# Local Fragments, Global Gains: Subgraph Counting using Graph Neural Networks

**Shubhajit Roy**
Indian Institute of Technology Gandhinagar
royshubhajit@iitgn.ac.in

**Shrutimoy Das**
Indian Institute of Technology Gandhinagar
shrutimoydas@iitgn.ac.in

**Binita Maity**
Indian Institute of Technology Gandhinagar
binitamaity@iitgn.ac.in

**Anant Kumar**
Indian Institute of Technology Gandhinagar
kumar_anant@iitgn.ac.in

**Anirban Dasgupta**
Indian Institute of Technology Gandhinagar
anirbandg@iitgn.ac.in

## Abstract

Subgraph counting is a fundamental task for analyzing structural patterns in graph-structured data, particularly crucial for applications in computational biology and social network analysis, where identifying recurring motifs reveals functional properties and organizational structures. We propose a novel three-stage differentiable learning algorithm that computes the counts of various patterns by learning to combine the counts of its subpatterns. Our approach leverages localized versions of Weisfeiler-Leman (WL) algorithms and introduces a novel fragmentation technique that decomposes complex subgraphs into simpler patterns. This technique enables exact counting of all induced subgraphs of size at most 4 using just 1-WL. This method significantly improves upon existing Graph Neural Network(GNN) based approaches for subgraph counting, being computationally efficient, making it well-suited for learning combinatorial algorithms.

## 1 Introduction

Subgraph counting represents one of the most fundamental challenges when working with graph-structured datasets, with profound implications for understanding complex network structures across diverse domains. In computational biology, the ability to count specific molecular substructures such as benzene rings or protein binding motifs directly impacts drug discovery and molecular property prediction. Similarly, in social network analysis, counts of triangular patterns, stars, and other local structures provide insights into community formation, information propagation, and social capital dynamics.

The importance of accurate subgraph counting extends beyond mere pattern detection—it serves as a critical measure of a GNN's expressive power. Traditional message-passing neural networks (MPNNs) are fundamentally limited by their equivalence to the 1-dimensional Weisfeiler-Leman (1-WL) test, which prevents them from distinguishing between graphs that differ in their subgraph counts but are otherwise structurally similar. This limitation has significant practical consequences: standard GNNs cannot count triangles, cycles larger than 3 nodes, or other complex substructures that are essential for many real-world applications.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: DiffCoALG: Differentiable Learning of Combinatorial Algorithms.

Previous research [1, 2, 3, 4, 5] has primarily focused on two main paradigms to overcome these expressivity limitations. The first approach involves developing more powerful GNN architectures based on higher-order WL tests ($k$-WL), which can theoretically count larger substructures but suffer from prohibitive computational complexity scaling as $O(n^k)$, with $n$ being the number of nodes in the graph. The second paradigm employs nested GNN approaches, which extract local subgraphs around each node and apply base GNNs to these subgraphs. While nested GNNs can count certain substructures, they require expensive preprocessing steps and suffer from high time and memory costs when encoding large graphs. [6, 7] show that message-passing GNNs can count subgraphs by leveraging node individualization or learning sufficient graph statistics.

Subgraph GNNs represent another significant direction, where the input graph is partitioned into numerous subgraphs, and GNNs are applied to each subgraph to augment the overall graph representation. However, these methods require repetitive application of GNNs across all subgraphs, leading to substantial computational overhead that limits their practical applicability to large-scale problems.

Our proposed technique is motivated by the development of localized variants of the WL algorithms [8]. We use the term "local $k$-WL " to refer to such algorithms. These algorithms reduce the computational overhead of the standard $k$-WL approaches. The local $k$-WL algorithms operate on restricted neighbourhoods of a graph rather than the entire graph. Since it has been established that the expressiveness of standard GNNs is the same as 1-WL [9, 10], we can consider subgraph GNNs to have expressive power equivalent to local 1-WL.

Here, we introduce a novel **fragmentation** technique that represents a paradigm shift in how we approach complex subgraph counting. Instead of attempting to count complex subgraphs directly, our fragmentation method decomposes target subgraphs into simpler constituent patterns whose counts can be computed exactly using efficient algorithms.

The fragmentation approach is particularly well-suited for differentiable combinatorial algorithm learning frameworks because it naturally decomposes the complex counting problem into learnable components. Our three-stage learning algorithm first learns to identify the required subpatterns for any target subgraph, then accurately counts these subpatterns using our localized WL variants, and finally aggregates the subpattern counts into the final global count. This decomposition not only provides theoretical guarantees for correctness but also enables efficient gradient-based optimization of the entire counting pipeline.

Our method represents a significant advancement over existing approaches by combining the theoretical rigor of WL-based methods with the practical efficiency needed for real-world applications, while providing a natural fit for the emerging field of differentiable combinatorial algorithms.

## 2 Preliminaries

We consider a simple undirected graph $G(V, E)$. For basic definitions of graph theory, we refer the reader to [11]. The neighbourhood of a node $v \in V$ is denoted as $\mathcal{N}_G(v)$. The *closed* neighbourhood of $v$ is the set of all neighbours, including the node $v$ (denoted as $\mathcal{N}_G[v]$). We use the notation $d_v$ to refer to the degree of a node $v$. The radius of a graph is the minimum over all the nodes of a graph of the maximum distance from a node to any other node in the graph.

A graph $H$ is called a *subgraph* of $G$ if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The subgraph induced on $S \subseteq V(G)$ is a graph whose node set $S$ contains all the edges in $G$ whose endpoints are in $S$ and is denoted by $G[S]$. The *induced subgraph* on an $r$-hop neighbourhood around node $v$ is denoted by $G_v^r$. The number of hops $r$ depends on the pattern of interest. For example, in order to count triangles, $r = 1$ is sufficient, whereas for counting $C_4$, we require $r = 2$. Attributed subgraphs are subgraphs where each node is marked with an attribute or a colour (also referred to as motifs). We note here that we count the number of patterns occurring as subgraphs in our work.

The Weisfeiler-Leman test [12] is a type of colour refinement algorithm used for testing the existence of an isomorphism between two graphs $G_1$ and $G_2$. Interested readers may refer to [13] for a detailed discussion. The local $k$-WL algorithm is applied to local subgraphs. Specifically, for each node $v \in V$, we extract $G_v^r$ and apply $k$-WL on it. While running $k$-WL on a graph with $n$ vertices require $O(n^{k+1} \log n)$ time, running local $k$-WL on any $G_v^r$ extracted from a graph of bounded degree $d$ requires times $O(n.d^{r(k+1)}. \log d)$.

Graph Neural Networks were first introduced by [14]. The model proposed a recursive use of two functions: MESSAGE and AGGREGATE to update the node embeddings $X^\ell$ (for the $\ell$th layer).

$$X_v^{(\ell)} = \texttt{UPDATE}^{(\ell)}\left(X_v^{(\ell-1)},\ \texttt{AGGREGATE}^{(\ell)}\left\{\texttt{MESSAGE}^{(\ell)}\left(X_v^{(\ell-1)},\ X_u^{(\ell-1)}\right) \mid u \in \mathcal{N}_G(v)\right\}\right) \quad (1)$$

## 3 Methodology

Before describing the fragmentation algorithm, we demonstrate the method using an example as shown in Figure 1. Suppose that the pattern to be counted is a tailed triangle. Rooted at a node $v$ (the red coloured node), we consider a 1-hop subgraph $G_v^1$. Now consider the subgraph $H_v = G_v^1 \setminus \{v\}$. If a tailed triangle is present, then $H_v$ must have an isolated node and an edge (Fragments 1 and 2 in Figure 1). Hence, the counts of such (isolated node, edge) pairs give a count of the number of tailed triangles.

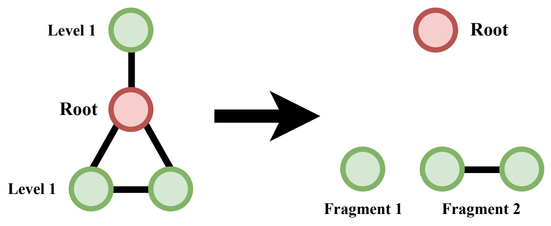


Figure 1: Fragmentation of a tailed triangle

Our proposed methodology comprises three components **(A) Pattern Learning**, **(B) Local Count Learning**, and **(C) Global Count Learning** as shown in Figure 2.

### 3.1 Pattern Learning

In the task of counting a particular pattern, such as a triangle or a 3-star, $G_v^1$ from each node $v$ is different. Counting triangles requires edges between the neighboring nodes of the root node only. Whereas, in the case of a 3-star, the subgraphs have edges between the root node and neighboring nodes. Therefore, we propose a learnable model that learns the required subgraph for a given pattern. For each node, we modify the $G_v^r$ by adding direction to the edges. We add a directed edge from the root node to other nodes. This modification is done to indicate which node is the root node in the subgraph. These modified subgraphs are then given as input to a GNN model, which updates the node embeddings. With the updated node embeddings, we construct the edge embeddings and classify the edges as 0 or 1, indicating which edges are required from $G_v^r$ as input for the next component.

### 3.2 Local Count Learning

From the previous section on pattern learning, we take the updated $G_v^r$ as input, where certain edges are removed based on the pattern that we want to take as input. For counting triangles, the local count (number of times a pattern appears in the local subgraph) is the number of edges in $G_v^r \setminus \{v\}$. For counting $k$-star graphs, the local count is the $\binom{d_v}{k}$, where $d_v$ is the degree of $v$. Likewise, we have different local counts as ground truth for different patterns. We input the updated $G_v^r$ to a GNN model and train the model based on the predictions of the local count.

### 3.3 Global Count Learning

While counting in different subgraphs, there is a potential for overcounting of the patterns. Therefore, we have a normalization model that takes the summation of all the local counts and learns the normalization factor required to counter the overcounted patterns.

### 3.4 Fragmentation

The fragmentation method, Algorithm 1, involves fragmenting the pattern $P$ into smaller subpatterns and counting these subpatterns to get the count of $P$ in the graph. The number of occurrences of

$P$ in $G_v^r$ can be computed by combining the counts of the simpler patterns (fragments). Instead of training a GNN for counting $P$, we can design GNNs for learning the easier tasks (i.e., for counting the fragments) and combine the outputs of those models. It should be noted that the fragmentation into smaller subgraphs depends on the structure of the pattern $P$.

Given a graph $G$ and a pattern $P$(whose number of occurrences is to be counted), we first fix a vertex $v \in V(G)$ as the key vertex. Now, assume that the radius of the pattern is $r$. Thus, for counting $P$ locally, it is sufficient to take $G_v^r$. Now, we fragment pattern $P$ into smaller subpatterns, say $\mathbb{P} = \{P_1, P_2, \dots P_l\}$. It should be noted that the decomposition of $P$ into $P_i$s is done in such a way that their counts can be calculated exactly (by a local 1-WL based trained GNN model). That is, we have learnt models $M_i^{\text{pattern}}$, corresponding to each $P_i$, which generates $\mathcal{P}_u$ for each node in $H_u = G_v^r \setminus \{v\}$. We also have learned models $M_i^{\text{count}}$, corresponding to each $P_i$, that count the number of subpatterns $P_i$ in $\mathcal{P}_u$. The array $c$ stores the count of $P_i$'s in each $H_u$. Now, for each subpattern $P_i$, we learn a function $\alpha$ as weights in a linear transformation to combine the counts in $c$ to get the count of $P_i$ in $H_u$. Then we learn the function $\beta$ to count $P$ for each root node. Finally, the function $\gamma$ finds the normalizing factor to get the actual count of the pattern in $G$.

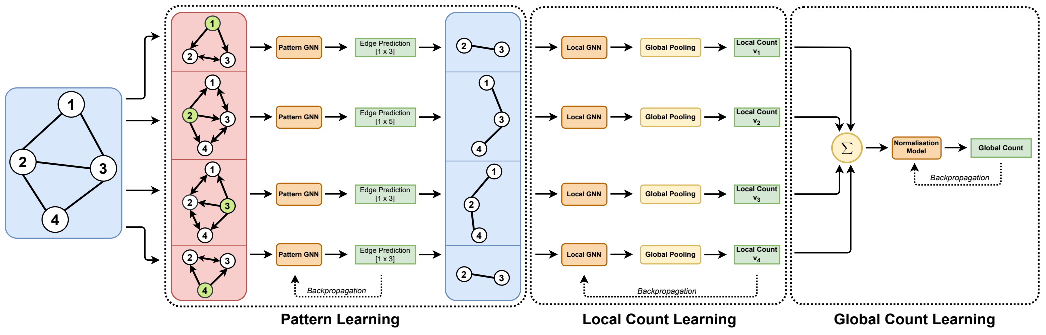


Figure 2: Overview of the framework. The pipeline is divided into three components – **(A) Pattern Learning:** $r$-hop neighborhood of each node is given as input with directed edges indicating the root node, where the model's task is to predict the required pattern corresponding to each node. **(B) Local Count Learning:** taking the predicted patterns as input from the **Pattern Learning** component, we predict the local count for each root node. **(C) Global Count Learning:** Upon addition of all local counts, we require to learn a normalization factor, which is learned in this section of the pipeline.

**Algorithm 1** Fragmentation Algorithm

```
Require: $G$; $\mathbb{P}$: List of patterns; $M_i^{\text{pattern}}$: learned model for generating pattern corresponding to
    $P_i \in \mathbb{P}$; $M_i^{\text{count}}$: learned model for counting $P_i \in \mathbb{P}$;
 1: $a \leftarrow []$
 2: for each node $v \in V(G)$ do
 3:     $H_v = G_v^r \setminus \{v\}$
 4:     $b \leftarrow []$
 5:     for each pattern $P_i \in \mathbb{P}$ do
 6:         $c \leftarrow []$
 7:         for each node $u \in H_v$ do
 8:             $\mathcal{P}_u = M_i^{\text{pattern}}(H_v, u)$
 9:             $c.append(M_i^{\text{count}}(\mathcal{P}_u))$
10:         end for
11:         $b.append(\alpha(c))$      #Learnable function
12:     end for
13:     $a.append(\beta(b))$      #Learnable function
14: end for
15: $Count = \gamma(a)$      #Learnable function
16: return $Count$
```

## 4 Experiments

### 4.1 Implementation Details

We refer to our proposed model as **InSig**. In the experiments, we predict the counts of the following substructures occurring as subgraphs: triangles, 3-Star, 2-Star, chordal $C_4$, $K_4$, $C_4$, and tailed triangles. When counting larger substructures like $K_4$, $C_4$, and Tailed Triangles, we use the

| **Task** | **Total pattern count** | **Zero Count graphs** | **Standard Deviation of Count** | **Average number of Nodes** | **Average number of Edges** | **Number of graphs** |
|---|---|---|---|---|---|---|
| *Triangle* | 25209 | 195 | 3.072 | | | |
| *2-Star* | 429463 | 0 | 18.015 | | | |
| *3-Star* | 309525 | 0 | 17.777 | | | |
| *Chordal* | 19088 | 1786 | 4.742 | 18.7976 | 62.678 | 5000 |
| *K4* | 643 | 4447 | 0.387 | | | |
| *C4* | 53002 | 16 | 6.938 | | | |
| *Tailed Triangle* | 177968 | 195 | 25.943 | | | |

Table 1: Dataset statistics. The total number of graphs in the dataset is 5000. We used 4000 graphs for training, 500 for validation, and 500 for testing.

**fragmentation** technique with the help of a model learned to predict triangles, 3-Star, 2-Star, and Chordal $C_4$. For counting $K_4$, the task of **Pattern Counting** Component is to learn which edges to prune. Later **Local Count Learning** component counts the number of substructures, which in the case of $K_4$, is triangles. Once the number of triangles is predicted using models learned to count triangles, the normalization factor is learned to output the global count. In other structures like tailed triangles, we have two patterns to learn: the nodes in the 1-hop neighborhood and the edges between them. During the inference phase, we use a rounding function, as the counts are integer numbers.

### 4.2 Hyperparameters

We use two GIN Convolutional layers for the Pattern Learning Local Count Learning component. We consider Linear transformations as readout layers in the previously mentioned components. Since we need to classify whether an edge should be present or not in a subgraph for counting a subpattern, we use Binary Cross-Entropy (BCE) loss for the pattern learning component. For the local counting and global counting components, we use Mean Absolute Error (MAE). For the models we have considered, as there can be paths of lengths more than 1 in the subgraph, 2 *GINConv* layers are sufficient to capture the information well.

We use a learning rate of $1e-4$ and a batch size of 1. We also experimented with different hidden dimensions for the node embeddings and obtained the best results when we used 4 as a hidden dimension size. The experiments were conducted using an NVIDIA A100 40GB GPU. The source code of the implementation is available at this Github Link.

### 4.3 Experimental Results

| **Models** | **Without Fragmentation** | | | | **Fragmentation** | | |
|---|---|---|---|---|---|---|---|
| | *Triangle* | *3-Star* | *2-Star* | *Chorcal C4* | $K_4$ | $C_4$ | *Tailed Triangle* |
| **ID-GNN** | 6.00E-04 | NA | NA | 4.52E-02 | 2.60E-03 | 2.20E-03 | 1.05E-01 |
| **NGNN** | 3.00E-04 | NA | NA | 3.92E-02 | 4.50E-03 | 1.30E-03 | 1.04E-01 |
| **GNNAK+** | 4.00E-04 | 1.50E-02 | NA | 1.12E-02 | 4.90E-03 | 4.10E-03 | 4.30E-03 |
| **PPGN** | 3.00E-04 | NA | NA | 1.50E-03 | 1.65E-01 | 9.00E-04 | 2.60E-03 |
| **I2-GNN** | 4.00E-04 | NA | NA | 1.00E-03 | 3.00E-04 | 1.60E-03 | 1.10E-03 |
| **InSig** | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |

Table 2: MAE for the subgraph count of different patterns. Some results, such as 2-*star*, 3-*star*, are not conducted by the given baselines; therefore, it is mentioned NA.

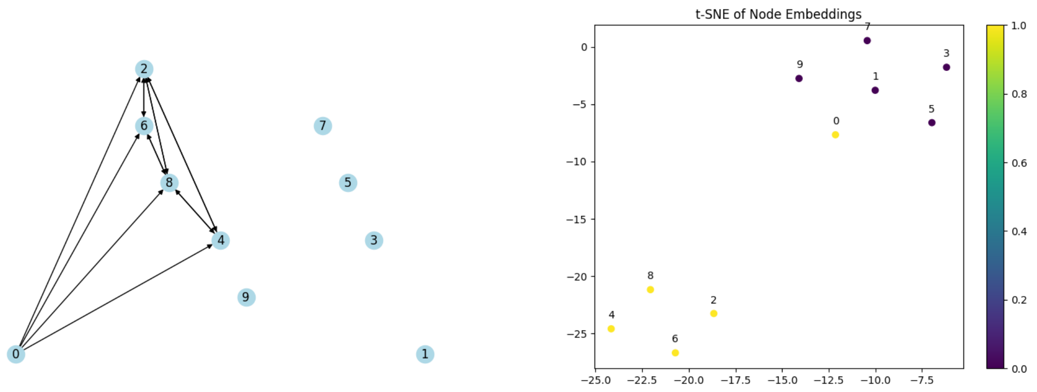
The dataset used for the experiments is a random graphs dataset prepared in [15]. In Table 1, we report the dataset statistics, specifically the counts of the various patterns used for our experiments, the number of graphs where these patterns do not appear, and so on. We report our experiments' Mean Absolute Error (MAE) in Table 2. We compare our results with those reported in *ID-GNN*

[2], *NGNN* [3], *GNNAK+* [4], *PPGN* [1] and *I2-GNN* [5]. We can observe that our approach of predicting substructures, the local counts, and then predicting the global counts, achieves zero error for all of the tests.

For all the patterns, we observed that the model gets to zero error after only 2 to 3 epochs. From Table 3, it can be observed that our approach requires a considerably smaller number of parameters and beats all the baselines. Our inference time comprises model inference as well as the graph preprocessing time, where it creates the set of subgraphs corresponding to each node in the graph.

| **Models** | **Number of Parameters** | **Inference Time (ms)** | **Memory Usage (GB)** |
|---|---:|---:|---:|
| **ID-GNN** | 102K | 5.73 | 2.35 |
| **NGNN** | 127K | 6.03 | 2.34 |
| **GNNAK+** | 251K | 16.07 | 2.35 |
| **PPGN** | 96K | 35.33 | 2.3 |
| **I2-GNN** | 143K | 20.62 | 3.59 |
| **InSig** | **268** | **20** | **1.2** |

Table 3: The table shows the comparison of the number of parameters required by the baselines and *InSig Model*. The values shown here correspond to the triangle counting task with the hidden dimension set as $4$.



(a) Figure shows an example of $G_0^1$ for a random graph with 10 nodes

(b) t-SNE plot of the updated node embeddings from the Pattern learning component

Figure 3: Figure showing the input and updated node embedding from the pattern learning component

In Figure 3, we show an example of a graph sent as input to the pattern learning component and the updated node embeddings output from the model. In Figure 3a, we see that the root node is node $0$ and there are directed edges from the root node to its neighboring nodes. Figure 3b shows the updated node embeddings of each node in the graph. We can observe that $8, 2, 4, 6$ nodes can be separated using some separator from the rest of the nodes. This indicates that the model is able to learn the updated node embedding such that we can distinguish nodes which has edges between them.

## 5 Conclusion

Subgraph counting is a fundamental combinatorial problem that arises in the study of graphs and graph-structured problems. Exactly counting the number of subgraphs in a graph is also a computationally hard problem. In this paper, we present a learnable algorithm that is able to compute exact counts of a number of commonly occurring patterns. The proposed fragmentation method has proven to be beneficial for the task of counting subgraphs. Additionally, since fragmentation results in smaller subgraphs, the parameter requirements of the proposed GNN-based architecture are orders of magnitude less than those of previous methods. As future work, we plan to analyse the fragmentation algorithm from a theoretical perspective. We also plan to investigate the effectiveness of this technique for increasing the expressiveness of GNNs for downstream tasks. Along with that, the current approach is limited to patterns that can be computed using 1-hop or 2-hop subgraphs; hence, we can study our algorithm for bigger patterns.

## References

[1] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks, 2019.

[2] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks, 2021.

[3] Muhan Zhang and Pan Li. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34:15734–15747, 2021.

[4] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

[5] Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. Boosting the cycle counting power of graph neural networks with $i^2$-gnns, 2023.

[6] Paolo Pellizzoni, Till Hendrik Schulz, Dexiong Chen, and Karsten Borgwardt. On the expressivity and sample complexity of node-individualized graph neural networks. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[7] Paolo Pellizzoni, Till Hendrik Schulz, and Karsten Borgwardt. Graph neural networks can (often) count substructures. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Christopher Morris, Kristian Kersting, and Petra Mutzel. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 327–336, 2017.

[9] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks, 2021.

[10] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[11] Douglas Brent West. *Introduction to graph theory*. Prentice Hall, Upper Saddle River, N.J., 2nd ed edition, 2001.

[12] Boris Weisfeiler and A. A. Lehman. A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. *Nauchno-Technicheskaya Informatsia*, Ser. 2(N9):12–16, 1968.

[13] Ningyuan Teresa Huang and Soledad Villar. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 8533–8537. IEEE, June 2021.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[15] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims justified using emperical results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Part of conclusion and future work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results presented.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Implementation details added in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code Link Provided

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Added details in Section 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Zero Error across multiple runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Provided in Section 4 and Table 3

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No ethics violation

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Theoretical paper on a synthetic dataset.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
  - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
  - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No Risks

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Citations provided.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No Assets released

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Work on the synthetic dataset.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

Justification: No LLM used

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.