DeSCo: Towards Generalizable and Scalable Deep Subgraph Counting

Tianyu Fu
Tsinghua University
Beijing, China
fty22@mails.tsinghua.edu.cn

Chiyue Wei
Tsinghua University
Beijing, China
chiyue.wei@duke.edu

Yu Wang*
Tsinghua University
Beijing, China
yu-wang@tsinghua.edu.cn

Rex Ying
Yale University
New Haven, Connecticut, USA
rex.ying@yale.edu

ABSTRACT

Subgraph counting is the problem of counting the occurrences of a given query graph in a large target graph. Large-scale subgraph counting is useful in various domains, such as motif analysis for social network and loop counting for money laundering detection. Recently, to address the exponential runtime complexity of scalable subgraph counting, neural methods are proposed. However, existing approaches fall short in three aspects. Firstly, the subgraph counts vary from zero to millions for different graphs, posing a much larger challenge than regular graph regression tasks. Secondly, current scalable graph neural networks have limited expressive power and fail to efficiently distinguish graphs for count prediction. Furthermore, existing neural approaches cannot predict query occurrence positions.

We introduce DeSCo, a scalable neural deep subgraph counting pipeline, designed to accurately predict both the count and occurrence position of queries on target graphs post single training. Firstly, DeSCo uses a novel *canonical partition* and divides the large target graph into small neighborhood graphs, greatly reducing the count variation while guaranteeing no missing or double-counting. Secondly, *neighborhood counting* uses an expressive subgraph-based heterogeneous graph neural network to accurately count in each neighborhood. Finally, *gossip propagation* propagates neighborhood counts with learnable gates to harness the inductive biases of motif counts. DeSCo is evaluated on eight real-world datasets from various domains. It outperforms state-of-the-art neural methods with 137× improvement in the mean squared error of count prediction, while maintaining the polynomial runtime complexity. Our open-source project is at https://github.com/fuvty/DeSCo.

CCS CONCEPTS

• **Information systems** → **Graph-based database models**.

*Corresponding Author

WSDM '24, March 4–8, 2024, Merida, Mexico

ACM ISBN 979-8-4007-0371-3/24/03
https://doi.org/10.1145/3616855.3635788

KEYWORDS

subgraph counting, graph mining, graph neural network

ACM Reference Format:
Tianyu Fu, Chiyue Wei, Yu Wang, and Rex Ying. 2024. DeSCo: Towards Generalizable and Scalable Deep Subgraph Counting. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3616855.3635788

1 INTRODUCTION

Given a *query* graph and a *target* graph, the problem of subgraph counting is to count the number of *patterns*, defined as subgraphs of the target graph, that are graph-isomorphic to the query graph [63].

Subgraph counting is crucial for domains including biology [1, 6, 8, 69, 73], social science [40, 60, 75, 80], risk management [3, 62], and software analysis [77, 85]. For example, in brain networks, it is used to identify important functional motifs and understand how the brain evolves [71]. In social networks, counts of stars, holes, or paths are used to characterize circles of friends [18].

While being essential in graph and network analysis, subgraph counting is a #P-complete problem [76]. Due to the computational complexity, existing exact counting algorithms are restricted to small query graphs with no more than 5 vertices [2, 56, 59]. The commonly used VF2 [21] algorithm fails to even count a single query of a 5-node chain within a week's time budget on a large target graph Astro [43] with nineteen thousand nodes.

Luckily, approximate counting of query graphs is sufficient in most real-world use cases [38, 41, 64]. Heuristic methods can scale to large targets by substructure sampling, random walk, and color-based sampling, allowing estimation of the frequency of query graph occurrences. However, they still cannot scale to large queries. Very recently, Graph Neural Networks (GNNs) are employed as a deep learning-based approach to scale the query graphs in subgraph counting [20, 45, 93]. The target graph and the query graph are embedded via a GNN, which predicts the motif count through a regression task.

However, there exist several major challenges with existing heuristic and GNN approaches: 1) The number of graph structures and count variation both grow super-exponentially with respect to the graph size [61, 68], resulting in large approximation error [63]. For different large target graphs, the counts of the same query can vary from zero to millions, making the task much harder than most

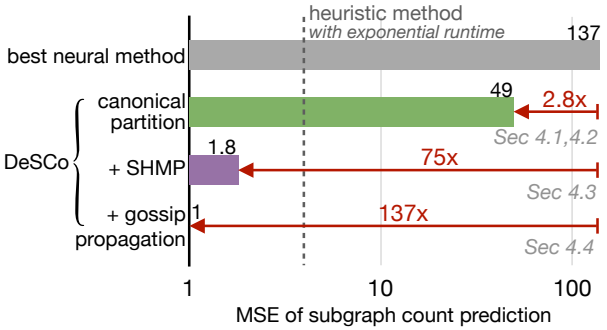

Figure 1: DeSCo Pipeline reduces the mean square error (MSE) of subgraph count prediction with three components: canonical partition, subgraph-based heterogeneous message passing (SHMP) and gossip propagation. The MSE is evaluated and averaged on eight real-world datasets.

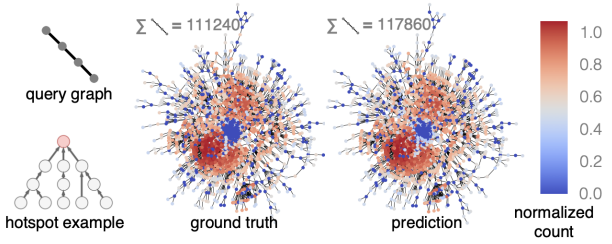

Figure 2: The total count and the position distribution of the query graph over the CiteSeer Citation Network. The figure compares between ground truth and DeSCo predictions. The hotspots are where the 4-chain patterns appear most often in CiteSeer.

graph regression tasks [70], which only predict a single-digit number with a small upper bound. 2) The expressive power of commonly used message passing GNNs is limited by the Weisfeiler-Lehman (WL) test [20, 42, 86]. Certain structures are not distinguishable with these GNNs, let alone counting them, resulting in the same count prediction for different queries. 3) Furthermore, most existing approximate heuristic and GNN methods only focus on estimating the total count of a query in the target graph [15, 16, 20, 45], but not the occurrence positions of the patterns, as shown in Figure 2. Yet such position distribution information is crucial in various applications [10, 26, 35, 74, 90].

Proposed work. To resolve the above challenges, we propose DeSCo, a GNN-based model that learns to predict both pattern counts and occurrence positions on any target graph. The main idea of DeSCo is to leverage the local information of neighborhood patterns to predict query counts and occurrences in the entire target graph. DeSCo first uses *canonical partition* to decompose the target graph into small neighborhoods. The local information is then encoded using a GNN with *subgraph-based heterogeneous message passing*. Finally, we perform *gossip propagation* to use inductive biases to improve counting accuracy over the entire graph. Our contributions are four-fold.

Canonical partition. Firstly, we propose *canonical partition* that divides the problem into subgraph counting for individual neighborhoods. We theoretically prove that no pattern will be double counted or missed for all neighborhoods. The algorithm allows the model to make accurate predictions on large target graphs with high count variation. Furthermore, we can predict the pattern position distribution for the first time, as shown in Figure 2. In this citation network, the hotspots represent overlapped linear citation chains, indicating original publications that motivate multiple future directions of incremental contributions [30, 89], which shed light on the research impact of works in this network.

Subgraph-based heterogeneous message passing. Secondly, we propose a general approach to enhance the expressive power of any MPGNNs by encoding the subgraph structure through heterogeneous message passing. The message type is determined by whether the edge presents in a certain subgraph, e.g., a triangle. We show that this architecture outperforms expressive GNNs, including GIN [86] and ID-GNN [92], while maintaining the polynomial runtime complexity for scalable subgraph counting.

Gossip propagation. We further improve the count prediction accuracy by utilizing two inductive biases of the counting problem: homophily and antisymmetry. Real-world graphs share similar patterns among adjacent nodes, as shown in Figure 2. Furthermore, since canonical count depends on node indices, there exists antisymmetry due to canonical partition. Therefore, we propose a *gossip propagation* phase featuring a learnable gate for propagation to leverage the inductive biases.

Generalization Framework. We propose a generalization framework that uses the carefully designed synthetic dataset to enable model generalization to different real-world datasets. After training on the synthetic dataset, the model can directly perform subgraph counting inference with high accuracy on real-world datasets.

To demonstrate the effectiveness of DeSCo, we compare it against state-of-the-art GNN-based subgraph counting methods [20, 45, 46], as well as approximate heuristic method [15, 16] on eight real-world datasets from various domains. DeSCo achieves 137× mean square error reduction of count predictions for both small and large targets, as shown in Figure 1. To the best of our knowledge, it is also the first approximate method to accurately predict pattern position distribution as illustrated in Figure 2. DeSCo also maintains polynomial runtime efficiency, demonstrating orders of magnitude speedup over the heuristic [15, 16] and exact methods [21, 72].

2 RELATED WORKS

There has been extensive lines of work for subgraph counting.

Exact counting algorithms. Exact methods generally count subgraphs by searching through all possible node combinations and finding the matching pattern. Early methods usually focus on improving the matching phase [21, 51, 83] Recent approaches emphasize the importance of pruning the search space and avoiding double counting [23, 48, 49, 67], which inspires the design our canonical count objective (Section 4.1). However, exact methods still scale poorly in terms of query size (often no more than five nodes) despite much effort [19, 59].

Approximate heuristic methods. To further scale up the counting problem, approximate counting algorithms sample from the target graph to estimate pattern counts. Strategies like path sampling [39, 79], random walk [66, 88], substructure sampling [29, 38],

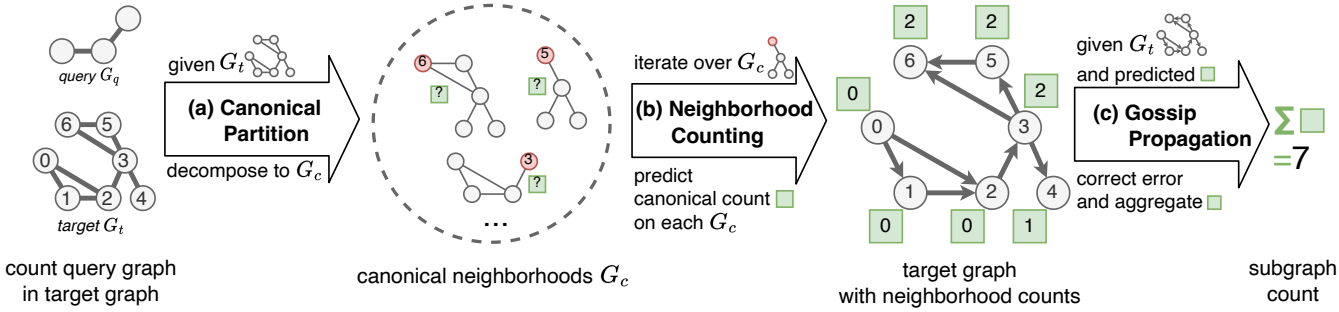

Figure 3: DeSCo Pipeline in 3 steps. (a) Step 1. Canonical Partition: Given *query* and *target*, decompose *target* into multiple node-induced subgraphs, i.e., *canonical neighborhoods*, based on node indices. Each neighborhood contains a *canonical node* that has the greatest index in the neighborhood. (b) Step 2. Neighborhood Counting: Predict the *canonical counts* of each neighborhood via an expressive GNN, and assign the count of the neighborhood to the corresponding *canonical node*. Neighborhood counting is the local count of queries. (c) Step 3. Gossip Propagation: Use GNN prediction results to estimate *canonical counts* on the *target* graph through learnable gates.

and color coding [14–16] are used to narrow the sample space and provides better error bound. However, large and rare queries are still hard to find in the vast sample space, leading to large approximation error [15, 16].

GNN-based approaches. Recently, GNNs have been used to attempt counting large queries. [45, 91] use GNNs to embed the query and target graph, and predict subgraph counts via embeddings. [20] theoretically analyzes the expressive power of GNNs for counting and proposes an expressive GNN architecture. [93] proposes an active learning scheme for the problem. [46] proposes expensive edge-to-vertex dual graph transformation to enhance the model expressive power for subgraph counting. Unfortunately, large target graphs have extremely complex structures and a high variation of pattern count, so accurate prediction remains challenging.

3 PRELIMINARY

Let $G_t = (V_t, E_t)$ be a large *target* graph with vertices V_t and edges E_t. Let $G_q = (V_q, E_q)$ be the *query* graph of interest. The *subgraph counting problem* $C(G_q, G_t)$ is to calculate the size of the *set of patterns* $\mathcal{P} = \{G_p | G_p \subseteq G_t\}$ in the target graph G_t that are isomorphic to the query graph G_q, that is, $\exists$ bijection $f : V_p \mapsto V_q$ such that $(f(v), f(u)) \in E_q$ if and only if $(v, u) \in E_p$, denoted as $G_p \cong G_q$.

Subgraph counting can be categorized into induced and non-induced counting [63]. A subgraph $G_p = (V_p, E_p)$ of G_t is an induced subgraph if it satisfies two conditions: $V_p \subseteq V_t$ and for any two vertices $u, v \in V_p$, they are adjacent in G_p if and only if they are adjacent in G_t. This relationship is denoted as $G_p \subseteq G_t$. Without loss of generality, we focus on the connected, induced subgraph counting problem, following modern mainstream graph processing frameworks [31, 58] and real-world applications [51, 84]. It is also possible to obtain non-induced occurrences from induced ones with a transformation [28]. Our GNN approach can natively support graphs with node features and edge directions. But in alignment with exact and heuristic methods, we use undirected graphs without node features in experiments to investigate the ability to capture graph topology.

4 DESCO PIPELINE

In this section, we introduce the pipeline of DeSCo as shown in Figure 3. To perform subgraph counting, DeSCo first performs **canonical partition** to decompose the target graph to many canonical neighborhood graphs. Then, **neighborhood counting** uses the subgraph-based heterogeneous GNN to embed the query and neighborhood graphs and performs a regression task to predict the canonical count on each neighborhood. Finally, **gossip propagation** propagates neighborhood count predictions over the target graph with learnable gates to further improve counting accuracy. We will first introduce the model objective before elaborating on each step.

4.1 Canonical Count Objective

Motivation. For commonly seen node-level tasks such as node classification, each node is responsible for predicting its own node value. However, for subgraph counting, since each pattern contains multiple nodes, it is unclear which node should be responsible for predicting the pattern's occurrence. As illustrated in Figure 4, the ambiguity can lead to missing or double-counting of the motif, especially for queries with symmetric nodes, e.g. triangle. So we propose the canonical count objective to eliminate the ambiguity by assigning a specific canonical node responsible for each pattern. The canonical node is used to represent the pattern position. The canonical count is used as the local count prediction objective for the GNN and gossip propagation.

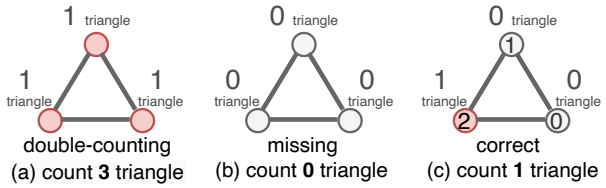

Figure 4: When counting, (a) double-counts and (b) misses the triangle in the neighborhoods due to symmetry. (c) DeSCo uses the canonical node to break symmetry and correctly count the triangle. ⓘ are the node indices.

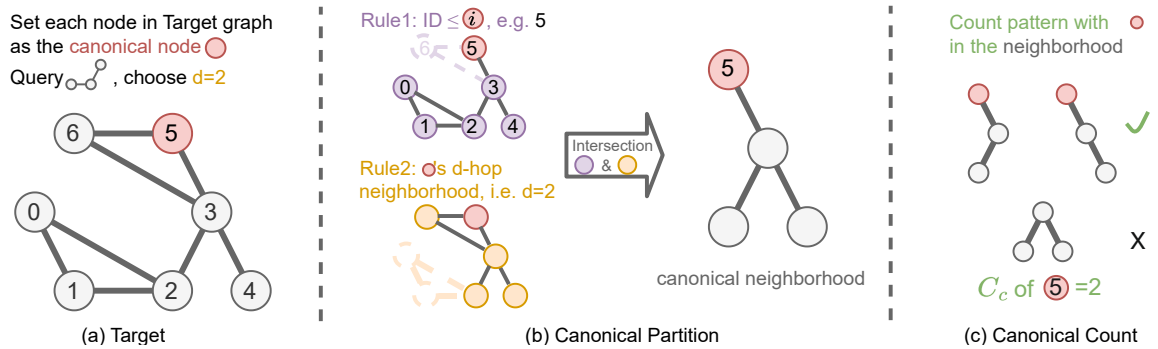

Figure 5: An example of canonical partition and canonical count. (a) Choose node 5 from the target graph as the *canonical node* (red circle). (b) *Canonical partition* generates the corresponding *canonical neighborhood* graph. It performs an ID-restricted breadth-first search to find the induced neighborhood that complies with both Rule 1 and Rule 2. (c) The corresponding *canonical count* is defined by the number of patterns containing the canonical node in the canonical neighborhood. DeSCo's *neighborhood counting* phase predicts the canonical count for each canonical neighborhood.

To break the symmetry, we randomly assign node indices on the target graph and define the *canonical node*.

Definition 4.1 (canonical node). Canonical node v_c *is the node with the largest node index in the pattern.*

$$v_c = \max_I V_p \tag{1}$$

Based on the index, we assign the count of the k-node pattern to its *canonical node* and define *canonical count*.

Definition 4.2 (canonical count). Canonical count C_c *equals the number of patterns that share the same canonical node.*

$$C_c(G_q, G_t, v_c) = |\{G_p \subseteq G_t | G_p \cong G_q, v_c = \max_I V_p\}| \tag{2}$$

The canonical count $C_c(G_q, G_t, v_c)$ differs from the regular count C_c, as it takes an additional variable - a node v_c from the target graph. As shown in Figure 4(c), a pattern is only counted by its canonical node in C_c. So the summation of C_c over all nodes equals the count of all patterns, C, as stated in Lemma 4.1 and proven in Appendix A.1.

Lemma 4.1. *The subgraph count C of query in target equals the summation of the canonical count of query in target for all target nodes.*

$$C(G_q, G_t) = \sum_{v_c \in V_t} C_c(G_q, G_t, v_c) \tag{3}$$

Advantage. By predicting the canonical count of each node, DeSCo can naturally get the pattern position distribution.

Lemma 4.1 allows the decomposition of the counting problem into multiple canonical count objectives. We use the following canonical partition to minimize the overhead for the decomposition.

4.2 Canonical Partition

Motivation. In Lemma 4.1, each canonical count C_c is obtained with the entire target graph G_t. In order to overcome the high computational complexity, we partition the target to reduce the graph size for the canonical count. We observe that each canonical count only depends on some local neighborhood structure as shown in Figure 5(c). So we propose *canonical partition* to efficiently get the small neighborhood.

Unique challenges of partition for canonical count. Commonly used graph partition strategies include cutting edges [5] and taking d-hop neighborhoods [32]. However, edge-cutting breaks the pattern structure, leading to incorrect count; D-hop neighborhoods guarantee correctness, yet are unnecessarily large since patterns exist in many overlapping neighborhoods.

Thus, we define *canonical partition*. It neglects the neighborhood structure that does not influence the canonical count of each node. Canonical partition uses node indices to filter nodes as illustrated in Figure 5(a), (b).

Definition 4.3 (canonical partition). Canonical partition $\mathcal{P}$ *crops the index-restricted d-hop neighborhood around the center node from the target graph.* $\mathcal{D}(G_t, v_i, v_c)$ *means the shortest distance between* v_i *and* v_c *on* G_t.

$$\begin{aligned} &\mathcal{P}(G_t, v_c, d) = G_c, \\ &\text{s. t. } G_c \subseteq G_t, V_c = \{v_i \in V_t | \mathcal{D}(G_t, v_i, v_c) \leq d, v_i \leq v_c\} \end{aligned} \tag{4}$$

The graph G_c obtained by canonical partition is called the *canonical neighborhood*. Canonical neighborhoods can correctly substitute the target graph in canonical count as proven in Appendix A.2. Thus, we derive Theorem 1.

Theorem 1. *The subgraph count of query in target equals the summation of the canonical count of query in canonical neighborhoods for all target nodes. Canonical neighborhoods are acquired with canonical partition $\mathcal{P}$, given any d greater than the diameter of the query.*

$$\begin{aligned} C(G_q, G_t) = \sum_{v_c \in V_t} C_c(G_q, \mathcal{P}(G_t, v_c, d), v_c), \\ d \geq \max_{v_i, v_j \in V_q} \mathcal{D}(G_q, v_i, v_j) \end{aligned} \tag{5}$$

In DeSCo, given the target graph G_t, it iterates over all nodes v of the target G_t and divides it into a set of canonical neighborhoods G_{v_c} with **canonical partition**. In practice, we set d as the maximum

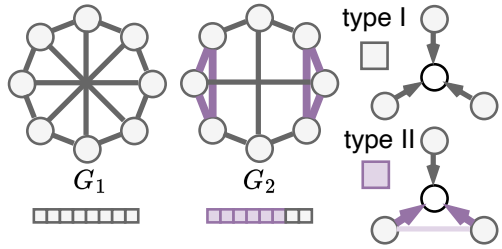

Figure 6: Proposed SHMP. Embedded with regular MP, graphs G_1 and G_2 are indistinguishable. While embedded with SHMP, G_2 is successfully distinguished with six type II node embeddings, demonstrating better expressive power of SHMP.

diameter of query graphs to meet the requirements of Theorem.1. See Appendix A.3 for the implementation of $\mathcal{P}(G_t, v_c, d)$.

Advantage. Canonical partition dramatically reduces the worst and average complexity of the subgraph counting problem by a factor of $1/10^{70}$ and $1/10^{11}$, thanks to the sparse nature of real-world graphs (discussed in Appendix A.4). Furthermore, diverse target graphs can have similar and limited kinds of canonical neighborhoods. So it boosts the generalization power of DeSCo as shown in Section 5.4.

This divide-and-conquer scheme not only greatly reduces the complexity of each GNN prediction, but also makes it possible to predict the count distribution over the entire graph. After the canonical partition, DeSCo uses the following model to predict the canonical count for each decomposed neighborhood.

4.3 Neighborhood Counting

After canonical partition, GNNs are used to predict the *canonical count* $C_c(G_q, G_{v_c}, v_c)$ on any canonical neighborhood G_{v_c} in the **neighborhood counting** stage. The canonical neighborhood and the query are separately embedded using GNNs. The embeddings are passed to a multilayer perceptron to predict the canonical count.

Motivation. Previous work [20] shows message passing (MP) GNNs confuse certain graph structures and harm the counting accuracy. To enhance GNN's expressive power while remaining scalable, we propose the Subgraph-based Heterogeneous Message Passing (SHMP) framework. Inspired by [52], SHMP incorporates subgraph information to boost the expressive power. In the meantime, SHMP avoids using super-node [52] or message permutation [20] that are computationally expensive during message passing.

Neighborhood counting with SHMP. To embed the input graph, SHMP uses small subgraph structures to categorize edges into different edge types, and uses different learnable weights for each edge type.

Definition 4.4 (subgraph-based heterogeneous message passing). *The SHMP computes each node's representation with equation 6. Here k denotes the layer; γ denotes the update function; ϕ_h^k denotes the message function of the h-th edge type; $N_h(i)$ denotes nodes that connect to node i with the h-th edge type; AGG and AGG′ are the permutation invariant aggregation function such as summation.*

$$\begin{aligned} \mathbf{x}_i^{(k)} &= \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, AGG'_{h\in H}(M_h)\right) \\ M_h &= AGG_{j\in N_h(i)}\left(\phi_h^{(k)}(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i})\right) \end{aligned} \tag{6}$$

Note that MP defined by major GNN frameworks [27, 78] is just a special case of SHMP if only one edge type is derived with the subgraph structure. We prove that SHMP can exceed the upper bound of MP in terms of expressiveness in Appendix B.1.

For example, Figure 6 demonstrates that triangle-based heterogeneous message passing has better expressive power. Regular MPGNNs fail to distinguish different d-regular graphs G_1 and G_2 because of their identical type I messages and embeddings, which is a common problem of MPGNNs [92]. SHMP, however, can discriminate the two graphs by giving different embeddings. The edges are first categorized into two edge types based on whether they exist in any triangles (edges are colored purple if they exist in any triangles). Since no triangles exist in G_2, all of its nodes still receive type I messages. While some nodes of G_1 now receive type II messages with two purple messages and one gray message in each layer. As a result, the model acquires not only the adjacency information between the message sender and receiver, but also information among their neighbors. Such subgraph structural information improves expressiveness by incorporating high-order information in both the query and the target. In DeSCo, the canonical node of the neighborhood is also treated as a special node type in the heterogeneous message passing.

Advantage. The triangle-based SHMP reduces the typical error of MPGNNs by 68% as discussed in Appendix B.2, while remaining polynomial runtime complexity of $O(V + E^{3/2})$ as discussed in Appendix F. The comparison with other expressive GNNs are shown in Table 7 and Appendix B.3.

The summation of the neighborhood counts (the predicted canonical counts of all canonical neighborhoods) can serve as the final subgraph count prediction. The counts also show the position of patterns. But to further improve counting accuracy, we pass the neighborhood counts to the *gossip propagation* stage.

4.4 Gossip Propagation

Given the count predictions $\hat{C}_c$ output by the GNN, DeSCo uses **gossip propagation** to improve the prediction quality, enforcing different homophily and antisymmetry inductive biases for different queries. Gossip propagation uses another GNN to model the error of neighborhood count. It uses the predicted $\hat{C}_c$ as input, and the canonical counts C_c as the supervision for corresponding nodes in the target graph.

Motivation. To further improve the counting accuracy, we identify two inductive biases: *Homophily* and *Antisymmetry*. 1) *Homophily*: Adjacent nodes within graphs share similar graph structures, resulting in analogous canonical counts (Figure 2). This phenomenon, termed *homophily* of canonical counts, stands out. 2) *Antisymmetry*: Nodes with similar neighborhood structures, per Definition 4.2, exhibit higher canonical counts for those with larger node indices. See right part of Figure 3 for an example. Details are in Appendix C.

We observe a negative correlation between *Antisymmetry* ratio and *Homophily* in different queries, as depicted in Figure 14 in Appendix C. This observation inspires us to learn this relationship within models.

The edges' direction in message passing can control the *homophily* and *antisymmetry* properties of the graph. With undirected

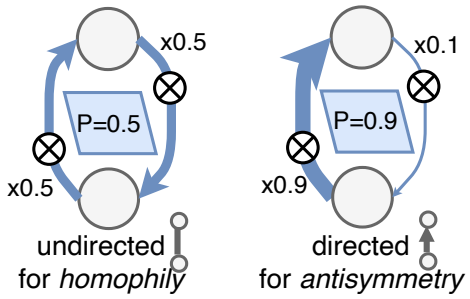

Figure 7: Proposed learnable gates in the gossip propagation model balance the influence of *homophily* and *antisymmetry* by controlling message directions.

edges, message propagation is a special low-pass filter [55], enhancing the homophily property of the node values. With directed edges pointing from small-index nodes to large-index nodes, message propagation accumulates value in large-index nodes, which enhances the antisymmetry property.

Gossip propagation with learnable gates. To learn the edge direction that correctly emphasizes homophily or antisymmetry, we propose the gossip propagation model as shown in Figure 7. It multiplies a learnable gate P for the message sent from the node with the smaller index, and $1-P$ for the reversed one. P is learned from the query embedding. For different queries, P ranges from 0 to 1 to balance the influence of *homophily* and *antisymmetry*. When $P \to 0.5$, messages from the smaller indexed node and the reversed one are weighed equally. So it simulates undirected message passing that stress *homophily* by taking the average of adjacent node values. When the gate value moves away from 0.5, the message from one end of the edge is strengthened. For example, when $P \to 1$, the node values only accumulate from nodes with smaller indices to nodes with larger ones. So that it simulates directed message passing that stress *antisymmetry* of the transitive partial order of node indices.

The messages of MPGNNs are multiplied with g_{ji} on both edge directions. With learnable gates, the model can balance the effects of homophily and antisymmetry for further performance improvement.

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \text{AGG}_{j\in N(i)} g_{ji}\cdot\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)\right)$$
$$g_{ji} = \begin{cases} P & v_j \le v_i \\ 1-P & v_j > v_i \end{cases} \tag{7}$$

Final count prediction. The neighborhood count with gossip propagation is a more accurate estimation of the canonical count. The summation of the neighborhood counts is the unbiased estimation of subgraph count on the whole target graph as Theorem 1 states.

5 EXPERIMENTS

We compare the performance of DeSCo with state-of-the-art neural subgraph counting methods, as well as the approximate heuristic method. Our evaluation showcases the scalability and generalization capabilities of DeSCo across diverse and larger target datasets, contrasting with prior neural methods that mostly focused on smaller datasets. We also demonstrate the runtime advantage of DeSCo compared to recent exact and approximate heuristic methods. Extensive ablation studies further show the benefit of each component of DeSCo.

Dataset	#graphs	Avg. #nodes	Avg. #edges
SYNTHETIC	1827	134.91	381.58
MUTAG	188	17.93	19.79
COX2	467	41.22	43.45
ENZYMES	600	32.63	62.14
IMDB-BINARY	1000	19.77	96.53
MSRC-21	563	77.52	198.32
FIRSTMM-DB	41	1.3K	3.0K
CITESEER	1	3.3K	4.5K
CORA	1	2.7K	5.4K

Table 1: Graph statistics of datasets used in experiments.

5.1 Experimental Setup

Datasets. Compared with previous neural methods, our evaluation extends to larger datasets across various domains, such as chemistry (MUTAG [22], COX2 [65]), biology (ENZYMES [13]), social networks (IMDB-BINARY [87]), computer vision (MSRC-21, FIRSTMM-DB [53]), and citation networks (CiteSeer, Cora [50]). A synthetic dataset, representing mixed graph characteristics, is also included (Table 1). Additional dataset details are in Appendix D.

Generalization framework. Our framework, trained on the Synthetic dataset with standard queries (size $3-5$), enables subgraph counting across diverse datasets and graph.

Baselines. DeSCo is compared with SOTA subgraph counting GNNs: LRP [20], DIAMNet [45], DMPNN [46], the heuristic MOTIVO [15], and exact methods VF2 [21] and IMSM [72]. Optimal configurations for each method are detailed in Appendix D.4 and F.

Evaluation metric. Evaluation utilizes mean square error (MSE) and mean absolute error (MAE) for subgraph count predictions, with MSE normalized by ground truth variance [20].

5.2 Neural Counting

Subgraph counting. Table 2 highlights DeSCo's performance in subgraph count prediction across twenty-nine standard queries of size $3-5$. It outperforms the best neural baseline and approximate heuristic method in normalized MSE by 49.7× and 17.5×, and in MAE by 8.4× and 4.1× respectively. The model shows robust performance even on dense graphs which is challenging for neural method, like IMDB-BINARY. Unlike the heuristic method with exponential complexity, DeSCo maintains linear runtime efficiency. Additional q-error metric analysis is in Appendix G.1.

Position distribution. DeSCo innovates in pattern position prediction, achieving 3.8×10^{-3} normalized MSE, further detailed in Appendix E.2.

5.3 Scalability

Large queries. We analyze 16 frequently appearing queries for sizes 6 to 13 from ENZYMES (details in Appendix D.2). All models, except DeSCo (zero-shot), are fine-tuned on larger queries using

Dataset	MUTAG			COX2			ENZYMES			IMDB-BINARY			MSRC-21		
Query-Size	3	4	5	3	4	5	3	4	5	3	4	5	3	4	5
normalized MSE															
MOTIVO	2.9E-1	6.7E-1	1.2E+0	1.6E-1	3.4E-1	5.9E-1	1.6E-1	1.9E-1	3.0E-1	2.7E-2	**3.9E-2**	**5.0E-2**	4.8E-2	7.2E-2	9.5E-2
LRP	1.5E-1	2.7E-1	3.5E-1	1.4E-1	2.9E-2	1.1E-1	8.5E-1	5.4E-1	6.2E-1	inf	inf	inf	2.4E+0	1.4E+0	1.1E+0
DIAMNet	4.1E-1	5.6E-1	4.7E-1	1.1E+0	7.8E-1	7.2E-1	1.4E+0	1.1E+0	1.0E+0	1.1E+0	1.0E+0	1.0E+0	2.7E+0	1.6E+0	1.3E+0
DMPNN	6.1E+2	6.6E+2	3.0E+2	2.6E+3	2.4E+3	3.0E+3	2.9E+3	1.4E+3	1.2E+3	2.1E+4	1.3E+2	1.4E+2	1.1E+4	1.3E+3	4.1E+2
DeSCo	**2.2E-3**	**7.5E-4**	**6.0E-3**	**6.6E-4**	**6.3E-4**	**4.9E-3**	**5.4E-3**	**5.9E-2**	**5.3E-2**	**8.5E-3**	2.1E-1	4.5E-1	**2.5E-3**	**3.8E-3**	**8.7E-2**
MAE															
MOTIVO	4.9E+0	5.1E+0	3.3E+0	8.3E+0	9.4E+0	7.3E+0	1.7E+1	2.3E+1	2.6E+1	4.7E+1	**1.6E+2**	**6.1E+2**	4.1E+1	9.5E+1	1.7E+2
LRP	3.8E+0	5.1E+0	4.5E+0	9.5E+0	4.0E+0	6.3E+0	4.3E+1	4.0E+1	3.7E+1	inf	inf	inf	3.2E+2	4.6E+2	5.9E+2
DIAMNet	8.3E+0	7.9E+0	4.2E+0	3.0E+1	1.7E+1	1.2E+1	5.4E+1	5.1E+1	4.0E+1	2.9E+2	8.3E+2	2.6E+3	3.4E+2	4.9E+2	6.3E+2
DMPNN	6.8E+2	6.9E+2	2.4E+2	3.6E+3	4.3E+3	3.8E+3	4.8E+3	5.8E+3	6.0E+3	1.7E+5	2.2E+5	2.8E+5	3.4E+4	4.6E+4	5.7E+4
DeSCo	**5.0E-1**	**1.8E-1**	**2.9E-1**	**6.1E-1**	**4.4E-1**	**7.7E-1**	**3.6E+0**	**1.1E+1**	**9.9E+0**	**2.4E+1**	3.0E+2	1.6E+3	**1.0E+1**	**2.5E+1**	**1.3E+2**

Table 2: Normalized MSE and MAE performance of approximate heuristic and neural methods on subgraph counting of twenty-nine standard queries.

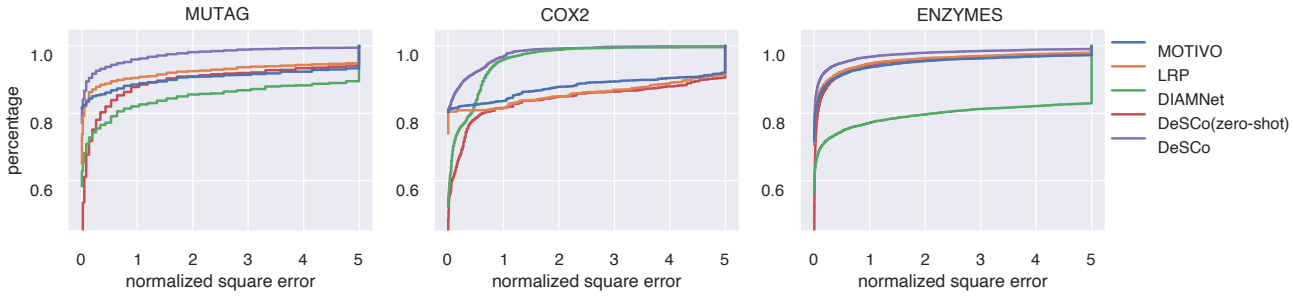

Figure 8: The accumulative distributions of normalized square error of large queries (size up to 13) on three target datasets. The x-axis is clipped at 5. Given any square error tolerance bound (x-axis), DeSCo has the highest percentage of predictions that meet the bound (y-axis). DeSCo(zero-shot) generalizes to unseen queries with competitive performance over specifically trained baselines.

the synthetic dataset. DeSCo (zero-shot) demonstrates its capability to generalize to unseen queries. The square error distribution for each query-target pair is in Figure 8, with numeric results in Appendix G.2.

Large target. In testing on large target graphs (Table 3), DeSCo surpasses other neural methods, handling up to 3.8×10^6 and 3.3×10^7 ground truth counts on CiteSeer and Cora, respectively. LRP's results, being infinite, are excluded from the table.

5.4 Generalization Ability

Synthetic Dataset. Using the Synthetic dataset, we showcase DeSCo's generalization. Real-world graphs' diversity in structure (Figure 9 (a)) contrasts with their local substructure similarities (Figure 9 (b)). The synthetic dataset's coverage of real-world graph characteristics (Figure 9 (c)) confirms DeSCo's training effectiveness and generalizability.

Generalization. DeSCo, pre-trained on the Synthetic dataset and tested on varied real-world datasets, demonstrates superior accuracy and generalization compared to models trained on existing datasets (Table 4). This underscores its robustness across different domains.

Dataset	CiteSeer			Cora		
Query-Size	3	4	5	3	4	5
normalized MSE						
DIAMNet	2.0E+0	1.5E+0	1.2E+0	1.0E+10	3.2E+7	3.7E+4
DMPNN	9.5E+4	2.5E+2	6.8E+1	1.8E+5	1.1E+2	6.7E+1
DeSCo	**3.5E-5**	**9.7E-2**	**1.6E-1**	**4.2E-3**	**2.1E-1**	**6.3E-2**
MAE						
DIAMNet	1.1E+4	6.0E+4	3.6E+05	2.1E+9	1.6E+9	8.3E+8
DMPNN	6.1E+6	7.6E+6	8.7E+6	1.8E+7	2.4E+7	3.0E+7
DeSCo	**6.0E+1**	**1.2E+4**	**1.1E+5**	**1.3E+3**	**7.3E+4**	**5.4E+5**

Table 3: Normalized MSE and MAE performance of neural methods on large targets with standard queries.

5.5 Ablation Study

In assessing DeSCo's components, the ablation study reveals significant contributions of each part. We demonstrate the MAE results on three datasets (Figure 10) and the geometric mean of normalized MSE on eight datasets (Figure 1), supported by numeric data in Appendix E.

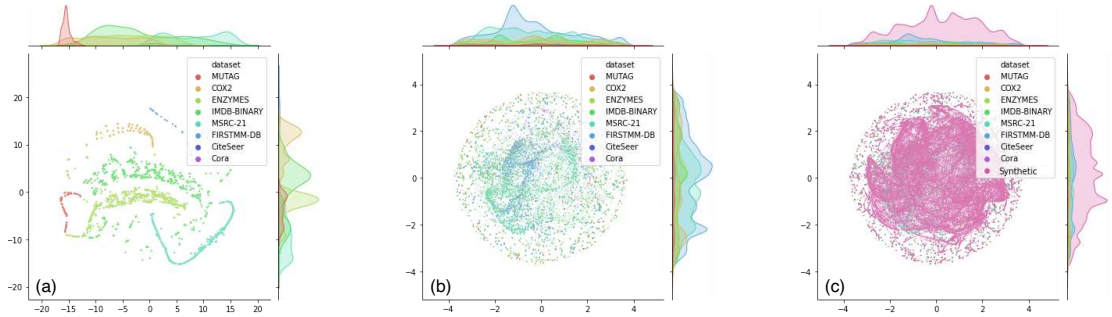

Figure 9: Visualization of statistics of diverse graph datasets. The embedding is obtained by projecting the vectors of graph statistics via t-SNE. (a) Each point represents a graph. (b) Each point represents a canonical neighborhood. (c) Canonical neighborhoods of the synthetic dataset cover most canonical neighborhoods of real-world graphs in terms of data distribution.

Test-Set	MUTAG			MSRC-21			FIRSTMM-DB		
Query-Size	3	4	5	3	4	5	3	4	5
Existing	6.5E-3	3.4E-3	8.7E-2	1.1E+1	1.9E+0	1.1E+0	1.1E-1	1.1E-1	1.6E-1
Synthetic	**2.3E-3**	**8.4E-4**	**6.5E-3**	**2.5E-3**	**3.8E-3**	**8.7E-2**	**2.1E-3**	**3.6E-2**	**5.4E-2**

Table 4: Normalized MSE performance with different training datasets. When pre-training on existing datasets, MSRC-21 uses MUTAG; CiteSeer uses Cora; FIRSTMM-DB uses CiteSeer.

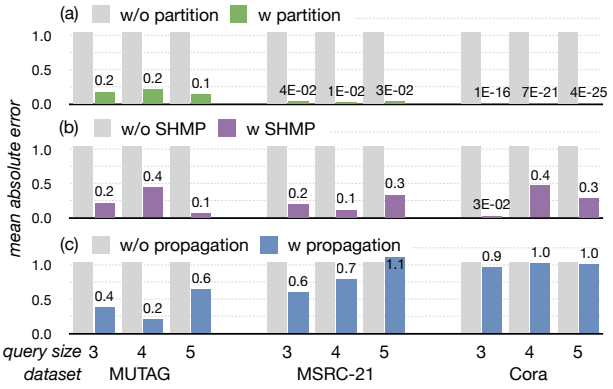

Figure 10: MAE performance with and without canonical partition, SHMP and gossip propagation.

Ablation of canonical partition. Removing the canonical partition and training DeSCo for subgraph count on whole targets (like other neural baselines) indicates the partition's vital role in error reduction and DeSCo's superiority over existing neural methods (Figure 1). Canonical partition in DeSCo brings a $1.3 * 10^{10}\times$ improvement in normalized MSE and $8.8 * 10^{4}\times$ in MAE.

Ablation of SHMP. SHMP enhances GraphSAGE's performance by transitioning to heterogeneous message passing, using triangles as the categorizing subgraph (Figure 6). SHMP reduces the normalized MSE by 27× and MAE by 5.8× over GraphSAGE. Further more, when compared with expressive GNNs, including GIN and ID-GNN, SHMP demonstrate a 24× and 14× reduction in normalized MSE, as well as a 5.3×, 3.9× reduction MAE, as detailed in Table 7.

Ablation of gossip propagation. Comparing direct summation of neighborhood counts with summation post-gossip propagation highlights its effectiveness. Gossip propagation further reduces normalized MSE and MAE by 1.8× and 1.4×, respectively.

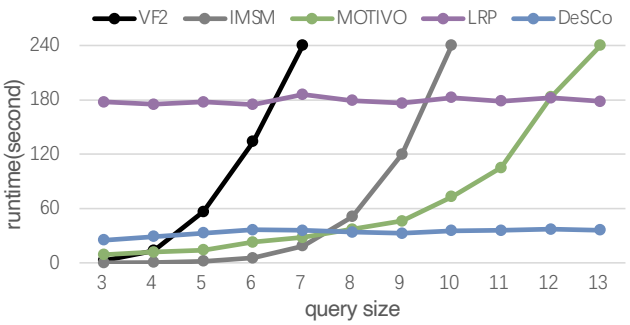

Figure 11: The runtime comparison between exact, heuristic approximate, neural methods and DeSCo. All tested on the ENZYMES dataset.

5.6 Runtime Comparison

Figure 11 illustrates the runtime of each method under a four-minute limit. Exact methods VF2 and IMSM exhibit exponential runtime increases due to the #P hard nature of subgraph counting. For the approximate heuristic method MOTIVO, exponential growth mainly stems from its coloring phase. In contrast, neural methods LRP and DeSCo show polynomial scalability. DeSCo achieves a 5.3× speedup over LRP, as it avoids heavy node feature permutations. Further runtime analysis is available in Appendix F.

6 CONCLUSION

We propose DeSCo, a neural network based pipeline for generalizable and scalable subgraph counting. With canonical partition, subgraph-based heterogeneous message passing, and gossip propagation, DeSCo accurately and efficiently predicts counts for both large queries and targets. It demonstrates magnitudes of improvements in mean square error. It additionally provides the important position distribution of patterns that previous works cannot.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No. U19B2019, 62325405, U21B2031, 61832007, 62104128, 62204164), and Beijing National Research Center for Information Science and Technology (BNRist), and Tsinghua-Meituan Joint Institute for Digital Life.

ETHICAL CONSIDERATIONS

In the realm of graph analysis, DeSCo stands as a fundamental tool rather than a specific application-driven solution. While the direct potential for DeSCo to induce negative societal impacts is minimal, it remains prudent to acknowledge and address potential adverse outcomes.

Accuracy. Similar to other non-exact counting methods, DeSCo cannot ensure absolute prediction correctness. Despite thorough testing on extensive real-world datasets, which has showcased significant error reductions and exceptional generalization capabilities, the potential for inaccurate predictions, especially for outlier graphs, remains a possibility. Therefore, it's advisable to exercise caution and validate basic graph statistics, such as maximum degree, before applying the DeSCo method.

Privacy. DeSCo introduces a breakthrough in accurately counting large subgraphs, previously unattainable. Moreover, it reveals the positional distribution of these counts. As subgraph counting finds applications in recommendation systems, social network analysis, and other domains, there's potential for corporations and governments to glean intelligence that was once inaccessible. This advancement could inadvertently compromise user privacy if not subjected to proper oversight. To mitigate this, it's essential to consider enforcing relevant regulations should corresponding technologies be developed.

REFERENCES

[1] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. 2006. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics* 22, 8 (2006), 1021–1023.

[2] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. 2015. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining*. IEEE, 1–10.

[3] Leman Akoglu and Christos Faloutsos. 2013. Anomaly, event, and fraud detection in large network datasets. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 773–774.

[4] Réka Albert and Albert-László Barabási. 2000. Topology of evolving networks: local events and universality. *Physical review letters* 85, 24 (2000), 5234.

[5] David A Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 2013. *Graph partitioning and graph clustering*. Vol. 588. American Mathematical Society Providence, RI.

[6] Gary D. Bader and Christopher W. V. Hogue. 2003. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* 4, 1 (2003), 2–2.

[7] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[8] Jordi Bascompte and Carlos J Melián. 2005. Simple trophic modules for complex food webs. *Ecology* 86, 11 (2005), 2868–2873.

[9] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 1–4.

[10] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[11] Bibek Bhattarai, Hang Liu, and H Howie Huang. 2019. Ceci: Compact embedding cluster index for scalable subgraph matching. In *Proceedings of the 2019 International Conference on Management of Data*. 1447–1462.

[12] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. 2013. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics* 14, 7 (2013), 1–13.

[13] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47–i56.

[14] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2018. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 4 (2018), 1–25.

[15] Marco Bressan, Stefano Leucci, and Alessandro Panconesi. 2019. Motivo: fast motif counting via succinct color coding and adaptive sampling. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1651–1663.

[16] Marco Bressan, Stefano Leucci, and Alessandro Panconesi. 2021. Faster motif counting via succinct color coding and adaptive sampling. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 6 (2021), 1–27.

[17] Gunnar Brinkmann, Kris Coolsaet, Jan Goedgebeur, and Hadrien Mélot. 2013. House of Graphs: a database of interesting graphs. *Discrete Applied Mathematics* 161, 1-2 (2013), 311–314.

[18] Raphaël Charbey and Christophe Prieur. 2019. Stars, holes, or paths across your Facebook friends: A graphlet-based characterization of many networks. *Network Science* 7, 4 (2019), 476–497.

[19] Jingji Chen and Xuehai Qian. 2020. Dwarvesgraph: A high-performance graph mining system with pattern decomposition. *arXiv preprint arXiv:2008.09682* (2020).

[20] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. 2020. Can graph neural networks count substructures? *ArXiv* abs/2002.04025 (2020).

[21] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence* 26, 10 (2004), 1367–1372.

[22] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34, 2 (1991), 786–797.

[23] Sofie Demeyer, Tom Michoel, Jan Fostier, Pieter Audenaert, Mario Pickavet, and Piet Demeester. 2013. The index-based subgraph matching algorithm (ISMA): fast subgraph enumeration in large networks using optimized search trees. *PloS one* 8, 4 (2013), e61183.

[24] Evan Donato, Ming Ouyang, and Cristian Peguero-Isalguez. 2018. Triangle Counting with A Multi-Core Computer. *2018 IEEE High Performance extreme Computing Conference (HPEC)* (2018), 1–7.

[25] Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.

[26] Katherine Faust. 2010. A puzzle concerning triads in social networks: Graph constraints and the triad census. *Social Networks* 32, 3 (2010), 221–233.

[27] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[28] Peter Floderus, Mirosław Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. 2015. Induced subgraph isomorphism: Are some patterns substantially easier than others? *Theoretical Computer Science* 605 (2015), 119–128.

[29] Tianyu Fu, Ziqian Wan, Guohao Dai, Yu Wang, and Huazhong Yang. 2020. LessMine: Reducing Sample Space and Data Access for Dense Pattern Mining. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–7.

[30] Chao Gao and John Lafferty. 2017. Testing for global network structure using small subgraph statistics. *arXiv preprint arXiv:1710.00862* (2017).

[31] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[32] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[33] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together. In *Proceedings of the 2019 International Conference on Management of Data*. 1429–1446.

[34] Huahai He and Ambuj K Singh. 2008. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 405–418.

[35] Paul W Holland and Samuel Leinhardt. 1976. Local structure in social networks. *Sociological methodology* 7 (1976), 1–45.

[36] Petter Holme and Beom Jun Kim. 2002. Growing scale-free networks with tunable clustering. *Physical review E* 65, 2 (2002), 026107.

[37] Alon Itai and Michael Rodeh. 1977. Finding a minimum circuit in a graph. In *Proceedings of the ninth annual ACM symposium on Theory of computing*. 1–10.

[38] Anand Padmanabha Iyer, Zaoxing Liu, Xin Jin, Shivaram Venkataraman, Vladimir Braverman, and Ion Stoica. 2018. {ASAP}: Fast, approximate graph pattern mining at scale. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 745–761.

[39] Madhav Jha, C Seshadhri, and Ali Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th international conference on world wide web*. 495–505.

[40] Yuval Kalish and Garry Robins. 2006. Psychological predispositions and network structure: The relationship between individual predispositions, structural holes and network closure. *Social networks* 28, 1 (2006), 56–84.

[41] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. 2004. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics* 20, 11 (2004), 1746–1758.

[42] AA Leman and Boris Weisfeiler. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya* 2, 9 (1968), 12–16.

[43] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[44] Wenqing Lin, Xiaokui Xiao, Xing Xie, and Xiao-Li Li. 2016. Network motif discovery: A GPU approach. *IEEE transactions on knowledge and data engineering* 29, 3 (2016), 513–528.

[45] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, and Xin Jiang. 2020. Neural Subgraph Isomorphism Counting. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).

[46] Xin Liu and Yangqiu Song. 2022. Graph convolutional networks with dual message passing for subgraph isomorphism counting and matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7594–7602.

[47] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. 2019. Provably powerful graph networks. *Advances in neural information processing systems* 32 (2019).

[48] Daniel Mawhirter, Sam Reinehr, Connor Holmes, Tongping Liu, and Bo Wu. 2019. Graphzero: Breaking symmetry for efficient graph mining. *arXiv preprint arXiv:1911.12877* (2019).

[49] Daniel Mawhirter and Bo Wu. 2019. Automine: harmonizing high-level abstraction and high performance for graph mining. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 509–523.

[50] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[51] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[52] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. *ArXiv* abs/1810.02244 (2019).

[53] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102 (2016), 209–245.

[54] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. 2020. k-hop Graph Neural Networks. *Neural networks : the official journal of the International Neural Network Society* 130 (2020), 195–205.

[55] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[56] Mark Ortmann and Ulrik Brandes. 2017. Efficient orbit-aware triad and quad census in directed and undirected graphs. *Applied network science* 2, 1 (2017), 1–17.

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[58] Tiago P. Peixoto. 2014. The graph-tool python library. *figshare* (2014). https://doi.org/10.6084/m9.figshare.1164194

[59] Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th international conference on world wide web*. 1431–1440.

[60] Christina Prell and John Skvoretz. 2008. Looking at social capital through triad structures. *Connections* 28, 2 (2008), 4–16.

[61] Ronald C Read and Robin J Wilson. 1998. *An atlas of graphs*. Vol. 21. Clarendon Press Oxford.

[62] Bernardete Ribeiro, Ning Chen, and Alexander Kovacec. 2017. Shaping graph pattern mining for financial risk. *Neurocomputing* (2017).

[63] Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. 2021. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.

[64] Pedro Ribeiro and Fernando Silva. 2010. Efficient subgraph frequency estimation with g-tries. In *International Workshop on Algorithms in Bioinformatics*. Springer, 238–249.

[65] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[66] Tanay Kumar Saha and Mohammad Al Hasan. 2015. Finding network motifs using MCMC sampling. In *Complex Networks VI*. Springer, 13–24.

[67] Tianhui Shi, Mingshu Zhai, Yi Xu, and Jidong Zhai. 2020. Graphpi: High performance graph pattern matching through effective redundancy elimination. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14.

[68] Neil James Alexander Sloane. 2014. *A handbook of integer sequences*. Academic Press.

[69] Ricard V Solé and Sergi Valverde. 2008. Spontaneous emergence of modularity in cellular networks. *Journal of The Royal Society Interface* 5, 18 (2008), 129–133.

[70] Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. 2019. AqSolDB, a curated reference set of aqueous solubility and 2D descriptors for a diverse set of compounds. *Scientific data* 6, 1 (2019), 143.

[71] Olaf Sporns, Rolf Kötter, and Karl J Friston. 2004. Motifs in brain networks. *PLoS biology* 2, 11 (2004), e369.

[72] Shixuan Sun and Qiong Luo. 2020. In-memory subgraph matching: An in-depth study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1083–1098.

[73] Ichigaku Takigawa and Hiroshi Mamitsuka. 2013. Graph mining: procedure, application to drug discovery and recent advances. *Drug discovery today* 18, 1-2 (2013), 50–57.

[74] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*. 1451–1460.

[75] Shahadat Uddin, Liaquat Hossain, et al. 2013. Dyad and triad census analysis of crisis communication network. *Social Networking* 2, 01 (2013), 32.

[76] Leslie G Valiant. 1979. The complexity of enumeration and reliability problems. *SIAM J. Comput.* (1979).

[77] Sergi Valverde and Ricard V Solé. 2005. Network motifs in computational graphs: A case study in software architecture. *Physical Review E* 72, 2 (2005), 026107.

[78] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[79] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John CS Lui, Don Towsley, Jing Tao, and Xiaohong Guan. 2017. MOSS-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (2017), 73–86.

[80] Stanley Wasserman, Katherine Faust, et al. 1994. Social network analysis: Methods and applications. (1994).

[81] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[82] Melanie Weber. 2019. Curvature and Representation Learning: Identifying Embedding Spaces for Relational Data.

[83] Sebastian Wernicke and Florian Rasche. 2006. FANMOD: a tool for fast network motif detection. *Bioinformatics* 22, 9 (2006), 1152–1153.

[84] Elisabeth Wong, Brittany Baur, Saad Quader, and Chun-Hsi Huang. 2012. Biological network motif detection: principles and practice. *Briefings in bioinformatics* 13, 2 (2012), 202–215.

[85] Peng Wu, Junfeng Wang, and Bin Tian. 2018. Software homology detection with software motifs based on function-call graph. *IEEE Access* 6 (2018), 19007–19017.

[86] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[87] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.

[88] Chen Yang, Min Lyu, Yongkun Li, Qianqian Zhao, and Yinlong Xu. 2018. SSRW: a scalable algorithm for estimating graphlet statistics based on random walk. In *International Conference on Database Systems for Advanced Applications*. Springer, 272–288.

[89] Guan-Can Yang, Gang Li, Chun-Ya Li, Yun-Hua Zhao, Jing Zhang, Tong Liu, Dar-Zen Chen, and Mu-Hsuan Huang. 2015. Using the comprehensive patent citation network (CPC) to evaluate patent value. *Scientometrics* 105, 3 (2015), 1319–1346.

[90] Hao Yin, Austin R Benson, and Jure Leskovec. 2019. The local closure coefficient: A new perspective on network clustering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 303–311.

[91] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/e77dbaf6759253c7c6d0efc5690369c7-Paper.pdf

[92] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-aware graph neural networks. *arXiv preprint arXiv:2101.10320* (2021).

[93] Kangfei Zhao, Jeffrey Xu Yu, Hao Zhang, Qiyan Li, and Yu Rong. 2021. A Learned Sketch for Subgraph Counting. *Proceedings of the 2021 International Conference on Management of Data* (2021).

[94] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11168–11176.