cdot \text{GNN}_{\text{node}}(\boldsymbol{X}_{[\mathbb{V}_i^{\text{sub}},:]}, \boldsymbol{A}_i^{\text{sub}})_{[v,:]},\end{aligned} \tag{5}$$

where ω_{vi} is a weight corresponding to the node v and the subgraph $\mathcal{S}_i$. These weights can be either learnable or constants (e.g., $\omega = 1$ means that $\hat{\boldsymbol{x}}$ is the sum of features).

Given $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{X}}$ of S2N+0 and S2N+A, we apply the S2N graph encoder $\texttt{ENC}_{\text{S2N}}$ which is another $\texttt{GNN}_{\text{S2N}}$ to generate the final node representations $\hat{\boldsymbol{H}}$ and logits $\hat{\boldsymbol{Y}}$ for prediction, That is, $\hat{\boldsymbol{H}} = \texttt{GNN}_{\text{S2N}}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{A}})$ and $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{H}}\boldsymbol{W}$ where $\boldsymbol{W} \in \mathbb{R}^{F \times C}$ is a matrix of parameters. We can take any GNNs that perform message-passing between nodes. This node-level message-passing on translated graphs is analogous to message-passing at the subgraph level in SubGNN (Alsentzer et al., 2020).

3.4. Coarsened S2N for a Data-Scarce Setting

By design, the S2N graph $\hat{\mathcal{G}}$ can approximate the global graph $\mathcal{G}$ covered by subgraphs, but cannot reflect parts of $\mathcal{G}$ where subgraphs do not exist. When a pair of subgraphs is distant on the global graph, they exist as unconnected nodes in S2N graphs as illustrated in Figure 2. These isolated subgraphs are likely to occur when the subgraph samples are scarce. In this case, GNNs cannot exchange supervised signals between subgraphs.

To solve this problem, we apply graph coarsening methods to the global graph $\mathcal{G}$ to generate a partition of nodes in $\mathcal{G}$. That is, graph coarsening summarizes a graph $\mathcal{G}$ by grouping its nodes into super-nodes. Each node in $\mathcal{G}$ corresponds to one super-node in the new graph $\hat{\mathcal{G}}$. We construct induced subgraphs $\mathbb{S}^{\text{co}} = \{\mathcal{S}_1^{\text{co}}, \mathcal{S}_2^{\text{co}}, ..., \mathcal{S}_{M^{\text{co}}}^{\text{co}}\}$ for each super-node in the global graph. Here, we call them 'virtual subgraphs.' Using the original (labeled) subgraphs $\mathbb{S}$ as is, the virtual subgraphs are merged with $\mathbb{S}$ to form the Coarsened S2N (CoS2N) graph, formally,

$$\begin{aligned}\mathbb{S}^{\text{co}} = \text{Coarsening}(\mathcal{G}), \quad \hat{\boldsymbol{A}}^{\text{co}}_{[i,j]} = T_e(\mathcal{S}_i, \mathcal{S}_j) \\ \text{where } (\mathcal{S}_i, \mathcal{S}_j) \in (\mathbb{S} \cup \mathbb{S}^{\text{co}}) \times (\mathbb{S} \cup \mathbb{S}^{\text{co}}).\end{aligned} \tag{6}$$

Here, any algorithm that coarsens the graph can be used for Coarsening($\mathcal{G}$) (See §2).

Training of CoS2N is done similarly to semi-supervised node classification. The virtual (unlabeled) subgraphs act as bridges to pass messages between labeled subgraphs. These allow S2N to better approximate the global graph that the existing set of subgraphs does not cover. We also show that adding virtual subgraphs to S2N can reduce the approximation error between representations of S2N and the global graph (Proposition 4.3).

The graph coarsening does not impair the efficiency for two reasons. First, it is performed only once before the training. Second, we can create a small CoS2N graph by tuning coarsening methods and their hyperparameters (e.g., the coarsening ratio).

4. Theoretical Analysis on S2N

This section analytically compares the efficiency and the representation quality between S2N and the original graph. We first show how much S2N reduces computational complexity. Then, we analyze the error bound of representations between S2N and the original graph. All proofs are provided in Appendix D.

4.1. How Much Does S2N Reduce Computational Complexity?

We introduce more notations for this analysis. For the global graph $\mathcal{G}$ and the S2N graph $\hat{\mathcal{G}}$, the numbers of edges are E and $\hat{E}$. Across a set $\mathbb{S}$ of subgraphs, the average numbers of nodes and edges are $\overline{N^{\text{sub}}}$ and $\overline{E^{\text{sub}}}$. Note that N is the number of nodes in $\mathcal{G}$ and M is the number of subgraphs.

In Proposition 4.1, we compare the time complexity of single-layer GLASS (the state-of-the-art model) (Wang & Zhang, 2022), Connected form, S2N+0, and S2N+A.

Proposition 4.1. *The time complexity of the 1-layer GLASS, Connected form, S2N+0, and S2N+A is*

$$\textit{GLASS \& Connected: } O(EF + M\overline{N^{sub}}F + NF^2), \tag{7}$$

$$\textit{S2N+0: } O(\hat{E}F + M\overline{N^{sub}}F + MF^2), \tag{8}$$

$$\textit{S2N+A: } O(\hat{E}F + M\overline{E^{sub}}F + M\overline{N^{sub}}F^2). \tag{9}$$

Considering that $N \ll E$ in real-world graphs (Chung, 2010), the significant difference between baselines and S2N is that E becomes $\hat{E}$. We can know that $\hat{E}$ cannot be higher than M^2. The smaller $\overline{N^{\text{sub}}}$, the smaller $\hat{E}$ since it lowers the number of possible connections between nodes in subgraphs. However, it is difficult to directly compare $\hat{E}$ and E without assumptions of the global graph and subgraphs.

Thus, we investigate what $\hat{E}$ takes in the random graph model as a global graph, specifically, the Configuration Model (CM). The CM graph of N nodes is randomly generated from a given degree sequence $[d_1, d_2, ..., d_N]$ (Newman, 2018). Note that CM can model graphs of arbitrary degree distributions, so its assumptions are mild compared to other models (See Appendix C for details). For the CM global graph with independent and identically distributed (i.i.d.) subgraphs, we analyze the distribution of S2N's edge weights (i.e., $\hat{\boldsymbol{A}}$) and demonstrate the probability that an edge weight is higher than a certain value. This probability is proportional to the number of S2N edges after normalization (Equation 3). The smaller this is, the smaller $\hat{E}$.

Proposition 4.2. *For Configuration Model as $\mathcal{G}$ and i.i.d. sampled subgraphs where the average size is $\overline{N^{sub}}$, the probability that the weight $\hat{\boldsymbol{A}}_{[i,j]}$ of an edge (i,j) in $\hat{\mathcal{G}}$ is bigger than $c > 0$ is $P(\hat{\boldsymbol{A}}_{[i,j]} \geq c) \leq \frac{(\overline{N^{sub}})^2 \mathbb{E}[d]}{cN}$ where $\mathbb{E}[d]$ is an average degree of $\mathcal{G}$.*

It is well-known that degrees follow a power-law distribution in many real-world graphs (Barabási & Albert, 1999). Most nodes have a low degree; thus, the average degree $\mathbb{E}[d]$ has a small value. Proposition 4.2 implies that edges with small weights are more likely to appear in S2N, and the edge normalization can make the S2N graph sparse (i.e., small $\hat{E}$). We also empirically confirm that edges in S2N are fewer than those of the global graph in Table 1.

4.2. How Does S2N Approximate Subgraph Representations?

For this subsection, we define the mapping matrix $\boldsymbol{M} \in \{0,1\}^{N \times M}$, where $\boldsymbol{M}_{[v,i]}$ is 1 if and only if the node v belongs to the subgraph $\mathcal{S}_i$ (i.e., $\hat{\boldsymbol{A}} = \boldsymbol{M}^\top \boldsymbol{A} \boldsymbol{M}$). Degree matrices of $\mathcal{G}$ and $\hat{\mathcal{G}}$ are $\boldsymbol{D} = \text{diag}(d_1, d_2, ..., d_N)$ and $\hat{\boldsymbol{D}} = \text{diag}(\hat{d}_1, \hat{d}_2, ..., \hat{d}_M)$. Also, $\|\cdot\|$ is the Frobenius norm.

This analysis aims to analytically compare node representations $\hat{\boldsymbol{H}} \in \mathbb{R}^{M \times F}$ of the S2N graph $\hat{\mathcal{G}}$ and subgraph representations of the global graph $\mathcal{G}$. Since outputs of GNN with the global graph are original nodes' representations $\boldsymbol{H} \in \mathbb{R}^{N \times F}$, we apply the readout to pool nodes in the subgraph where $\boldsymbol{R} \in \mathbb{R}^{N \times M}$ is a readout matrix:

$$\texttt{READOUT}(\boldsymbol{H}) = \boldsymbol{R}^\top \boldsymbol{H} \in \mathbb{R}^{M \times F}. \tag{10}$$

In this paper, we adopt a degree-dependent readout matrix $\boldsymbol{R}$ inspired by configuration-based reconstruction (Zhou et al., 2021; 2023), which is defined as follows:

$$\boldsymbol{R} = \boldsymbol{D}^{\frac{1}{2}} \boldsymbol{M} \hat{\boldsymbol{D}}^{-\frac{1}{2}} \quad \text{i.e.,} \quad \boldsymbol{R}_{[v,i]} = (d_v / \hat{d}_i)^{\frac{1}{2}}. \tag{11}$$

We now demonstrate that the S2N's node representations $\hat{\boldsymbol{H}}$ approximate the global graph's subgraph representations $\boldsymbol{R}^\top \boldsymbol{H}$, particularly when the model is a single-layer GCN. The error bound between $\hat{\boldsymbol{H}}$ and $\boldsymbol{R}^\top \boldsymbol{H}$ is introduced in Proposition 4.3. We also conduct a similar analysis for Graph Isomorphism Networks (Xu et al., 2019) in Corollary D.6 (Appendix D).

Proposition 4.3. *Using the single-layer GCN parametrized by $\boldsymbol{W}$, subgraph representations $\boldsymbol{R}^\top \boldsymbol{H}$ of the global graph $\mathcal{G}$ can be approximated by node representations $\hat{\boldsymbol{H}}$ of the S2N graph $\hat{\mathcal{G}}$, that is, $\hat{\boldsymbol{H}} \approx \boldsymbol{R}^\top \boldsymbol{H}$. The error between two representations is bounded by:*

$$\|\boldsymbol{R}^\top \boldsymbol{H} - \hat{\boldsymbol{H}}\| \leq M^{\frac{1}{2}} \|\boldsymbol{X} - \boldsymbol{R}\hat{\boldsymbol{X}}\| \cdot \|\boldsymbol{W}\|. \tag{12}$$

The error between representations is bounded by the error between input features $\boldsymbol{X}$ and $\boldsymbol{R}\hat{\boldsymbol{X}}$. As in Zhou et al. (2023), if we use the initial features $\hat{\boldsymbol{X}} = \boldsymbol{R}^\top \boldsymbol{X}$ for S2N, $\boldsymbol{R}\hat{\boldsymbol{X}}$ is $(\boldsymbol{R}\boldsymbol{R}^\top)\boldsymbol{X}$. The matrix $\boldsymbol{R}\boldsymbol{R}^\top \in \mathbb{R}^{N \times N}$ has rank $M (< N)$, then $\boldsymbol{R}\hat{\boldsymbol{X}}$ is a low-rank approximation of $\boldsymbol{X}$. Since $\boldsymbol{R}$ is given by subgraphs, $\boldsymbol{R}\hat{\boldsymbol{X}}$ may not sufficiently approximate $\boldsymbol{X}$ for the downstream task. In particular, when there are only a few subgraph samples (i.e., very small rank M), the expressiveness of S2N can be weakened. This theoretical observation implies that the proposed CoS2N (§3.4) higher-rank approximates $\boldsymbol{X}$ for a data-scarce setting.

5. Experiments

This section describes the experimental setup, including datasets, training, evaluation, and models.

Datasets We use four real-world datasets (PPI-BP, HPO-Neuro, HPO-Metab, and EM-User) and four synthetic datasets (Density, Cut-Ratio, Coreness, and Component) introduced in Alsentzer et al. (2020). The task is subgraph classification, where the global graph $\mathcal{G}$ and subgraphs $\mathbb{S}$ are given. The input node features $\boldsymbol{X}$ are pre-trained embedding from Wang & Zhang (2022) for real-world datasets and constant features for synthetic datasets. Dataset statistics and descriptions are in Tables 4, 5, and Appendix E.

Training and evaluation In the original setting from Alsentzer et al. (2020), evaluation (i.e., validation and test) subgraphs cannot be seen during the training stage. Following this protocol, we create different S2N graphs for each stage using train and evaluation sets of subgraphs ($\mathbb{S}_{\text{train}}$ and $\mathbb{S}_{\text{eval}}$). For the S2N translation, we use $\mathbb{S}_{\text{train}}$ only in the training stage and use both $\mathbb{S}_{\text{train}} \cup \mathbb{S}_{\text{eval}}$ in the evaluation stage. We predict unseen nodes based on structures translated from $\mathbb{S}_{\text{train}} \cup \mathbb{S}_{\text{eval}}$ in the evaluation stage. In this respect, node classification on S2N-translated graphs is inductive.

Models We use two well-known GNNs for $\texttt{GNN}_{\texttt{S2N}}$: GCN (Kipf & Welling, 2017) and GCNII (Chen et al., 2020).

Table 1: The number of nodes and edges of real-world graphs before and after S2N translation. The number of edges in S2N is averaged across all experiments.

		PPI-BP	HPO-Neuro	HPO-Metab	EM-User
# Nodes	Original	1.7×10^4	1.5×10^4	1.5×10^4	5.7×10^4
	S2N	1.6×10^3	4.0×10^3	2.4×10^3	3.2×10^2
# Edges	Original	3.2×10^5	3.2×10^6	3.2×10^6	4.6×10^6
	S2N	4.8×10^3	6.3×10^5	6.0×10^4	3.0×10^3

For the node-level encoder $\texttt{GNN}_{\text{node}}$ in S2N+A, we use the same kind of GNN as $\texttt{GNN}_{\text{S2N}}$. See Appendix F for their hyperparameters. We also test these models for connected and separated forms.

Baselines We use basic and state-of-the-art models for subgraph classification tasks as baselines: Sub2Vec (Adhikari et al., 2018), GBDT, SubGNN (Alsentzer et al., 2020), and GLASS (Wang & Zhang, 2022). We report the best performance among the variants of Sub2Vec. All baseline results are reprinted from Alsentzer et al. (2020) and Wang & Zhang (2022).

Efficiency measurement We use each model's best hyperparameters (including batch sizes) and take the mean wall-clock time over 50 epochs. Throughput and latency are all measured using training and validation sets for each stage. We count the number of all trainable parameters, including node embeddings. The maximum allocated GPU VRAM is measured by the PyTorch API. We fix the computation device as Intel(R) Xeon(R) CPU E5-2640 v4 and a single GeForce GTX 1080 Ti in measuring efficiency metrics. We describe details in Appendix G.

Data-scarce experiments Experiments in a data-scarce setting are conducted on benchmarks with the smallest and largest global graphs (PPI-BP and EM-User), and we set the number of training samples per class to 10, 20, 40, and 80. To coarse the global graph, we employ the Variation Edges method (Loukas, 2019) and select the coarsening ratio that generates subgraphs smaller than average sizes. All experiments use GCNII, which performs well across datasets in a fully supervised setting.

6. Results and Discussions

We analyze the characteristics of S2N graphs and compare our models and baselines on classification performance and efficiency. We show that S2N translation results in graph compression (§6.1), which results in better or similar classification accuracy (§6.2) but significantly improves efficiency in terms of computation and memory (§6.3). Finally, we study Coarsened S2N (CoS2N)'s performance and efficiency in a data-scare setting (§6.4).

Table 2: Mean performance in micro F1-score on real-world datasets over 10 runs. For the top 50% of results, the higher the performance, the darker the blue color. The unpaired t-test result with the best is denoted by superscripts at a level of 0.01 ($*$: significantly outperforms, $\sim$: no significant difference). We mark with daggers reprinted results from Alsentzer et al. (2020) (†) and Wang & Zhang (2022) (‡).

Model	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
Sub2Vec†	$30.9_{\pm 2.3}$	$22.3_{\pm 6.5}$	$13.2_{\pm 4.7}$	$85.9_{\pm 1.4}$
GBDT‡	$44.6_{\pm 0.0}$	$51.3_{\pm 0.0}$	$40.4_{\pm 0.0}$	$69.4_{\pm 0.0}$
SubGNN†	$59.9_{\pm 2.4}$	$63.2_{\pm 1.0}$	$53.7_{\pm 2.3}$	$81.4_{\pm 4.6}$
GLASS‡	$61.9_{\pm 0.7}$	$68.5_{\pm 0.5}$	$61.4_{\pm 0.5}$	$88.8_{\pm 0.6}$
Sep. GCN	$61.4_{\pm 2.0}$	$67.6_{\pm 1.0}$	$60.1_{\pm 2.8}$	$84.5_{\pm 4.1}$
Sep. GCNII	$61.3_{\pm 1.2}$	$67.7_{\pm 0.6}$	$59.4_{\pm 2.7}$	$84.7_{\pm 4.1}$
Con. GCN	$62.6_{\pm 1.7}$	$65.7_{\pm 0.8}$	$60.6_{\pm 2.0}$	$85.9_{\pm 2.8}$
Con. GCNII	$63.5_{\pm 2.0}$	$66.7_{\pm 0.8}$	$61.7_{\pm 2.7}$	$85.5_{\pm 4.8}$
S2N+0 GCN	$63.0^{\sim}_{\pm 2.3}$	$66.4_{\pm 0.7}$	$62.0^{\sim}_{\pm 1.6}$	$85.7_{\pm 2.9}$
S2N+0 GCNII	$63.5^{\sim}_{\pm 2.4}$	$66.4_{\pm 1.1}$	$61.6^{\sim}_{\pm 1.7}$	$86.5^{\sim}_{\pm 3.2}$
S2N+A GCN	$63.3^{\sim}_{\pm 2.3}$	$68.3^{\sim}_{\pm 0.9}$	$62.0^{\sim}_{\pm 3.0}$	$86.5_{\pm 2.3}$
S2N+A GCNII	$63.7^{\sim}_{\pm 2.3}$	$68.4^{\sim}_{\pm 1.0}$	$63.2^{\sim}_{\pm 2.7}$	$89.0^{\sim}_{\pm 1.6}$
with RWPE				
S2N+0 GCN	$63.1^{\sim}_{\pm 2.2}$	$66.7_{\pm 0.6}$	$62.3^{\sim}_{\pm 1.9}$	$85.9_{\pm 2.8}$
S2N+0 GCNII	$63.5^{\sim}_{\pm 1.7}$	$66.7_{\pm 0.6}$	$62.3^{\sim}_{\pm 1.1}$	$86.5^{\sim}_{\pm 4.7}$
S2N+A GCN	$63.4^{\sim}_{\pm 2.0}$	$68.4^{\sim}_{\pm 0.7}$	$61.9^{\sim}_{\pm 2.3}$	$87.3^{\sim}_{\pm 9.7}$
S2N+A GCNII	$64.3^{*}_{\pm 1.8}$	$68.6^{\sim}_{\pm 0.8}$	$63.9^{*}_{\pm 1.7}$	$89.0^{\sim}_{\pm 3.1}$

6.1. Analysis of S2N-Translated Graphs

Table 1 summarizes the number of nodes and edges before and after S2N translation. These statistics are from S2N graphs (S2N+0 and S2N+A) tuned for the best performance on GCN and GCNII. The translated graphs have a smaller number of nodes ($\times 0.006$ – $\times 0.27$) and edges ($\times 10^{-4}$ – $\times 0.45$) than the original graphs. See Appendix H for detailed discussion. We also find that they are non-homophilous, meaning many connected node pairs differ in their class. The edge homophily of S2N graphs is 0.25 ± 0.01 for PPI-BP, 0.20 ± 0.03 for HPO-Neuro[1], 0.24 ± 0.01 for HPO-Metab, and 0.51 ± 0.01 for EM-User.

6.2. Performance

Real-World Datasets In Table 2, we report the micro F1-score on real-world datasets. We confirm that S2N with simple GNN models outperforms or is similar to GLASS, the state-of-the-art model. In 16 experiments (4 datasets and 4 models), S2N models outperform GLASS in 9 cases and SubGNN in all 16 cases. Moreover, S2N models are on par with GLASS in 12 of 16 experiments; they have no significant difference at the level of 0.01. The best models with S2N show 102.9% (PPI-BP), 99.9% (HPO-Neuro), 102.9% (HPO-Metab), and 100.2% (EM-User) of GLASS's perfor-

[1]We propose multi-label edge homophily for multi-label datasets (HPO-Neuro). It generalizes the existing multi-class homophily, and we discuss more in Appendix I.

Table 3: Mean performance in micro F1-score on synthetic datasets over 10 runs. For the top 50% of results, the higher the performance, the darker the blue color. The unpaired *t*-test result between S2N and the best is denoted by superscripts ($\sim$: no significant difference at a level of 0.01). We mark with daggers the reprinted results from Alsentzer et al. (2020) (†) and Wang & Zhang (2022) (‡).

Model	Density	Cut-Ratio	Coreness	Component
Sub2Vec‡	$45.9_{\pm 1.2}$	$35.4_{\pm 1.4}$	$36.0_{\pm 1.9}$	$65.7_{\pm 1.7}$
SubGNN†	$91.9_{\pm 1.6}$	$62.9_{\pm 3.9}$	$65.9_{\pm 9.2}$	$95.8_{\pm 9.8}$
GLASS‡	$93.0_{\pm 0.9}$	$93.5_{\pm 0.6}$	$84.0_{\pm 0.9}$	$100.0_{\pm 0.0}$
S2N+0 GCNII	$67.2_{\pm 2.4}$	$56.0_{\pm 0.0}$	$57.0_{\pm 4.9}$	$100.0^{\sim}_{\pm 0.0}$
S2N+A GCNII	$93.2^{\sim}_{\pm 2.6}$	$56.0_{\pm 0.0}$	$85.7^{\sim}_{\pm 5.8}$	$100.0^{\sim}_{\pm 0.0}$
with RWPE				
S2N+0 GCNII	$74.8_{\pm 3.6}$	$85.2_{\pm 5.1}$	$56.1_{\pm 3.0}$	$100.0^{\sim}_{\pm 0.0}$
S2N+A GCNII	$93.6^{\sim}_{\pm 2.0}$	$89.2_{\pm 2.6}$	$72.6_{\pm 6.2}$	$100.0^{\sim}_{\pm 0.0}$

mances. We interpret that message-passing between subgraphs in S2N improves performance by capturing distant interactions that cannot occur in message-passing between nodes in the global graph. We also observe that RWPE generally increases performance. S2N models with RWPE are on par with GLASS in 14 cases and outperform in 13 cases. Plus, S2N+A outperforms S2N+0; internal structure also contributes to subgraph representation. However, the importance of internal structures varies across datasets. For HPO-Neuro as an example, the performance improvement of S2N+A over S2N+0 is high compared to other datasets.

Synthetic Datasets In Table 3, we summarize the performance of S2N models with GCNII and baselines on synthetic datasets. S2N+A performs better than or the same as the state-of-the-art (GLASS) on Density, Coreness, and Component. S2N+0 shows the same performance as GLASS only in Component. We can explain these results through known subgraph properties associated with synthetic labels (Table 6). Because S2N compresses the global graph structure, it is challenging to learn Cut-Ratio, which requires exact information about the global structure. Learning the density and coreness of subgraphs requires their internal structures. Therefore, S2N+0, which does not maintain internal structure, relatively underperforms baselines. RWPE allows S2N to improve the performance of Density and Cut-Ratio significantly, but not of Coreness. We interpret that RWPE for subgraphs can encode internal and border structures well but cannot encode border positions. We leave the development of structural encoding for S2N as future work.

6.3. Efficiency

In Figure 3, we show throughput, latency, the number of parameters, and the maximum allocated GPU VRAM of two models with three data structures and state-of-the-art baselines. We cannot experiment on PPI-BP with SubGNN since it takes more than 48 hours in pre-computation. We make the following five observations from these results.

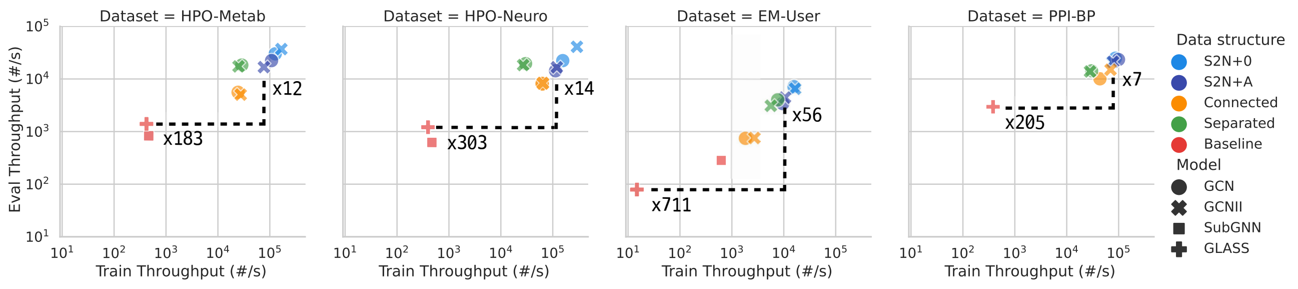
S2N models show significantly high throughput (Fig. 3a). The best S2N models can process $\times 183$ – $\times 711$ more samples than the state-of-the-art model (GLASS) for the same training time. At the evaluation stage, they show 7 – 56 times higher throughput than GLASS. This difference is not as large as the training stage, but S2N is still significantly more efficient than GLASS. In addition, S2N shows higher training throughput than connected and separated forms.

S2N models even with full batch show lower latency than others with small batch size (Fig. 3b). Comparing the best S2N model and GLASS, the training latency is $\times 0.05$ – $\times 0.17$ and evaluation latency is $\times 0.16$ – $\times 0.43$. Note that measuring latency ignores the parallelism from large batch sizes. S2N's superiority over other data structures can be underestimated in latency rather than throughput because it requires full batch computation. Note that existing models need to use small batch sizes because of intensive memory requirements (SubGNN) or model design (GLASS).

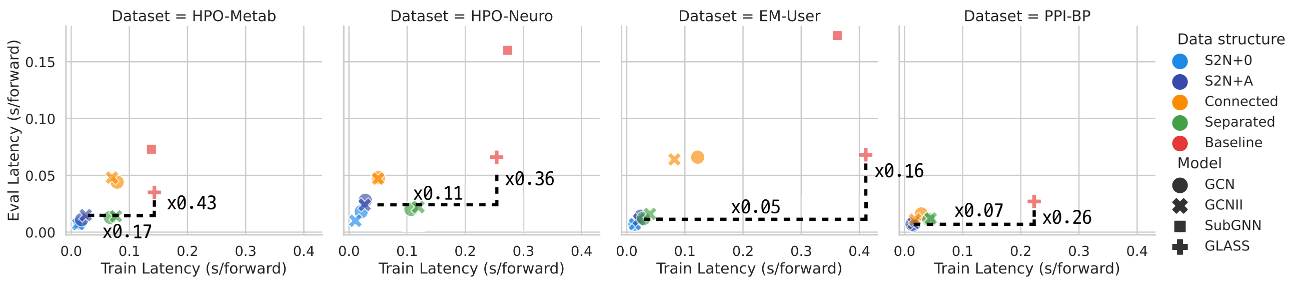
S2N models require less memory even with a similar level of parameters (Fig. 3c). For a given dataset, the number of parameters of each model does not vary much, but the GPU VRAM in actual runtime varies by a large margin. The best models with S2N need less memory ($\times 0.2$ – $\times 0.45$) than GLASS except for HPO-Neuro. HPO-Neuro, which has a large number of subgraphs, requires the same level of memory ($\times 1.05$). In particular, since S2N does not employ a large global graph, S2N works with only $\times 0.13$ memory on average compared to the connected form.

S2N+A does not show a significant difference from S2N+0 in efficiency. Recall that S2N+A differs from S2N+0 by using the internal edges of subgraphs. However, the number of internal edges is negligible compared to the original global edges, as in Table 4. Consequently, the added internal edges require only a small amount of additional computation and memory, allowing S2N+A to perform training and inference efficiently.

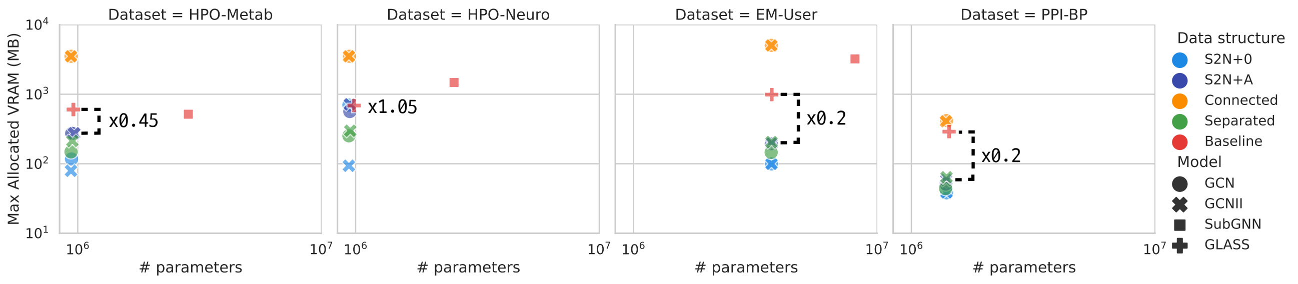
Overall, S2N models outperform baselines in all computational and memory efficiency metrics. Models with S2N process many samples faster (i.e., higher throughput and lower latency), and require less GPU memory than other data structures and state-of-the-art models. The separated form, which does not use a global graph, shows a similar level of efficiency as S2N in some experiments but loses performance by completely ignoring the global structure.



(a) The throughput (the number of subgraphs / second) at training and evaluation stages. *The higher, the better.*



(b) The latency (seconds / forward pass) at training and evaluation stages. *The lower, the better.*



(c) The number of parameters and maximum allocated GPU VRAM. *The lower, the better.*

Figure 3: Efficiency of S2N models and baselines on real-world datasets. The ratio of the best S2N model and the state-of-the-art model for each metric is notated in the figure (dashed lines).

6.4. Performance and Efficiency in a Data-Scarce Setting

In this section, we report the performance and efficiency of S2N, Coarsened S2N (CoS2N), connected form, and separated form. Figure 4 summarizes the performance, throughput, and max allocated VRAM by the number of training samples on PPI-BP. In Appendix L, we analyze similar results on the other dataset and the ablation study on the coarsening ratio.

Virtual subgraphs created in Coarsened S2N contribute to performance improvements of S2N (Figure 4a). We observe that CoS2N consistently outperforms S2N in all conditions. This implies that virtual subgraphs created from graph coarsening in CoS2N enhance communications between subgraphs, leading to better representations. CoS2N+A generally surpasses all other models, including CoS2N+0 and the connected form. When the number of training samples is extremely small, both S2N and CoS2N demonstrate superior performance compared to both baseline models. We confirm that the message-passing between subgraphs is more effective when supervised signals are scarce. In conclusion, CoS2N approximates representations of the global graph well, even though the virtual subgraphs created through coarsening do not follow the distribution of real subgraphs.

CoS2N has higher throughput (Figures 4b, 4c) and uses less memory (Figure 4d) than using the global graph. Although the virtual subgraphs by coarsening are added, both CoS2N methods show higher throughput than using the global graph (i.e., the connected form). CoS2N+0 even shows higher throughput than the separated form in all stages. CoS2N+A shows higher throughput than the separated form in the evaluation stage, where there are more subgraphs to be processed. The training throughput in-

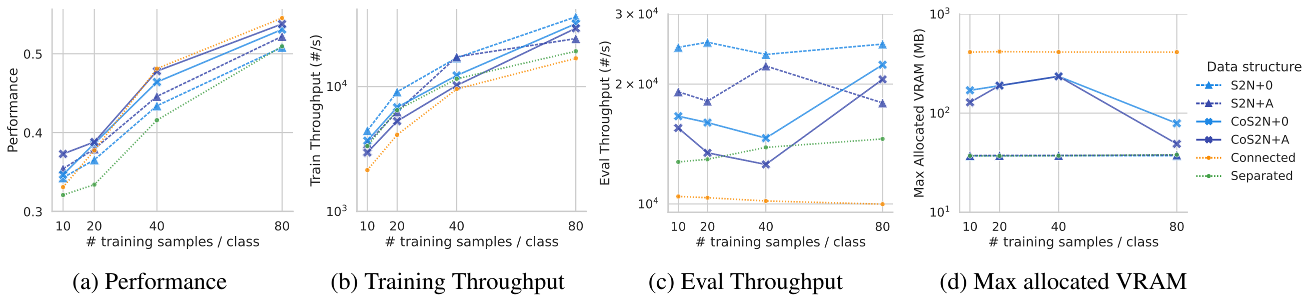


Figure 4: Performance and efficiency on PPI-BP of S2N, CoS2N, connected, and separated forms by the number of training samples in a data-scarce setting.

creases as more training samples are used since the full batch parallelization of GPUs can efficiently process additional samples.

Like computational requirements, CoS2N uses less memory than the connected form. This is because graph coarsening creates fewer subgraphs than the size of the global graph. By the training set size, the memory consumption is constant for the connected form and fluctuates for CoS2N. The memory bottleneck of the connected form and CoS2N is the largest component of each dataset: the global graph and coarsened nodes, respectively. Adding training samples does not substantially affect memory demand; instead, the size of the coarse graph does for CoS2N.

7. Limitations and Future Work

This section discusses the limitations and future work. First, a large S2N graph will be generated if there are excessive subgraphs. This is because the current method translates all subgraphs into nodes in the S2N graph. Future research is needed to sample important subgraphs that sparsely summarize the original global graph. Second, we can observe that the performance of S2N varies depending on the dataset, but this can be known once experiments are conducted. Practitioners can only know which data structures or models are appropriate for a given task by empirical experiments. We leave it to future work to identify which subgraph characteristics affect the performance of models, including S2N. Lastly, we need synthetic tasks for better evaluation. For synthetic datasets, labels are created by pre-designed rules that depend only on structures (e.g., density, cut ratio, and core numbers). These datasets reflect only a very narrow aspect of a subgraph's properties. However, the labels of real-world subgraphs depend on various information about the structures and features. This implies a gap between synthetic and real-world subgraphs, and future studies are required to develop more realistic synthetic tasks.

8. Conclusion

Subgraph-to-node (S2N) translation is a novel, efficient way to learn representations of subgraphs. S2N takes the original subgraphs and creates a new graph where the nodes are the subgraphs, and the edges are the relations between the subgraphs, thereby performing subgraph-level tasks as node-level tasks. We empirically and theoretically show that S2N translation significantly reduces memory and computation costs without performance degradation. Specifically, the best-performing models with S2N on real-world datasets show $\times 183 - \times 711$ of throughput and achieve at least 99.9% of the state-of-the-art models for classification performance.

Acknowledgements

This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921)

Impact Statement

This paper aims to advance the field of graph representation learning, and it is difficult to expect direct societal consequences. Instead, we discuss a potential concern about bias and fairness that has not been confirmed in theory or practice. Models with S2N pass messages between all samples within the batch; thus, the majority in a batch can be overrepresented during prediction and the bias in representations can be amplified. The tasks covered in this paper are unrelated to this, but practitioners should be aware of this for tasks that contain sensitive attributes.

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 170–182. Springer, 2018.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery &amp; data mining*, pp. 2623–2631, 2019.

Alsentzer, E., Finlayson, S. G., Li, M. M., and Zitnik, M. Subgraph neural networks. *Proceedings of Neural Information Processing Systems, NeurIPS*, 2020.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

Barabási, A.-L. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Bordes, A., Chopra, S., and Weston, J. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 615–620, 2014.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Cai, C., Wang, D., and Wang, Y. Graph coarsening with neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uxpzitPEooJ.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Chung, F. Graph theory in the information age. *Notices of the AMS*, 57(6):726–732, 2010.

Consortium, G. O. The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1): D330–D338, 2019.

Dehghani, M., Tay, Y., Arnab, A., Beyer, L., and Vaswani, A. The efficiency misnomer. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iulEMLYh1uR.

Deng, C., Zhao, Z., Wang, Y., Zhang, Z., and Feng, Z. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lGO0EKDH.

Dong, Z., Cao, W., Zhang, M., Tao, D., Chen, Y., and Zhang, X. Cktgnn: Circuit graph neural network for electronic design automation. In *The Eleventh International Conference on Learning Representations*, 2023.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

Falcon, W. and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL https://github.com/PyTorchLightning/pytorch-lightning.

Fatemi, B., El Asri, L., and Kazemi, S. M. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pp. 1972–1982. PMLR, 2019.

Hamidi Rad, R., Bagheri, E., Kargar, M., Srivastava, D., and Szlichta, J. Subgraph representation learning for team mining. In *Proceedings of the 14th ACM Web Science Conference 2022*, pp. 148–153, 2022.

Hartley, T., Lemire, G., Kernohan, K. D., Howley, H. E., Adams, D. R., and Boycott, K. M. New diagnostic approaches for undiagnosed rare genetic diseases. *Annual review of genomics and human genetics*, 21:351–372, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, Z., Zhang, S., Xi, C., Liu, T., and Zhou, M. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 675–684, 2021.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Jin, W., Zhao, L., Zhang, S., Liu, Y., Tang, J., and Shah, N. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WLEx3Jo4QaB.

Kim, D. and Oh, A. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Wi5KUNlqWty.

Kim, D., Jin, J., Ahn, J., and Oh, A. Models and benchmarks for representation learning of partially observed subgraphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4118–4122, 2022.

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

Köhler, S., Carmody, L., Vasilevsky, N., Jacobsen, J. O. B., Danis, D., Gourdine, J.-P., Gargano, M., Harris, N. L., Matentzoglu, N., McMurry, J. A., et al. Expansion of the human phenotype ontology (hpo) knowledge base and resources. *Nucleic acids research*, 47(D1):D1018–D1027, 2019.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Li, S., Zheng, P., Pang, S., Wang, X. V., and Wang, L. Self-organising multiple human–robot collaboration: A temporal subgraph reasoning-based method. *Journal of Manufacturing Systems*, 68:304–312, 2023.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34, 2021.

Loukas, A. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20:1–42, 2019.

Loukas, A. and Vandergheynst, P. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pp. 3237–3246. PMLR, 2018.

Luo, Y. Shine: Subhypergraph inductive neural network. *Advances in Neural Information Processing Systems*, 35: 18779–18792, 2022.

Maheshwari, P., Ren, H., Wang, Y., Sosic, R., and Leskovec, J. Timegraphs: Graph-based temporal reasoning. *arXiv preprint arXiv:2401.03134*, 2024.

Meng, C., Mouli, S. C., Ribeiro, B., and Neville, J. Subgraph pattern neural networks for high-order graph evolution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Mordaunt, D., Cox, D., and Fuller, M. Metabolomics to improve the diagnostic efficiency of inborn errors of metabolism. *International journal of molecular sciences*, 21(4):1195, 2020.

Newman, M. *Networks*. Oxford university press, 2018.

Ni, J., Muhlstein, L., and McAuley, J. Modeling heart rate and activity data for personalized fitness recommendation. In *The World Wide Web Conference*, pp. 1343–1353, 2019.

Ouyang, S., Bai, Q., Feng, H., and Hu, B. Bitcoin money laundering detection via subgraph contrastive learning. *Entropy*, 26(3):211, 2024.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

Trümper, L., Ben-Nun, T., Schaad, P., Calotoiu, A., and Hoefler, T. Performance embeddings: A similarity-based transfer tuning approach to performance optimization. In *Proceedings of the 37th International Conference on Supercomputing*, pp. 50–62, 2023.

Wang, X. and Zhang, M. GLASS: GNN with labeling tricks for subgraph representation learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XLxhEjKNbXj.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Zhang, M. and Li, P. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34: 15734–15747, 2021.

Zhang, X., Xu, S., Lin, W., and Wang, S. Constrained social community recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5586–5596, 2023.

Zhou, H., Liu, S., Lee, K., Shin, K., Shen, H., and Cheng, X. Dpgs: Degree-preserving graph summarization. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 280–288. SIAM, 2021.

Zhou, H., Liu, S., Koutra, D., Shen, H., and Cheng, X. A provable framework of learning graph embeddings via summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4946–4953, 2023.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Zitnik, M., Sosic, R., and Leskovec, J. Biosnap datasets: Stanford biomedical network dataset collection. *Note: http://snap. stanford. edu/biodata Cited by*, 5(1), 2018.

A. Reproducibility Statements

To reproduce the results, we open our code public via the GitHub link[2]. Datasets, including the downloadable link from Alsentzer et al. (2020), are described in Appendix E.

B. Discussion on Related Work

B.1. Detailed Comparison with State-of-the-Art Models: SubGNN and GLASS

SubGNN (Alsentzer et al., 2020), GLASS (Wang & Zhang, 2022), and S2N improve different parts of the machine learning pipeline to solve subgraph-level tasks. SubGNN designs a whole model, GLASS augments input data through a labeling trick, and S2N uses a new data structure.

SubGNN performs message-passing between subgraphs (or patches). Through this, the properties of internal and border structures for three channels (position, neighborhood, and structure) are learned independently. To learn a total of 6 (2×3) properties, SubGNN designs patch samplers, patch representation, and similarity (weights of messages) for each property in an ad hoc manner. For example, SubGNN patches nodes inside the subgraph using its representation as a message and distance-based similarity as weights to learn internal positions. By the complex model design, SubGNN requires a lot of computational resources for data pre-processing, model training, and inference.

GLASS uses plain GNNs but labels input nodes as to whether they belong to the subgraph (the label of one) or not (the label of zero). Separate node-level message-passing is performed for each label to distinguish the internal and border structures of the subgraph. GLASS's labeling trick is effective, but handling multiple labels from multiple subgraphs in a batch is hard. Although the authors of GLASS propose a max-zero-one trick to address this issue, small batches are still recommended. In addition, using a large global graph requires significant computational and memory resources.

S2N uses the new data structure that stores and processes subgraphs efficiently. By compressing the global graph, computational and memory resource requirements are reduced. There are no restrictions on batch sizes; thus, we can train S2N graphs in the full batch. Our contribution lies in addressing the current research direction in subgraph representation learning, where enhancements in performance are often achieved by increasing model complexity. The S2N model challenges it by demonstrating better or comparable performance through a more efficient approach. This efficiency does not merely encompass computational resources. Still, it extends to ease of implementation and adaptability to diverse tasks, making it a significant advancement over current state-of-the-art methods.

B.2. Detailed Comparison with Similar Architectures

Circuit Graph Neural Network (CktGNN) (Dong et al., 2023) and Nested Graph Neural Network (NGNN) (Zhang & Li, 2021) are similar to our S2N+A in that they employ two-level GNNs, where the first GNN learns the embedding of subgraphs, and the second GNN performs message-passing between subgraphs. However, our study differs from these in the following aspects. First, they do not focus on subgraph-level tasks. CktGNN is applied for circuit design automation, and NGNN is applied for graph regression and classification. They do not demonstrate how the two-level GNNs approach affects subgraph representations rather than the whole graph. In contrast, we analyzed our S2N models empirically and theoretically on subgraph representation learning. Second, they also make strong assumptions about subgraphs and cannot be generalized to all subgraph-level tasks. Specifically, CktGNN uses a set of pre-designed subgraphs specialized for circuits. A subgraph in NGNN is the rooted subgraph of each node in the given graph. For these two models, the subgraph has to be in a fixed shape; thus, they cannot handle subgraphs of various structures and sizes.

Junction Tree Variational Autoencoder (JT-VAE) (Jin et al., 2018) decomposes a molecular graph into a junction tree, where a node corresponds to the motif (particularly a ring of atoms), and edges link the nodes that share the nodes. This method is a graph generation model to learn the ring substructure well in chemical tasks but has not been used in subgraph-level tasks. Due to the nature of the Junction Tree algorithm, only the ring (or cycle) structure of input graphs is used as subgraphs, and the output is restricted to trees, which leads to limited usage. Our proposed S2N's primary contribution is to explore the fundamental question of subgraph representation learning and propose a novel perspective. In addition, S2N can be generally applied to graphs and subgraphs of any structure.

DiffPool (Ying et al., 2018) learns the hierarchy of a graph to obtain graph-level representations. DiffPool softly assigns

[2]https://github.com/dongkwan-kim/S2N

each node to a cluster during training by optimizing the downstream task loss. To stabilize the soft clustering assignment, the authors of DiffPool employ link prediction loss and entropy regularization loss. The problem is that the assignment matrix must be maintained in GPU memory, which requires quadratic memory complexity regarding the number of nodes. In other words, we cannot apply DiffPool to large graphs such as global graphs in our use cases. We aim to perform subgraph representation learning efficiently by compressing data and reducing GPU load. Memory-intensive graph coarsening, such as DiffPool's soft clustering assignment, should not be used to keep CoS2N efficient. Instead, we can secure the efficiency of CoS2N by performing graph coarsening before training the model, relying only on the structure of the global graph.

C. Justification for the Choice of the Random Graph Model

The configuration model (CM) only requires a degree sequence or a distribution. That means CM can also generate graphs generated by other random graph models. For example, when the degree distribution is Poisson distribution, CM generates graphs close to the Erdős–Rényi model. CM can also adopt other degree distributions, for example, power-law distributions. See Newman (2018) for more details.

We also emphasize that the complexity of S2N strongly depends on the number of edges in S2N, that is, how many edges of small weights are removed during normalization. Thus, we need a random graph model that can analytically calculate the distribution of edge weights (i.e., the number of shared edges in two subgraphs). When using the CM, the distribution of S2N's edge weights can be derived from the degree distribution of the global graph. This is possible because CM calculates the probability of edge existence through the degrees of a pair of nodes. Note that CM is frequently used in analytically calculating numerous network measures (Barabási, 2013).

D. Proofs of Theoretical Analysis

This section describes proofs of theoretical claims in the paper: Propositions 4.1, 4.2, and 4.3. In addition, we analyze the error bound between S2N and the global graph for a variant of Graph Isomorphism Networks (Xu et al., 2019) in Proposition D.5 and Corollary D.6.

Proposition D.1 (Proposition 4.1)**.** *The time complexity of the 1-layer GLASS, Connected form, S2N+0, and S2N+A is*

$$\text{GLASS \& Connected: } O(EF + M\overline{N^{\text{sub}}}F + NF^2), \tag{13}$$

$$\text{S2N+0: } O(\hat{E}F + M\overline{N^{\text{sub}}}F + MF^2), \tag{14}$$

$$\text{S2N+A: } O(\hat{E}F + M\overline{E^{\text{sub}}}F + M\overline{N^{\text{sub}}}F^2). \tag{15}$$

Proof. Let $\mathcal{G} = (\mathbb{V}, \boldsymbol{A}, \boldsymbol{X})$ be a global graph where $\mathbb{V}$ is a set of nodes ($|\mathbb{V}| = N$), $\boldsymbol{A} \in \{0,1\}^{N \times N}$ is an adjacency matrix, and $\boldsymbol{X} \in \mathbb{R}^{N \times F_0}$ is a node feature matrix. A subgraph $\mathcal{S} = (\mathbb{V}^{\text{sub}}, \boldsymbol{A}^{\text{sub}})$ is a graph formed by subsets of nodes and edges in the global graph $\mathcal{G}$. For the subgraph classification task, there is a set of M subgraphs $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_M\}$, and for $\mathcal{S}_i = (\mathbb{V}_i^{\text{sub}}, \boldsymbol{A}_i^{\text{sub}})$, the goal is to learn subgraph representations $\hat{\boldsymbol{H}} \in \mathbb{R}^{M \times F}$.

Baselines and S2N models are computed by following steps:

- GLASS & Connected: $\hat{\boldsymbol{H}} = \boldsymbol{R}^\top \texttt{GNN}(\boldsymbol{X}, \boldsymbol{A})$ where $\boldsymbol{R} \in \mathbb{R}^{N \times M}$ is a readout matrix.
- S2N+0: $\hat{\boldsymbol{H}} = \texttt{GNN}_{\texttt{S2N}}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{A}})$ where $\hat{\boldsymbol{X}}_{[i]} = \sum_{v \in \mathbb{V}_i^{\text{sub}}} \omega_{vi} \cdot \boldsymbol{X}_{[v,:]}$.
- S2N+A: $\hat{\boldsymbol{H}} = \texttt{GNN}_{\texttt{S2N}}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{A}})$ where $\hat{\boldsymbol{X}}_{[i]} = \sum_{v \in \mathbb{V}_i^{\text{sub}}} \omega_{vi} \cdot \texttt{GNN}_{\texttt{node}}(\boldsymbol{X}_{[\mathbb{V}_i^{\text{sub}},:]}, \boldsymbol{A}_i^{\text{sub}})_{[v,:]}$.

Graph neural networks (GNNs) that use the message-passing mechanism to learn subgraph representations can be decomposed into feature transformation (FT), feature propagation (FP), and subgraph-level readout (SR). Feature transformation requires $O(\text{the number of nodes} \times F^2)$ computations and feature propagation by sparse implementation requires $O(\text{the number of edges} \times F)$ computations. Plus, for the readout of representations or input features, we need the computations of $O(\text{the total number of nodes in subgraphs} \times F)$.

- GLASS & Connected: $O(EF)$ from FP, $O(M\overline{N^{\text{sub}}}F)$ from SR, and $O(NF^2)$ from FT.
- GLASS: $O(M\overline{N^{\text{sub}}})$ from the node labeling trick (Wang & Zhang, 2022).

- S2N+0: $O(\hat{E}F)$ from FP, $O(M\overline{N^{\text{sub}}}F)$ from SR, and $O(MF^2)$ from FT.
- S2N+A: $O(\hat{E}F)$ and $O(M\overline{E^{\text{sub}}}F)$ from FP in $\texttt{GNN}_{\text{S2N}}$ and $\texttt{GNN}_{\text{node}}$, $O(M\overline{N^{\text{sub}}}F)$ from SR, and $O(MF^2)$ and $O(M\overline{N^{\text{sub}}}F^2)$ from FT in $\texttt{GNN}_{\text{S2N}}$ and $\texttt{GNN}_{\text{node}}$.

By adding up all the terms, we can get the final result. □

Proposition D.2 (Proposition 4.2)**.** *For Configuration Model as $\mathcal{G}$ and i.i.d. sampled subgraphs where the average size is $\overline{N^{sub}}$, the probability that the weight $\hat{\boldsymbol{A}}_{[i,j]}$ of an edge (i,j) in $\hat{\mathcal{G}}$ is bigger than $c > 0$ is $P(\hat{\boldsymbol{A}}_{[i,j]} \geq c) \leq \frac{(\overline{N^{sub}})^2\mathbb{E}[d]}{cN}$ where $\mathbb{E}[d]$ is an average degree of $\mathcal{G}$.*

Proof. We first note that the probability of edge (u,v) in the Configuration Model for large E is $\frac{d_u d_v}{2E}$ and $E = \frac{1}{2}\sum_k d_k = \frac{1}{2}N\mathbb{E}[d]$ (Newman, 2018).

$$\begin{aligned}
P(\hat{\boldsymbol{A}}_{[i,j]} \geq c) &\leq \frac{\mathbb{E}[\hat{\boldsymbol{A}}_{[i,j]}]}{c} \quad (\because \text{Markov's inequality}) && (16)\\
&= \frac{\mathbb{E}[\sum_{u\in\mathbb{V}_i^{\text{sub}}}\sum_{v\in\mathbb{V}_j^{\text{sub}}}\boldsymbol{A}_{[u,v]}]}{c} && (17)\\
&= \frac{\mathbb{E}[\sum_{u\in\mathbb{V}_i^{\text{sub}}}\sum_{v\in\mathbb{V}_j^{\text{sub}}}\frac{d_u d_v}{2E}]}{c} && (18)\\
&= \frac{\mathbb{E}_{(i,j)\in\mathbb{S}\times\mathbb{S}}[\sum_{u\in\mathbb{V}_i^{\text{sub}}}\sum_{v\in\mathbb{V}_j^{\text{sub}}}\mathbb{E}[d]^2]}{2cE} && (19)\\
&= \frac{(\overline{N^{\text{sub}}}\mathbb{E}[d])^2}{2cE} && (20)\\
&= \frac{(\overline{N^{\text{sub}}})^2\mathbb{E}[d]}{cN} && (21)
\end{aligned}$$

□

To prove Proposition 4.3, we first introduce Lemma D.3.

Lemma D.3.

$$\boldsymbol{R}^\top\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{R} = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}} \quad (22)$$

Proof.

$$\begin{aligned}
\boldsymbol{R}^\top\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{R} &= (\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{M}\hat{\boldsymbol{D}}^{-\frac{1}{2}})^\top\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{M}\hat{\boldsymbol{D}}^{-\frac{1}{2}} && (23)\\
&= \hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{M}^\top\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{M}\hat{\boldsymbol{D}}^{-\frac{1}{2}} && (24)\\
&= \hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{M}^\top\boldsymbol{A}\boldsymbol{M}\hat{\boldsymbol{D}}^{-\frac{1}{2}} && (25)\\
&= \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}. && (26)
\end{aligned}$$

□

Proposition D.4 (Proposition 4.3)**.** *Using the single-layer GCN parametrized by $\boldsymbol{W}$, subgraph representations $\boldsymbol{R}^\top\boldsymbol{H}$ of the global graph $\mathcal{G}$ can be approximated by node representations $\hat{\boldsymbol{H}}$ of the S2N graph $\hat{\mathcal{G}}$, that is, $\hat{\boldsymbol{H}} \approx \boldsymbol{R}^\top\boldsymbol{H}$. The error between two representations is bounded by:*

$$\|\boldsymbol{R}^\top\boldsymbol{H} - \hat{\boldsymbol{H}}\| \leq M^{\frac{1}{2}}\|\boldsymbol{X} - \boldsymbol{R}\hat{\boldsymbol{X}}\| \cdot \|\boldsymbol{W}\|. \quad (27)$$

Proof.

$$\begin{aligned}
\|\boldsymbol{R}^\top \boldsymbol{H} - \hat{\boldsymbol{H}}\| &= \|\boldsymbol{R}^\top \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{W} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{X}}\boldsymbol{W}\| && (28)\\
&= \|\boldsymbol{R}^\top \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{W} - \boldsymbol{R}^\top\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{R}\hat{\boldsymbol{X}}\boldsymbol{W}\| \quad (\because \text{Lemma D.3}) && (29)\\
&= \|\boldsymbol{R}^\top (\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})(\boldsymbol{X} - \boldsymbol{R}\hat{\boldsymbol{X}})\boldsymbol{W}\| && (30)\\
&\le \|\boldsymbol{R}^\top\|\|\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\|\|\boldsymbol{X} - \boldsymbol{R}\hat{\boldsymbol{X}}\|\|\boldsymbol{W}\| && (31)\\
&\le M^{\frac{1}{2}}\|\boldsymbol{X} - \boldsymbol{R}\hat{\boldsymbol{X}}\| \cdot \|\boldsymbol{W}\|. && (32)
\end{aligned}$$

□

Although Proposition 4.3 is analyzed using GCN models only, it is not limited to GCN in its applicability. Intuitively, when sufficient subgraph samples are unavailable, message-passing in any GNNs fails in the global graph not covered by existing subgraphs. Moreover, we can obtain theoretical results similar to Proposition 4.3 for other GNNs. However, we might not get the approximation bound analytically depending on GNN architectures. For Graph Isomorphism Network (GIN) (Xu et al., 2019) as an example, the non-linearity in multi-layer perceptron (MLP) makes it hard to analytically compare the GIN outputs of S2N and the original graph. Instead, we introduce an approximation error bound on 'GIN Sum-1-Layer', a less powerful variant of GINs that replaces MLP with single-layer perceptron (SLP).

$$\begin{aligned}
\text{GIN: } \boldsymbol{H} &= \text{MLP}\left((\mathbf{A} + (1+\epsilon)\cdot \mathbf{I})\cdot \boldsymbol{X}\right), && (33)\\
\text{GIN Sum-1-Layer: } \boldsymbol{H} &= \text{SLP}\left((\mathbf{A} + (1+\epsilon)\cdot \mathbf{I})\cdot \boldsymbol{X}\right). && (34)
\end{aligned}$$

The error bound between S2N's node representations and the global graph's subgraph representations is demonstrated in Proposition D.5. Here, we use the sum-readout $\texttt{READOUT}(\boldsymbol{H}) = \boldsymbol{M}^\top \boldsymbol{H}$ to get subgraph representations.

Proposition D.5. *Using the single-layer GIN Sum-1-Layer parametrized by $\boldsymbol{W}$, subgraph representations $\boldsymbol{M}^\top\boldsymbol{H}$ of the global graph $\mathcal{G}$ can be approximated by node representations $\hat{\boldsymbol{H}}$ of the S2N graph $\hat{\mathcal{G}}$, that is, $\hat{\boldsymbol{H}} \approx \boldsymbol{M}^\top\boldsymbol{H}$. The error between two representations is bounded by:*

$$\|\boldsymbol{M}^\top \boldsymbol{H} - \hat{\boldsymbol{H}}\| \le \left((M\overline{N^{sub}}E)^{\frac{1}{2}}\|\boldsymbol{X} - \boldsymbol{M}\hat{\boldsymbol{X}}\| + (1+\epsilon)\|\boldsymbol{M}^\top\boldsymbol{X} - \hat{\boldsymbol{X}}\|\right)\cdot\|\boldsymbol{W}\|. \tag{35}$$

Proof.

$$\begin{aligned}
\|\boldsymbol{M}^\top \boldsymbol{H} - \hat{\boldsymbol{H}}\| &= \|\boldsymbol{M}^\top(\boldsymbol{A} + (1+\epsilon)\boldsymbol{I}_N)\boldsymbol{X}\boldsymbol{W} - (\hat{\boldsymbol{A}} + (1+\epsilon)\boldsymbol{I}_M)\hat{\boldsymbol{X}}\boldsymbol{W}\| && (36)\\
&= \|\boldsymbol{M}^\top(\boldsymbol{A} + (1+\epsilon)\boldsymbol{I}_N)\boldsymbol{X}\boldsymbol{W} - (\boldsymbol{M}^\top\boldsymbol{A}\boldsymbol{M} + (1+\epsilon)\boldsymbol{I}_M)\hat{\boldsymbol{X}}\boldsymbol{W}\| && (37)\\
&= \|\boldsymbol{M}^\top\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{M}\hat{\boldsymbol{X}})\boldsymbol{X}\boldsymbol{W} + (1+\epsilon)(\boldsymbol{M}^\top\boldsymbol{X} - \hat{\boldsymbol{X}})\boldsymbol{W}\| && (38)\\
&\le \|\boldsymbol{M}^\top\|\|\boldsymbol{A}\|\|\boldsymbol{X} - \boldsymbol{M}\hat{\boldsymbol{X}}\|\|\boldsymbol{W}\| + (1+\epsilon)\|\boldsymbol{M}^\top\boldsymbol{X} - \hat{\boldsymbol{X}}\|\|\boldsymbol{W}\| && (39)\\
&\le \left((M\overline{N^{\text{sub}}}E)^{\frac{1}{2}}\|\boldsymbol{X} - \boldsymbol{M}\hat{\boldsymbol{X}}\| + (1+\epsilon)\|\boldsymbol{M}^\top\boldsymbol{X} - \hat{\boldsymbol{X}}\|\right)\cdot\|\boldsymbol{W}\|, && (40)
\end{aligned}$$

where $\boldsymbol{I}_N$ is an identity matrix of size N. □

If we set the initial features of S2N as a sum of the original features (i.e., $\hat{\boldsymbol{X}} = \boldsymbol{M}^\top\boldsymbol{X}$), Corollary D.6 then follows from Proposition D.5.

Corollary D.6. *Using the single-layer GIN Sum-1-Layer parametrized by $\boldsymbol{W}$, subgraph representations $\boldsymbol{M}^\top\boldsymbol{H}$ of the global graph $\mathcal{G}$ can be approximated by node representations $\hat{\boldsymbol{H}}$ of the S2N graph $\hat{\mathcal{G}}$, that is, $\hat{\boldsymbol{H}} \approx \boldsymbol{M}^\top\boldsymbol{H}$. If the initial feature matrix of S2N is $\hat{\boldsymbol{X}} = \boldsymbol{M}^\top\boldsymbol{X}$, the error between two representations is bounded by:*

$$\|\boldsymbol{M}^\top \boldsymbol{H} - \hat{\boldsymbol{H}}\| \le (M\overline{N^{sub}}E)^{\frac{1}{2}}\|\boldsymbol{X} - \boldsymbol{M}\hat{\boldsymbol{X}}\|\cdot\|\boldsymbol{W}\|. \tag{41}$$

Table 4: Statistics of real-world datasets in original forms (before S2N translation).

	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
# nodes in $\mathcal{G}$	17,080	14,587	14,587	57,333
# edges in $\mathcal{G}$	316,951	3,238,174	3,238,174	4,573,417
# internal edges in subgraphs	9,627	217,555	390,450	86,648
# subgraphs	1,591	4,000	2,400	324
Density of $\mathcal{G}$	0.0022	0.0304	0.0304	0.0028
Average density of subgraphs	$0.216_{\pm 0.188}$	$0.767_{\pm 0.141}$	$0.757_{\pm 0.149}$	$0.010_{\pm 0.006}$
Average # nodes / subgraph	$10.2_{\pm 10.5}$	$14.8_{\pm 6.5}$	$14.4_{\pm 6.2}$	$155.4_{\pm 100.2}$
Average # components / subgraph	$7.0_{\pm 5.5}$	$1.5_{\pm 0.7}$	$1.6_{\pm 0.7}$	$52.1_{\pm 15.3}$
# classes	6	10	6	2
Single- or multi-label	Single-label	Multi-label	Single-label	Single-label
Train/Valid/Test splits	80/10/10	80/10/10	80/10/10	70/15/15

E. Datasets

All real-world subgraph datasets (PPI-BP, HPO-Neuro, HPO-Metab, and EM-User) and synthetic subgraph datasets (Density, Cut-Ratio, Coreness, and Component) are proposed in Alsentzer et al. (2020). They can be downloaded from the author's GitHub repository[3]. Pre-trained embeddings can be downloaded from the GitHub repository[4] of Wang & Zhang (2022). The following paragraphs describe their nodes, edges, subgraphs, tasks, and references. Note that the number of edges in the real-world datasets compared to datasets referred to as large-scale (Lim et al., 2021) is at a similar level; thus, similar scalability is required to model real-world graphs using GNNs.

E.1. Real-World Datasets

PPI-BP The global graph of PPI-BP (Zitnik et al., 2018; Subramanian et al., 2005; Consortium, 2019; Ashburner et al., 2000) is a human protein-protein interaction (PPI) network; nodes are proteins, and edges are whether there is a physical interaction between proteins. Subgraphs are sets of proteins in the same biological process (e.g., alcohol bio-synthetic process). The task is to classify processes into six categories.

HPO-Neuro and HPO-Metab These two HPO (Human Phenotype Ontology) datasets (Hartley et al., 2020; Köhler et al., 2019; Mordaunt et al., 2020) are knowledge graphs of phenotypes (i.e., symptoms) of rare neurological and metabolic diseases. Each subgraph is a collection of symptoms associated with a monogenic disorder. The task is to diagnose the rare disease: classifying the disease type among subcategories (ten for HPO-Neuro and six for HPO-Metab).

EM-User EM-User (Users in EndoMondo) dataset is a social fitness network from Endomondo (Ni et al., 2019). Here, subgraphs are users, nodes are workouts, and edges exist between workouts completed by multiple users. Each subgraph represents the workout history of a user. The task is to profile a user's gender.

E.2. Synthetic Datasets

Density, Cut-Ratio, Coreness, and Component For these synthetic datasets, the task is to predict the properties of subgraphs: density, cut ratio, average core number, and the number of components, respectively. Refer to Alsentzer et al. (2020) for details on how to generate the synthetic graphs. We use a vector of 64 dimensions initialized to 1 or its L1-normalized vector as input node embedding.

F. Models

We describe the hyperparameter details and the tuning method. All models are implemented with PyTorch (Paszke et al., 2019), PyTorch Geometric (Fey & Lenssen, 2019), and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019).

[3] `https://github.com/mims-harvard/SubGNN`
[4] `https://github.com/Xi-yuanWang/GLASS`

Table 5: Statistics of synthetic datasets in original forms (before S2N translation).

	Density	Cut-Ratio	Coreness	Component
# nodes in $\mathcal{G}$	5,000	5,000	5,000	19,555
# edges in $\mathcal{G}$	29,521	83,969	118,785	43,701
# subgraphs	250	250	221	250
Density of $\mathcal{G}$	0.0024	0.0067	0.0095	0.0002
Average density of subgraphs	$0.232_{\pm 0.146}$	$0.945_{\pm 0.028}$	$0.219_{\pm 0.062}$	$0.150_{\pm 0.161}$
Average # nodes / subgraph	$20.0_{\pm 0.0}$	$20.0_{\pm 0.0}$	$20.0_{\pm 0.0}$	$74.2_{\pm 52.8}$
Average # components / subgraph	$3.8_{\pm 3.7}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$4.9_{\pm 3.5}$
# classes	3	3	3	2
Single- or multi-label	Single-label	Single-label	Single-label	Single-label
Train/Valid/Test splits	80/10/10	80/10/10	80/10/10	80/10/10

Table 6: The attributes that affect the subgraph properties (labels) of synthetic datasets, introduced in Alsentzer et al. (2020).

Density	Cut-Ratio	Coreness	Component
Internal structure	Border structure	Internal structure, border structure & position	Internal & external position

We tune hyperparameters using TPE (Tree-structured Parzen Estimator) algorithm in Optuna (Akiba et al., 2019) by 400 trials: learning rate (5×10^{-4} – 10^{-2}), weight decay (10^{-9} – 10^{-6}), the number of layers in GNN (1 – 2), dropout of channels and edges ($\{0.0, 0.1, ..., 0.5\}$), gradient clipping ($\{0.0, 0.1, ..., 0.5\}$), the readout matrix ($\omega_{vi} = \boldsymbol{R}_{[v,i]}$ in Equation 11 or $\omega_{vi} = \boldsymbol{M}_{[v,i]}$), and whether to use batch normalization (Ioffe & Szegedy, 2015) and skip-connection (He et al., 2016). Hyperparameters specialized on GCNII are also tuned: α ($\{0.1, 0.2, ..., 0.9\}$), θ ($\{0.1, 0.2, ..., 2.0\}$), weight sharing (True or False). For S2N translation, we tune edge normalization range (a and $b = a + \Delta$ in Equation 3, $a \in \{1.0, 1.25, ..., 4.0\}$, $\Delta \in \{0.5, 1.0, 1.5, 2.0\}$). We add frozen Random Walk Positional Encoding (RWPE) (Dwivedi et al., 2022) to input features for real-world datasets. For synthetic datasets, we allocate RWPE to 1/2 or 1/4 of the total embedding dimension.

All hyperparameters are reported in the code.

G. Efficiency Measurement

We compute throughput (subgraphs per second) and latency (seconds per forward pass) using the following equations. In addition, we use `torch.cuda.max_memory_allocated` to measure the maximum allocated GPU VRAM[5].

$$\text{Training throughput} = \frac{\text{\# of training subgraphs}}{\text{training wall-clock time (seconds) / \# of epochs}}, \tag{42}$$

$$\text{Evaluation throughput} = \frac{\text{\# of validation subgraphs}}{\text{validation wall-clock time (seconds) / \# of epochs}}, \tag{43}$$

$$\text{Training latency} = \frac{\text{training wall-clock time (seconds)}}{\text{\# of training batches}}, \tag{44}$$

$$\text{Evaluation latency} = \frac{\text{validation wall-clock time (seconds)}}{\text{\# of validation batches}}. \tag{45}$$

While throughput is a primary metric in practice, focusing solely on this metric is suboptimal for best model selection. Specifically, we observed that connected forms suffer from extensive memory consumption, and separated forms exhibit significant performance degradation compared to our proposed S2N models. The holistic evaluation of performance, throughput, and memory requirements is crucial for practitioners to decide which models to employ based on their specific constraints and requirements, as stated in Dehghani et al. (2022).

[5]https://pytorch.org/docs/1.9.0/generated/torch.cuda.max_memory_allocated.html

Table 7: Mean performance in micro F1-score over 10 runs. We mark with daggers the reprinted results from Alsentzer et al. (2020) (†) and Wang & Zhang (2022) (‡).

Model	Data Structure	PPI-BP	EM-User
Sub2Vec Best†		$30.9_{\pm 2.3}$	$85.9_{\pm 1.4}$
SubGNN†		$59.9_{\pm 2.4}$	$81.4_{\pm 4.6}$
GLASS‡		$61.9_{\pm 0.7}$	$88.8_{\pm 0.6}$
GCNII	Separated	$61.3_{\pm 1.2}$	$84.7_{\pm 4.1}$
GCNII	Connected	$63.5_{\pm 2.0}$	$85.5_{\pm 4.8}$
GCNII	S2N+0	$63.5_{\pm 2.4}$	$86.5_{\pm 3.2}$
GCNII	S2N+A	$63.7_{\pm 2.3}$	$89.0_{\pm 1.6}$
GIN	Separated	$60.6_{\pm 2.1}$	$82.2_{\pm 6.6}$
GIN	Connected	$61.0_{\pm 3.3}$	$83.7_{\pm 4.8}$
GIN	S2N+0	$63.3_{\pm 1.6}$	$84.9_{\pm 5.3}$
GIN	S2N+A	$62.2_{\pm 1.9}$	$83.1_{\pm 1.6}$
GATv2	Separated	$61.4_{\pm 2.6}$	$84.7_{\pm 4.9}$
GATv2	Connected	$61.0_{\pm 1.5}$	OOM
GATv2	S2N+0	$62.8_{\pm 1.7}$	$84.9_{\pm 2.4}$
GATv2	S2N+A	$62.6_{\pm 1.4}$	$86.7_{\pm 3.2}$

H. Discussion on the Number of Nodes and Edges in S2N

The number of nodes in S2N+A in Table 1 only includes nodes translated from subgraphs. The total number of nodes in S2N+A may be larger than the original global graph if we count all internal nodes in the subgraph. However, this is not significantly related to actual efficiency. The reason is that in S2N+A, internal nodes in the subgraph are kept sparse (i.e., indexing) rather than dense embedding by implementation, so memory cost is low. Specifically, node indexing requires the space complexity of $O(N)$, and dense embedding requires $O(NF)$, where N is the number of nodes and F is the number of features. The dominant factors of computational and memory bottlenecks are the number of subgraphs (the number of nodes in S2N), which determines the size of the final representation, and the number of edges, which determines the number of message-passing, as illustrated in Table 1.

I. Generalization of Homophily to Multi-label Classification

Node (Pei et al., 2020) and edge homophily (Zhu et al., 2020) are defined by,

$$h^{\text{edge}} = \frac{|\{(u,v)|(u,v) \in \mathbb{A} \wedge y_u = y_v\}|}{|\mathbb{A}|},\ h^{\text{node}} = \frac{1}{|\mathbb{V}|} \sum_{v \in \mathbb{V}} \frac{|\{(u,v)|u \in \mathcal{N}(v) \wedge y_u = y_v\}|}{|\mathcal{N}(v)|}, \tag{46}$$

where y_v is the label of the node v. In the main paper, we define multi-label node and edge homophily by,

$$h^{\text{edge, ml}} = \frac{1}{|\mathbb{A}|} \sum_{(u,v) \in \mathbb{A}} \frac{|\mathbb{L}_u \cap \mathbb{L}_v|}{|\mathbb{L}_u \cup \mathbb{L}_v|},\ h^{\text{node, ml}} = \frac{1}{|\mathbb{V}|} \sum_{v \in \mathbb{V}} \left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \frac{|\mathbb{L}_u \cap \mathbb{L}_v|}{|\mathbb{L}_u \cup \mathbb{L}_v|} \right). \tag{47}$$

If we compute $r = \frac{|\mathbb{L}_u \cap \mathbb{L}_v|}{|\mathbb{L}_u \cup \mathbb{L}_v|}$ for single-label multi-class graphs, $r = \frac{1}{1} = 1$ for nodes of same classes, and $r = \frac{0}{2} = 0$ for nodes of different classes. That makes $h^{\text{edge, ml}} = h^{\text{edge}}$ and $h^{\text{node, ml}} = h^{\text{node}}$ for single-label graphs.

J. Performance of Different GNN Layers

In Table 7, we demonstrate the performance of S2N models using additional GNN layers: Graph Isomorphism Networks (GIN) (Xu et al., 2019) and Graph Attention Networks v2 (GATv2) (Brody et al., 2022) on PPI-BP and EM-User.

GIN and GATv2 (S2N+0 and S2N+A) outperform GLASS on PPI-BP but perform worse than on EM-User. We confirm that S2N outperforms classic data structures: separated and connected forms. For GATv2, we cannot experiment with the

Table 8: Mean performance in micro F1-score over 10 runs using GCNII models with different readout methods.

Data Structure	Readout	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
S2N+0	Sum	$\mathbf{63.5}_{\pm\mathbf{2.4}}$	$\mathbf{66.4}_{\pm\mathbf{1.1}}$	$61.6_{\pm1.7}$	$\mathbf{86.5}_{\pm\mathbf{3.2}}$
	Mean	$59.9_{\pm2.0}$	$63.9_{\pm0.5}$	$\mathbf{62.0}_{\pm\mathbf{1.0}}$	$85.3_{\pm3.9}$
	Max	$48.6_{\pm2.8}$	$54.1_{\pm1.0}$	$51.8_{\pm2.2}$	$72.2_{\pm7.1}$
	Degree	$60.3_{\pm2.2}$	$65.6_{\pm1.0}$	$58.8_{\pm2.3}$	$83.1_{\pm2.9}$
S2N+A	Sum	$\mathbf{63.7}_{\pm\mathbf{2.3}}$	$\mathbf{68.4}_{\pm\mathbf{1.0}}$	$\mathbf{63.2}_{\pm\mathbf{2.7}}$	$88.8_{\pm2.1}$
	Mean	$59.6_{\pm1.5}$	$66.6_{\pm1.0}$	$60.8_{\pm1.1}$	$88.0_{\pm3.2}$
	Max	$58.5_{\pm1.7}$	$59.6_{\pm2.0}$	$59.4_{\pm2.7}$	$81.2_{\pm3.1}$
	Degree	$63.1_{\pm2.2}$	$\mathbf{68.4}_{\pm\mathbf{0.9}}$	$61.0_{\pm2.0}$	$\mathbf{89.0}_{\pm\mathbf{1.6}}$

Table 9: Mean performance in micro F1-score over 10 runs using GCNII models with different numbers of layers of $\texttt{GNN}_{\text{S2N}}$.

Data Structure	# layers	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
S2N+0	0	$57.7_{\pm1.6}$	$65.2_{\pm1.6}$	$55.5_{\pm1.9}$	$77.6_{\pm9.4}$
	1	$61.1_{\pm2.4}$	$\mathbf{66.4}_{\pm\mathbf{1.1}}$	$\mathbf{61.6}_{\pm\mathbf{1.7}}$	$79.2_{\pm9.2}$
	2	$\mathbf{63.5}_{\pm\mathbf{2.4}}$	$65.6_{\pm1.4}$	$59.4_{\pm1.0}$	$\mathbf{86.5}_{\pm\mathbf{3.2}}$
	4	$62.8_{\pm2.0}$	$65.8_{\pm0.8}$	$61.1_{\pm1.7}$	$79.2_{\pm7.9}$
S2N+A	0	$59.7_{\pm2.2}$	$68.2_{\pm0.8}$	$61.8_{\pm1.7}$	$87.1_{\pm3.5}$
	1	$\mathbf{63.7}_{\pm\mathbf{2.3}}$	$\mathbf{68.4}_{\pm\mathbf{1.0}}$	$61.9_{\pm2.0}$	$\mathbf{89.0}_{\pm\mathbf{1.6}}$
	2	$61.8_{\pm1.4}$	$68.0_{\pm0.8}$	$\mathbf{63.2}_{\pm\mathbf{2.7}}$	$86.3_{\pm4.9}$
	4	$61.6_{\pm1.7}$	$67.7_{\pm0.8}$	$62.0_{\pm1.6}$	$86.3_{\pm5.2}$

connected form on EM-User due to the requirements of large GPU memory. Nonetheless, all S2N models with GIN and GATv2 outperform SubGNN on all datasets.

Compared to GCNII, which showed the best performance in our paper, GIN and GATv2 generally perform worse. This implies that architectures designed for node or link-level tasks are sub-optimal for subgraph-level tasks. We suggest further studies on model architectures for learning subgraph representations.

K. Ablation Study of Hyperparameters

We conduct ablation studies on the readout method (Equation 10) (sum, mean, max, and degree-dependent), the number of layers in $\texttt{GNN}_{\text{S2N}}$ (0, 1, 2, 4), and the positional encoding (Dwivedi et al., 2022). We report the performance of S2N+0 and S2N+A with GCNII by the readout method in Table 8, the number of layers in Table 9, and the positional encoding in Table 10.

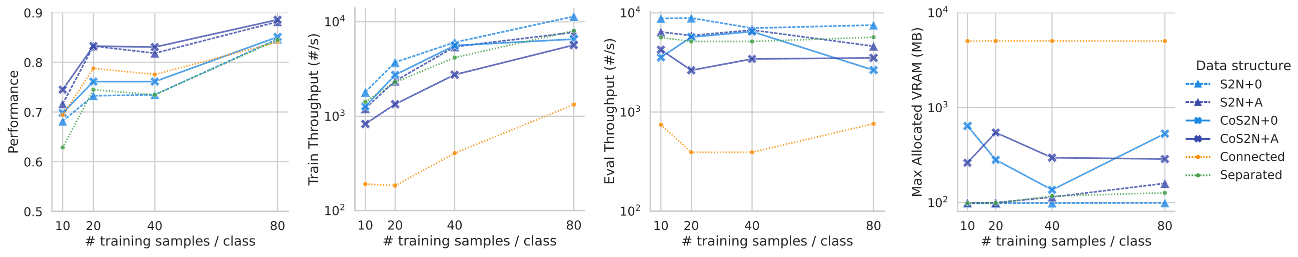
Readout Generally, the sum-readout performs best, and the max-readout performs the worst, as illustrated in Table 8. The performance of mean-readout and degree-dependent readout varies by dataset. In S2N+A, degree-dependent readout performs similarly to sum-readout and slightly outperforms on EM-User.

The Number of Layers In Table 9, we find that using message-passing (i.e., the number of layers > 0) always increases the performance on all datasets. That is, modeling the S2N graph structures helps to learn the representation of subgraphs. The performance improvement by $\texttt{GNN}_{\text{S2N}}$ in S2N+0 is higher than in S2N+A, which leverages internal structures. The performance decreases when we use a deeper $\texttt{GNN}_{\text{S2N}}$ than the optimum; that is, an over-smoothing effect exists in $\texttt{GNN}_{\text{S2N}}$ (Li et al., 2018).

Positional Encoding In Table 10, we report the performance of GCNII models with Random Walk Positional Encoding (RWPE) and Laplacian Positional Encoding (LapPE) (Dwivedi et al., 2022). We find that LapPE also contributes to performance improvement in general, and there is no significant difference between RWPE and LapPE.

Table 10: Mean performance in micro F1-score over 10 runs using GCNII models with different positional encoding.

Data Structure	Positional Encoding	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
S2N+0	None	$63.5_{\pm 2.4}$	$66.4_{\pm 1.1}$	$61.6_{\pm 1.7}$	$86.5_{\pm 3.2}$
	RWPE	$63.5_{\pm 1.7}$	$\mathbf{66.7_{\pm 0.6}}$	$\mathbf{62.3_{\pm 1.1}}$	$86.5_{\pm 4.7}$
	LapPE	$\mathbf{63.9_{\pm 2.4}}$	$66.1_{\pm 0.8}$	$61.9_{\pm 1.7}$	$\mathbf{87.3_{\pm 2.5}}$
S2N+A	None	$63.7_{\pm 2.3}$	$68.4_{\pm 1.0}$	$63.2_{\pm 2.7}$	$\mathbf{89.0_{\pm 1.6}}$
	RWPE	$\mathbf{64.3_{\pm 1.8}}$	$68.6_{\pm 0.8}$	$\mathbf{63.9_{\pm 1.7}}$	$\mathbf{89.0_{\pm 3.1}}$
	LapPE	$64.2_{\pm 1.9}$	$\mathbf{68.8_{\pm 0.9}}$	$63.7_{\pm 1.4}$	$\mathbf{89.0_{\pm 3.2}}$



(a) Performance (b) Training Throughput (c) Eval Throughput (d) Max allocated VRAM

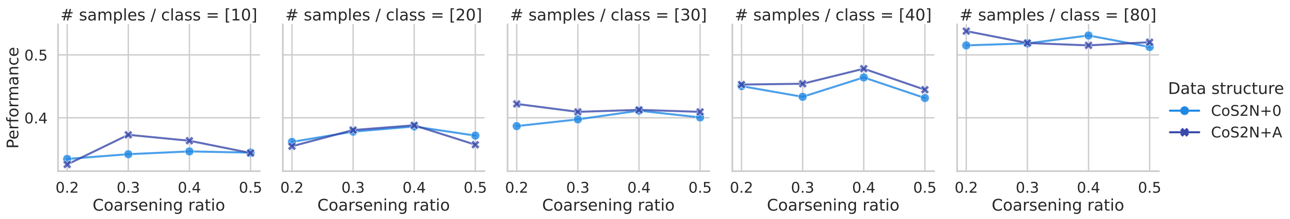
Figure 5: Performance and efficiency on EM-User of S2N, CoS2N, connected, and separated forms by the number of training samples in a data-scarce setting.

L. Performance and Efficiency of Coarsened S2N in a Data-Scarce Setting

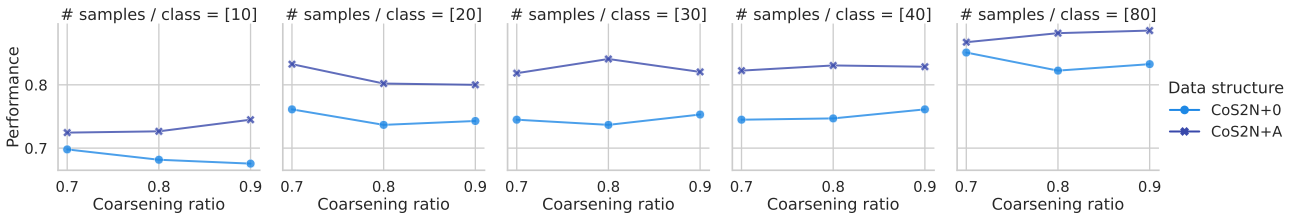
For experiments in a data-scare setting, we narrow the search space of hyperparameters. Specifically, we fix to use batch normalization but not skip-connections. We use a coarsening ratio that creates virtual subgraphs smaller than the average size: $[0.2, 0.3, 0.4, 0.5]$ for PPI-BP and $[0.7, 0.8, 0.9]$ for EM-User. After graph coarsening, we remove subgraphs that consist of a single node. We follow the same tuning procedures in Appendix F for the remaining details.

As stated in §6.4, we summarize performance and efficiency on EM-User in Figure 5. Overall, results on EM-User do not show a notable difference from trends in data-scarce experiments on PPI-BP at §6.4. We can observe that (1) subgraphs created by coarsening contribute to performance improvements of S2N, and (2) CoS2N has higher throughput and uses less memory than using the global graph.

We report the performance on PPI-BP and EM-User by the coarsening ratio in Figure 6. Although there is no consistently optimal coarsening ratio by the number of training samples, we can conclude that finding the optimal coarsening ratio for each dataset can increase the performance.



(a) Performance on PPI-BP by coarsening ratio.



(b) Performance on EM-User by coarsening ratio.

Figure 6: Performance of CoS2N on PPI-BP and EM-User by coarsening ratio.