GENERALIZING WEISFEILER-LEHMAN KERNELS TO SUBGRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Subgraph representation learning has been effective in solving various real-world problems. However, current graph neural networks (GNNs) produce suboptimal results for subgraph-level tasks due to their inability to capture complex interactions within and between subgraphs. To provide a more expressive and efficient alternative, we propose WLKS, a Weisfeiler-Lehman (WL) kernel generalized for subgraphs by applying the WL algorithm on induced k-hop neighborhoods. We combine kernels across different k-hop levels to capture richer structural information that is not fully encoded in existing models. Our approach can balance expressiveness and efficiency by eliminating the need for neighborhood sampling. In experiments on eight real-world and synthetic benchmarks, WLKS significantly outperforms leading approaches on five datasets while reducing training time, ranging from 0.01x to 0.25x compared to the state-of-the-art.

1 INTRODUCTION

Subgraph representation learning has effectively tackled various real-world problems (Bordes et al., 2014; Luo, 2022; Hamidi Rad et al., 2022; Maheshwari et al., 2024). However, existing graph neural networks (GNNs) still produce suboptimal representations for subgraph-level tasks since they fail to capture arbitrary interactions between and within subgraph structures. These GNNs cannot capture high-order interactions beyond and even in their receptive fields. Thus, state-of-the-art models for subgraphs have to employ hand-crafted channels (Alsentzer et al., 2020), node labeling (Wang & Zhang, 2022), and structure approximations (Kim & Oh, 2024) to encode subgraphs' complex internal and border structures.

As an elegant and efficient alternative, we generalize graph kernels to subgraphs, which measure the structural similarity between pairs of graphs. We propose WLKS, the Weisfeiler-Lehman (WL) Kernel for Subgraphs based on WL graph kernel (Shervashidze & Borgwardt, 2009). Specifically, we apply the WL algorithm (Leman & Weisfeiler, 1968) on induced k-hop subgraphs around the target subgraph for all possible ks. The WL algorithm's output (i.e., the color histogram) for each k encodes structures in the receptive field of the k-layer GNNs; thus, the corresponding kernel matrix can represent the similarity of k-hop subgraph pairs. A classifier using this kernel can be trained without GPUs in a computationally efficient way compared to deep GNNs.

To enhance the expressive power, we linearly combine kernel matrices of different k-hops. The motivation is that simply using larger hops for WL histograms does not necessarily lead to more expressive representations. We theoretically demonstrate that WL histograms of the $(k+1)$-hop are not strictly more expressive than those of k-hop in distinguishing isomorphic structures, while $(k+1)$-hop structures include entire k-hop structures. Therefore, combining kernel matrices across multiple k-hop levels can capture richer structural information around subgraphs.

However, sampling k-hop subgraphs can increase the time and space complexity, as the number of nodes in the k-hop neighborhoods grows exponentially (Hamilton et al., 2017). To mitigate this issue, we choose only two values of k: 0 and the diameter D of the global graph. No neighborhood sampling is required for the case where $k = 0$ since it only uses the internal structure. When k is set to the diameter D, the expanded subgraph encompasses the entire global graph, making the k-hop neighborhood identical for all subgraphs. Consequently, there is no need for explicit neighborhood sampling in this case; we only perform the WL algorithm on the global graph once. This approach balances expressiveness and efficiency, providing a practical solution for subgraph-level tasks.

We evaluate WLKS's classification performance and efficiency with four real-world and four synthetic benchmarks (Alsentzer et al., 2020). Our model outperforms the best-performed methods across five of the eight datasets. Remarkably, this performance is achieved with $\times 0.01$ to $\times 0.53$ training time compared to the state-of-the-art models. Moreover, unlike existing models, WLKS does not require pre-computation, pre-training embeddings, utilizing GPUs, and searching a large hyperparameter space.

The main contributions of our paper are summarized as follows. First, we propose WLKS, a generalization of graph kernels to subgraphs. Second, we theoretically show that combining WLKS matrices from multiple k-hop neighborhoods can increase the expressiveness. Third, we evaluate our method on real-world and synthetic benchmarks and demonstrate superior performance in a significantly efficient way. We make our code available for future research[1].

2 Related Work

WLKS is a 'graph kernel' method designed for 'subgraph representation learning.' This section explains both of these areas and their relationship to our model.

Subgraph Representation Learning Subgraph representation learning can address various real-world challenges by capturing higher-order interactions that nodes, edges, or entire graphs cannot model. For example, subgraphs can formulate diseases and patients in gene networks (Luo, 2022), teams in collaboration networks (Hamidi Rad et al., 2022), and communities in mobile game user networks (Zhang et al., 2023). Existing methods are often domain-specific (Zhang et al., 2023; Li et al., 2023; Trümper et al., 2023; Ouyang et al., 2024; Maheshwari et al., 2024) or rely on strong assumptions about the subgraph (Meng et al., 2018; Hamidi Rad et al., 2022; Kim et al., 2022; Luo, 2022; Liu et al., 2023), limiting their generalizability.

Recent deep graph neural networks designed for subgraph-level tasks can apply to any subgraph type without specific assumptions. However, they often generate suboptimal representations due to their inability to capture arbitrary interactions between and within subgraph structures. They struggle to account for high-order interactions beyond their limited receptive fields; thus, they should incorporate additional techniques including hand-crafted channels (Alsentzer et al., 2020), node labeling (Wang & Zhang, 2022), random-walk sampling (Jacob et al., 2023), and structural approximations (Kim & Oh, 2024). In contrast, we design kernels that can capture local and global interactions of subgraphs, respectively, to enable simple but strong subgraph prediction. We formally compare our proposed WLKS with representative prior models in Appendix A.

Graph Kernels Graph kernels are algorithms to measure the similarity between graphs to enable the kernel methods, such as Support Vector Machines (SVMs) to graph-structured data (Vishwanathan et al., 2010). Early examples measure the graph similarity based on random walks (Kashima et al., 2003) or shorted paths (Borgwardt & Kriegel, 2005). One of the most influential graph kernels is the Weisfeiler-Lehman (WL) kernel (Shervashidze & Borgwardt, 2009), which leverages the WL isomorphism test to refine node labels iteratively, improving the expressiveness of the graph structure comparison. While the WL test is designed for graph isomorphism, WL kernels capture structural similarities using the WL test's outcomes even when graphs are not strictly isomorphic (See Appendix B for detailed comparison). Kernels for graph-level prediction by counting, matching, and embedding subgraphs have been deeply explored (Shervashidze et al., 2009; Kriege & Mutzel, 2012; Yanardag & Vishwanathan, 2015; Narayanan et al., 2016). However, there has been no research on kernels to solve subgraph-level tasks by computing the similarity of subgraphs and their surroundings. To the best of our knowledge, our paper is the first to investigate this approach.

3 WL Graph Kernels for Subgraph-Level Tasks

This section introduces WLKS, the WL graph kernels generalized for subgraphs. We first describe the original WL algorithm and its extension for subgraphs, which is a foundation of WLKS. Then, we suggest WLKS and its enhancement of expressiveness and efficiency.

[1]see supplementary materials

3.1 Subgraph Representation Learning

We first formalize subgraph representation learning as a classification task. Let $\mathcal{G} = (\mathbb{V}, \mathbb{A})$ represent a global graph, where $\mathbb{V}$ denotes a set of nodes (with $|\mathbb{V}| = N$) and $\mathbb{A} \subset \mathbb{V} \times \mathbb{V}$ represents a set of edges (with $|\mathbb{A}| = E$). A subgraph $\mathcal{S} = (\mathbb{V}^{\text{sub}}, \mathbb{A}^{\text{sub}})$ is a graph formed by subsets of nodes and edges in the global graph $\mathcal{G}$ (with $|\mathbb{V}^{\text{sub}}| = N^{\text{sub}}$ and $|\mathbb{A}^{\text{sub}}| = E^{\text{sub}}$). There exists a set of M subgraphs, with $M < N$, denoted as $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_M\}$. In a subgraph classification task, the model learns representation $\boldsymbol{h}_i \in \mathbb{R}^F$ and the logit vector $\boldsymbol{y}_i \in \mathbb{R}^C$ for $\mathcal{S}_i$ where F and C are the dimension size and the number of classes, respectively.

3.2 1-WL Algorithm for k-hop Subgraphs

1-WL for Graphs We briefly introduce the 1-dimensional Weisfeiler-Lehman (1-WL) algorithm. As illustrated in Algorithm 1, the 1-WL is an iterative node-color refinement by updating node colors based on a multiset of neighboring node colors. This process produces a histogram of refined coloring that captures graph structure, which can distinguish non-isomorphic graphs in the WL isomorphism test.

Algorithm 1: 1-WL Algorithm

```
Input: Graph $\mathcal{G} = (\mathbb{V}, \mathbb{A})$ and $T$ iterations
Output: Refined node coloring $(c_1^T, c_2^T, ..., c_{|\mathbb{V}|}^T)$ for nodes in $\mathbb{V}$ after $T$ iterations
Initialize $c_v^0$ for all $v \in \mathbb{V}$
for $i \leftarrow 1$ to $T$ do
    for node $v \in \mathbb{V}$ do
        $\mathbb{M}_v \leftarrow$ multiset of labels $\{c_u^{i-1} \mid u \in \mathcal{N}(v)\}$
        $\tilde{c}_v^i \leftarrow$ concatenate $c_v^{i-1}$ and sorted $\mathbb{M}_v$
    end
    Use a bijective function to map each unique $\tilde{c}_v^i$ to a new color $c_v^i$
end
return $(c_1^T, c_2^T, ..., c_{|\mathbb{V}|}^T)$
```

1-WL for Subgraphs (WLS) We then propose the WLS, the 1-WL algorithm generalized for subgraphs. Since surrounding structures are the core difference between graphs and subgraphs, the main contribution of the WLS lies in encoding the k-hop neighborhoods of the subgraph. Here, k will be denoted in superscript as $\texttt{WLS}^k$ if a specific k is given.

Formally, for a subgraph $\mathcal{S} = (\mathbb{V}^{\text{sub}}, \mathbb{A}^{\text{sub}})$ in a global graph $\mathcal{G} = (\mathbb{V}, \mathbb{A})$, the $\texttt{WLS}^k$'s goal is to get the refined colors of nodes in $\mathbb{V}^{\text{sub}}$, where each color represents a unique subtree in k-hop neighborhoods. As in the Algorithm 2, we first extract the k-hop subgraph $\mathcal{S}^k$ of $\mathcal{S}$, which contains all nodes in $\mathcal{S}$ as well as any nodes in $\mathcal{G}$ that are reachable from the nodes in $\mathcal{S}$ within k hops. The 1-WL algorithm is then run on this induced k-hop subgraph to generate the colors of the nodes in $\mathcal{S}^k$. The WLS returns the node coloring belonging to the original $\mathcal{S}$, not in $\mathcal{S}^k$. In general, k-hop neighborhoods are much larger than the original subgraph, so using all the colors in $\mathcal{S}^k$ will likely produce a coloring irreverent to the target subgraph.

Algorithm 2: $\texttt{WLS}^k$ Algorithm: 1-WL for subgraphs with their k-hop neighborhoods

```
Input: A subgraph $\mathcal{S} = (\mathbb{V}^{\text{sub}}, \mathbb{A}^{\text{sub}})$, a global graph $\mathcal{G} = (\mathbb{V}, \mathbb{A})$, and $T$ iterations
Output: Refined node coloring $(c_1^T, c_2^T, ..., c_{|\mathbb{V}^{\text{sub}}|}^T)$ for nodes in $\mathbb{V}^{\text{sub}}$ after $T$ iterations
Sample $\mathcal{S}^k = (\mathbb{V}^{\text{sub},k}, \mathbb{A}^{\text{sub},k})$, which is the induced $k$-hop subgraph of $\mathcal{G}$ around all nodes in $\mathcal{S}$
  reachable within $k$ hops
Run 1-WL (Algorithm 1) on $(\mathcal{S}^k, T)$ to get node colors in $\mathbb{V}^{\text{sub},k}$
return $(c_1^T, c_2^T, ..., c_{|\mathbb{V}^{sub}|}^T)$. Note that this coloring is about nodes in $\mathcal{S}$, not $\mathcal{S}^k$.
```

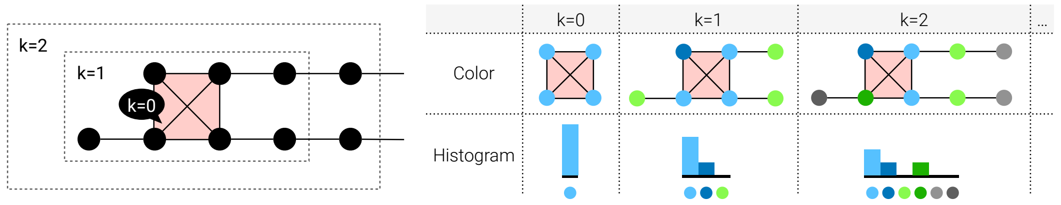


Figure 1: An example of $\texttt{WLS}^k$ algorithm (Algorithm 2) for $k \in \{0, 1, 2\}$. **Left:** A subgraph (red shade) and its k-hop neighborhoods (dashed lines). **Right:** The outputs of $\texttt{WLS}^k$ algorithm as colors and histograms for the left subgraph. We visualize each iteration of $\texttt{WLS}^k$ algorithm in Appendix C. The WLKS kernel matrix for each k is constructed by an inner product of histogram pairs.

After $\texttt{WLS}^k$'s color refinement, we can get a feature vector (or a color histogram) $\phi^k_{\mathcal{S}} \in \mathbb{R}^{\#\text{colors}}$ which is the aggregation of the refined colors in $\mathcal{S}$. Each element $\phi^k_{\mathcal{S}}[c]$ is the number of occurrences of the color c in the output of $\texttt{WLS}^k$. We illustrate an example of a subgraph and its $\texttt{WLS}^k$ outputs for different ks in Figure 1.

WL Kernels for Subgraphs (WLKS) Now, we suggest WLKS, the corresponding kernel matrix $\boldsymbol{K}^k_{\texttt{WLS}} \in \mathbb{R}^{M\times M}$ of which is defined as the number of common subtree patterns of two subgraphs in their k-hop neighborhoods. That is, each element can be formulated as an inner product of a pair of ϕ^k_*. This WLKS is a valid kernel since $\boldsymbol{K}^k_{\texttt{WLS}}$ is positive semi-definite for all non-negative ks, as demonstrated in Proposition 3.1.

Proposition 3.1. $\forall k \geq 0$, $\boldsymbol{K}^k_{\texttt{WLS}}$ *is positive semi-definite (p.s.d.).*

Proof. Each element in $\boldsymbol{K}^k_{\texttt{WLS}}$ is defined as an inner product of two feature vectors ϕ^k_*. This leads $\sum_{i=1}^M \sum_{j=1}^M c_i c_j \langle \phi^k_i, \phi^k_j \rangle = \langle \sum_{i=1}^M c_i \phi^k_i, \sum_{j=1}^M c_j \phi^k_j \rangle = \| \sum_{i=1}^M c_i \phi^k_i \|^2 \geq 0$ for any real c. Thus, $\boldsymbol{K}^k_{\texttt{WLS}}$ is positive semi-definite. □

3.3 Expressiveness Difference of the WLS between k and $k+1$

How do we choose k? Intuitively, selecting one large k seems reasonable since the k-hop neighborhoods include the k'-hop structures of all smaller k's. Against this intuition, we present a theoretical analysis that the $\texttt{WLS}^{k+1}$ histogram is not strictly more expressive than the $\texttt{WLS}^k$ histogram.

In Proposition 3.2, we show that non-equivalent colorings of two subgraphs in $\texttt{WLS}^{k+1}$ do not guarantee non-equivalent colorings in $\texttt{WLS}^k$. This is also true for the inverse: equivalent colorings in $\texttt{WLS}^{k+1}$ do not guarantee equivalent colorings in $\texttt{WLS}^k$. We obtain the same conclusion as Proposition 3.2 for GNNs as powerful as the WL test (e.g., Wang & Zhang (2022)), and some recent models are based on even less powerful GNNs than the WL test (e.g., Kim & Oh (2024)).

Proposition 3.2. *Given two subgraphs $\mathcal{S}_1$ and $\mathcal{S}_2$ of a global graph $\mathcal{G}$ and T iterations,*

$$\texttt{WLS}^{k+1}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k+1}(\mathcal{S}_2) \not\Rightarrow \texttt{WLS}^{k}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k}(\mathcal{S}_2), \quad (1)$$

$$\texttt{WLS}^{k+1}(\mathcal{S}_1) \equiv \texttt{WLS}^{k+1}(\mathcal{S}_2) \not\Rightarrow \texttt{WLS}^{k}(\mathcal{S}_1) \equiv \texttt{WLS}^{k}(\mathcal{S}_2), \quad (2)$$

for any $k < T$ where $\texttt{WLS}^k(\mathcal{S}) := \texttt{WLS}^k(\mathcal{S}, \mathcal{G}, T)$ and '$\equiv$' denotes the equivalence of colorings.

Proof. We will prove both statements by contradiction.

Proof of Equation 1 For the sake of contradiction, assume that whenever $\texttt{WLS}^{k+1}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$, it must follow that $\texttt{WLS}^{k}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k}(\mathcal{S}_2)$. Consider two subgraphs $\mathcal{S}_1$ and $\mathcal{S}_2$ of a global graph $\mathcal{G}$ such that their k-hop neighborhoods are isomorphic, i.e., $\mathcal{S}^k_1 \equiv \mathcal{S}^k_2$, but their $(k+1)$-hop neighborhoods have non-identical subtree patterns of height-T rooted at subgraphs. That is, within the k-hop radius, $\mathcal{S}_1$ and $\mathcal{S}_2$ have identical structures, but beyond that, their structures are distinguishable by the 1-WL algorithm (i.e., distinct colorings). This implies that $\texttt{WLS}^{k}(\mathcal{S}_1) \equiv \texttt{WLS}^{k}(\mathcal{S}_2)$, but $\texttt{WLS}^{k+1}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$ (e.g., the top part in Figure 2). This contradicts our assumption that $\texttt{WLS}^{k+1}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$ implies $\texttt{WLS}^{k}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k}(\mathcal{S}_2)$.

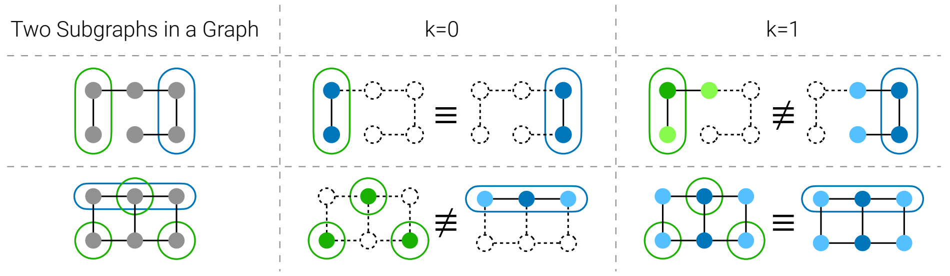


Figure 2: Example pairs of subgraphs that $\texttt{WLS}^k$ produces equivalent colorings while $\texttt{WLS}^{k+1}$ does not, and vice versa (where $k = 0$). The gray area represents the subgraph.

Proof of Equation 2 For the sake of contradiction, assume that whenever $\texttt{WLS}^{k+1}(\mathcal{S}_1) \equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$, it must follow that $\texttt{WLS}^{k}(\mathcal{S}_1) \equiv \texttt{WLS}^{k}(\mathcal{S}_2)$. Let $\mathcal{G}$ be a global graph, and consider its two subgraphs $\mathcal{S}_1$ and $\mathcal{S}_2$. Suppose that in $\mathcal{G}$, $(k+1)$-hop neighborhoods of $\mathcal{S}_1$ and $\mathcal{S}_2$ are identical, $\texttt{WLS}^{k+1}(\mathcal{S}_1) \equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$. However, within the k-hop neighborhoods, the local structures can differ such that the rooted subtree patterns of $\mathcal{S}_1$ and $\mathcal{S}_2$ up to height $T > k$ are not identical, $\texttt{WLS}^{k}(\mathcal{S}_1) \not\equiv \texttt{WLS}^{k}(\mathcal{S}_2)$ (e.g., the bottom part in Figure 2). This contradicts our assumption that $\texttt{WLS}^{k+1}(\mathcal{S}_1) \equiv \texttt{WLS}^{k+1}(\mathcal{S}_2)$ implies $\texttt{WLS}^{k}(\mathcal{S}_1) \equiv \texttt{WLS}^{k}(\mathcal{S}_2)$. □

Proposition 3.2 demonstrates that $\texttt{WLS}^k$ cannot represent the structural information of a smaller k'-hop structure ($k' < k$) from the perspective of graph isomorphism. This limitation suggests that relying solely on a single k for $\texttt{WLS}^k$ may be insufficient for encoding comprehensive information from various levels of structures. To address this, we propose combining $\texttt{WLS}^k$ for multiple values of k, allowing the representation to capture both local and global structures effectively. Building on this insight, we design WLKS-$\mathbb{K}$, a mixture of WLKS for multiple hops $k \in \mathbb{K}$ where its kernel matrix $\boldsymbol{K}_{\texttt{WLS}}$-$\mathbb{K}$ is a linear combination of $\boldsymbol{K}^k_{\texttt{WLS}}$.

$$\boldsymbol{K}_{\texttt{WLS}}\text{-}\mathbb{K} = \textstyle\sum_{k \in \mathbb{K}} \alpha_k \boldsymbol{K}^k_{\texttt{WLS}} \quad \text{where } \alpha_k \in \mathbb{R}^+. \tag{3}$$

Note that WLKS-$\mathbb{K}$ can be defined even when only one k is used (e.g., $\boldsymbol{K}_{\texttt{WLS}}\text{-}\{0\} = \alpha_0 \boldsymbol{K}^0_{\texttt{WLS}}$ for WLKS-$\{0\}$). WLKS-$\mathbb{K}$ is a valid kernel since a positive linear combination of p.s.d. kernels is p.s.d. (Shervashidze et al., 2011).

3.4 Selecting k for Minimal Complexity

In WLKS, selecting appropriate values of k during the k-hop subgraph sampling is crucial for balancing expressive power and complexity. As the number of nodes in the k-hop neighborhood grows exponentially with increasing k (Hamilton et al., 2017), an unbounded increase in k can result in substantial computation and memory overhead. To mitigate this, we strategically limit the choice of k to two specific values: $k = 0$ and $k = D$, where D is the diameter of the global graph $\mathcal{G}$.

When $k = 0$, the WLKS consumes the least computation and memory by using only the internal structure of the subgraph without neighborhood sampling. In contrast, when k is set to diameter D, every subgraph has the same k-hop neighborhood, which is the global graph $\mathcal{G}$; thus, the WLS is performed just once on $\mathcal{G}$ without per-subgraph computations. By using 0 and D, WLKS-$\{0, D\}$ can capture both the local and the largest global structure of subgraphs. This approach offers a practical model that balances expressive power and efficiency, avoiding excessive computation and memory consumption from intermediate k values.

3.5 Computational Complexity

The original WL Kernel has a computational complexity of $\mathcal{O}\left(T \sum_i E^{\text{sub}}_i + MT \sum_i N^{\text{sub}}_i\right)$ for M subgraphs, T iterations, and the number of nodes N^{sub}_i and edges E^{sub}_i of the subgraph i (Shervashidze & Borgwardt, 2009). When k is 0, a set of subgraphs is identical to a set of individual graphs, so its complexity is the same as the original's. When k is D, after performing the WL algorithm on the global graph once (i.e., $\mathcal{O}(TE)$), the coloring of each subgraph is aggregated to a histogram (i.e., $\mathcal{O}(\sum_i N_i^{\text{sub}})$). Thus, the computational complexity of WLKS-$\{0, D\}$ is $\mathcal{O}\left(T(E + \sum_i E_i^{\text{sub}}) + MT\sum_i N_i^{\text{sub}}\right)$.

We note that WLKS-$\{0, D\}$ do not perform k-hop neighborhood sampling, which adds a complexity of $\mathcal{O}(N^{\text{sub},k} + E^{\text{sub},k})$ per subgraph from a breadth-first search from $\mathbb{V}^{\text{sub}}$. Learning SVM with pre-computed kernels has a complexity of $\mathcal{O}(M^2)$ dependent on the number of subgraphs M, but this step is typically secondary to the WLS in practice.

4 Incorporating Continuous Features for WLKS

WLKS is designed to capture structural information but it can be simply integrated with continuous features. This section introduces four methods to incorporate continuous features for WLKS.

4.1 Combination with Kernels on Continuous Features

WLKS can be linearly combined with kernel matrices $\boldsymbol{K_X}$ derived from continuous features as in Equation 4. This combination enables the model to account for structure and feature similarities between subgraphs. One straightforward way is to directly compute a kernel on features, which measures the similarity between subgraphs based on their feature vectors. Another approach involves applying the Continuous Weisfeiler-Lehman operator (Togninalli et al., 2019) to features, generating a kernel matrix. This operator extends the original WL framework to include continuous feature spaces and enhances structural information with interactions from continuous features.

$$\alpha_{\text{structure}} \cdot \boldsymbol{K}_{\texttt{WLS}}\text{-}\mathbb{K} + \alpha_{\text{feature}} \cdot \boldsymbol{K_X} \quad \text{where } \alpha_{\cdot} \in \mathbb{R}^{+}. \tag{4}$$

In both cases, the kernel matrix from continuous features tends to be denser and has a different scale compared to those from the WL histogram. To address this, we apply standardization pre-processing and use the RBF and linear kernels.

4.2 GNNs with the WLKS Kernel Matrix as Adjacency Matrix

Another way to integrate features with WLKS is to use the kernel matrix as an adjacency matrix. Specifically, we consider the WLKS kernel matrix $\boldsymbol{K}_{\texttt{WLS}}$ as the adjacency matrix of a weighted graph where subgraphs $\mathbb{S}$ serve as nodes. The rationale for this approach is that a kernel represents the similarity between data points.

In this graph, the edge weight between subgraphs i and j corresponds to $\boldsymbol{K}_{\texttt{WLS}}[i, j]$. By applying deep GNNs to this graph, we can leverage the expressive power of WLKS for structural information and the capabilities of GNNs for feature representation. For this paper, we adopt state-of-the-art GNN-based models, S2N+0 and S2N+A (Kim & Oh, 2024), for the graph created by WLKS-$\{0, D\}$ as an instantiation of this approach.

Given the original feature $\boldsymbol{X} \in \mathbb{R}^{N \times \text{\# features}}$, in S2N+0, the hidden feature $\boldsymbol{H} \in \mathbb{R}^{M \times \text{\# features}}$ is a sum of original features in the subgraph, and then a GNN on $\boldsymbol{K}_{\texttt{WLS}} \in \mathbb{R}^{M \times M}$ is applied to get the logit matrix $\boldsymbol{Y} \in \mathbb{R}^{M \times \text{\# classes}}$ for the prediction. S2N+A first encodes each subgraph as an individual graph with a GNN, readout its output to get the hidden feature $\boldsymbol{H}$, then the other GNN on $\boldsymbol{K}_{\texttt{WLS}}$ is applied for the prediction. Formally,

$$\text{WLKS for S2N+0:} \quad \boldsymbol{H}[i, :] = \mathbf{1}_{N^{\text{sub}}}^{\top} \boldsymbol{X}[\mathbb{V}_i^{\text{sub}}, :], \quad \boldsymbol{Y} = \text{GNN}(\boldsymbol{H}, \boldsymbol{K}_{\texttt{WLS}}), \tag{5}$$

$$\text{WLKS for S2N+A:} \quad \boldsymbol{H}[i, :] = \mathbf{1}_{N^{\text{sub}}}^{\top} \text{GNN}_1(\boldsymbol{X}[\mathbb{V}_i^{\text{sub}}, :], \mathbb{A}_i^{\text{sub}}), \quad \boldsymbol{Y} = \text{GNN}_2(\boldsymbol{H}, \boldsymbol{K}_{\texttt{WLS}}), \tag{6}$$

where $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ is a vector of ones. Since the kernel matrix is dense for GPUs, we sparsify and normalize it using the same method in the S2N's paper.

5 Experiments

This section outlines the experimental setup, covering the datasets, training details, and baselines.

Table 1: Statistics of real-world and synthetic datasets.

	PPI-BP	HPO-Neuro	HPO-Metab	EM-User	Density	Cut-Ratio	Coreness	Component
# nodes in $\mathcal{G}$	17,080	14,587	14,587	57,333	5,000	5,000	5,000	19,555
# edges in $\mathcal{G}$	316,951	3,238,174	3,238,174	4,573,417	29,521	83,969	118,785	43,701
# subgraphs ($\mathcal{S}$)	1,591	2,400	4,000	324	250	250	221	250
# nodes / $\mathcal{S}$	$10.2_{\pm 10.5}$	$14.4_{\pm 6.2}$	$14.8_{\pm 6.5}$	$155.4_{\pm 100.2}$	$20.0_{\pm 0.0}$	$20.0_{\pm 0.0}$	$20.0_{\pm 0.0}$	$74.2_{\pm 52.8}$
# components / $\mathcal{S}$	$7.0_{\pm 5.5}$	$1.6_{\pm 0.7}$	$1.5_{\pm 0.7}$	$52.1_{\pm 15.3}$	$3.8_{\pm 3.7}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$4.9_{\pm 3.5}$
Density ($\mathcal{G}$)	0.0022	0.0304	0.0304	0.0028	0.0024	0.0067	0.0095	0.0002
Avg. density ($\mathcal{S}$)	0.216	0.757	0.767	0.010	0.232	0.945	0.219	0.150
# classes	6	10	6	2	3	3	3	2
Labels	Single	Multi	Single	Single	Single	Single	Single	Single
Dataset splits	80/10/10	80/10/10	80/10/10	70/15/15	80/10/10	80/10/10	80/10/10	80/10/10

Datasets We employ four real-world datasets (PPI-BP, HPO-Neuro, HPO-Metab, and EM-User) and four synthetic datasets (Density, Cut-Ratio, Coreness, and Component) introduced by Alsentzer et al. (2020). Given the global graph $\mathcal{G}$ and subgraphs $\mathbb{S}$, the goal of the real-world benchmark is subgraph classification on various domains: protein-protein interactions (PPI-BP), medical knowledge graphs (HPO-Neuro and HPO-Metab), and social networks (EM-User). For synthetic benchmarks, the goal is to determine the structural properties (density, cut ratio, the average core number, and the number of components) formulated as a classification. Note that WLKS does not need pretrained embeddings. We summarize dataset statistics in Table 1.

Models We experiment with five WLKS-$\mathbb{K}$ where $\mathbb{K}$ is $\{0\}, \{1\}, \{2\}, \{D\}, \{0, D\}$. Coefficients α is set to 1 when one k is selected, and $\alpha_0 + \alpha_D = 1$ for WLKS-$\{0, D\}$. We do a grid search of five hyperparameters: the number of iterations ($\{1, 2, 3, 4, 5\}$), whether to combine kernels of all iterations, whether to normalize histograms, L2 regularization ($\{2^3/100, 2^4/100, ..., 2^{14}/100\}$), and the coefficient $\alpha_0(\{0.999, 0.99, 0.9, 0.5, 0.1, 0.01, 0.001\})$. For fusing WLKS-$\{0, D\}$ to S2N, we follow the GCNII-based (Chen et al., 2020) architecture and settings presented in Kim & Oh (2024).

Baselines We use state-of-the-art GNN-based models for subgraph classification tasks as baselines: Subgraph Neural Network (SubGNN; Alsentzer et al., 2020), GNN with LAbeling trickS for Subgraph (GLASS; Wang & Zhang, 2022), Variational Subgraph Autoencoder (VSubGAE; Liu et al., 2023), Stochastic Subgraph Neighborhood Pooling (SSNP; Jacob et al., 2023) and Subgraph-To-Node Translation (S2N; Kim & Oh, 2024). Baseline results are taken from the corresponding research papers.

Efficiency Measurement When measuring the complete training time, we run models of the best hyperparameters from each model's original code, including batch sizes and total epochs, using Intel(R) Xeon(R) CPU E5-2640 v4 and a single GeForce GTX 1080 Ti (for deep GNNs).

Implementation All models are implemented with PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). We use the implementation of Support Vector Machines (SVMs) in Scikit-learn (Pedregosa et al., 2011).

6 Results and Discussions

In this section, we compare the classification performance and efficiency of WLKS and baselines. In addition, the performance of WLKS according to $\mathbb{K}$ is demonstrated to exhibit the usefulness of the kernel combination. Finally, we investigate how structure and features across subgraph datasets affect downstream performance by fusing WLKS and GNNs.

Performance and Efficiency In Table 2, the classification performance of WLKS-$\{0, D\}$ and baselines on eight datasets is summarized. Our results show that our model outperforms the best-performing baseline in five out of eight datasets. Specifically, WLKS-$\{0, D\}$ achieves the highest micro F1-score on PPI-BP, EM-User, Density, Coreness, and Component. For HPO-Neuro, HPO-Metab,

Table 2: Mean performance in micro F1-score on real-world and synthetic datasets over 10 runs. A subscript indicates the standard deviation. The higher the performance, the darker the blue color. The results of baselines are reprinted from respective papers.

Model	PPI-BP	HPO-Neuro	HPO-Metab	EM-User	Density	Cut-Ratio	Coreness	Component
SubGNN	$59.9_{\pm 2.4}$	$63.2_{\pm 1.0}$	$53.7_{\pm 2.3}$	$81.4_{\pm 4.6}$	$91.9_{\pm 1.6}$	$62.9_{\pm 3.9}$	$65.9_{\pm 9.2}$	$95.8_{\pm 9.8}$
GLASS	$61.9_{\pm 0.7}$	$68.5_{\pm 0.5}$	$61.4_{\pm 0.5}$	$88.8_{\pm 0.6}$	$93.0_{\pm 0.9}$	$93.5_{\pm 0.6}$	$84.0_{\pm 0.9}$	$100.0_{\pm 0.0}$
VSubGAE	-	$65.2_{\pm 1.4}$	$56.3_{\pm 0.9}$	$85.0_{\pm 3.5}$	-	-	-	-
SSNP-NN	$63.6_{\pm 0.7}$	$68.2_{\pm 0.4}$	$58.7_{\pm 1.0}$	$88.8_{\pm 0.5}$	-	-	-	-
S2N+0 $_{\text{GCNII}}$	$63.5_{\pm 2.4}$	$66.4_{\pm 1.1}$	$61.6_{\pm 1.7}$	$86.5_{\pm 3.2}$	$67.2_{\pm 2.4}$	$56.0_{\pm 0.0}$	$57.0_{\pm 4.9}$	$100.0_{\pm 0.0}$
S2N+A $_{\text{GCNII}}$	$63.7_{\pm 2.3}$	$68.4_{\pm 1.0}$	$63.2_{\pm 2.7}$	$89.0_{\pm 1.6}$	$93.2_{\pm 2.6}$	$56.0_{\pm 0.0}$	$85.7_{\pm 5.8}$	$100.0_{\pm 0.0}$
WLKS-$\{0, D\}$	$64.8_{\pm 0.0}$	$65.3_{\pm 0.0}$	$57.9_{\pm 0.0}$	$91.8_{\pm 0.0}$	$96.0_{\pm 0.0}$	$60.0_{\pm 0.0}$	$91.3_{\pm 0.0}$	$100.0_{\pm 0.0}$

Table 3: Runtime in seconds for the entire training stage and 1-epoch inference on the validation set for our model and baselines on real-world datasets.

Stage	Entire Training				Inference (1 epoch)			
Model	PPI-BP	HPO-Neuro	HPO-Metab	EM-User	PPI-BP	HPO-Neuro	HPO-Metab	EM-User
SubGNN	N/A	1798.2	1082.1	108.1	N/A	432.9	257.1	35.8
GLASS	1009.6	2462.6	1397.0	4597.4	8.2	27.0	26.4	39.0
S2N+0 $_{\text{GCNII}}$	16.7	36.7	37.1	31.0	9.9	9.3	8.3	14.6
S2N+A $_{\text{GCNII}}$	14.9	78.0	72.2	39.0	8.4	11.1	9.6	13.4
WLKS-{0,D}	3.5	25.2	10.9	9.6	1.0	11.7	2.7	1.8

and Cut-Ratio, our model shows similar performance to SubGNN but relatively lower performance than the state-of-the-art model.

In terms of efficiency, we present the training and inference time of our model and four representative baselines on real-world datasets in Table 3. Specifically, we measure the runtime for the entire training stage (including data and model loading, training steps, validation steps, and logging) and 1-epoch inference on the validation set. Note that an experiment on PPI-BP with SubGNN cannot be conducted since it takes more than 48 hours in pre-computation. WLKS-$\{0, D\}$ demonstrates significantly faster training and inference times across all real-world datasets compared to other models (e.g., the shorter training time of $\times 0.01$ – $\times 0.25$ and inference time of $\times 0.12$ – $\times 0.43$). This metric does not include the pre-computation or embedding pretraining required in baselines, so the actual training of WLKS is more efficient. Additionally, WLKS does not require a GPU in training and inference, unlike other GNN baselines.

Our results highlight the balance between efficiency and representation quality of WL histograms. Despite the loss of structural information from simplicity, the empirical evidence underscores the task-relevant nature of the retained information in WLKS-$\{0, D\}$. This is reflected in the superior performance of WLKS-$\{0, D\}$ on five out of eight datasets with significantly less training and inference time. However, the WL histogram lacks expressiveness in some subgraph tasks. The expressiveness can be enhanced by using the structural encoding (or labeling tricks) (Zhang & Chen, 2018; Li et al., 2020; Zhang et al., 2021; Dwivedi et al., 2022) in these cases. For example, on the Cut-Ratio dataset, where the performance of WLKS is low compared to the state-of-the-art, a linear combination with the inner product kernel of Random Walk Structural Encoding (Dwivedi et al., 2022) significantly improves the performance from 60.0 to 96.0. However, this improvement is not observed on other datasets. Detailed discussions are in Appendix D.

Performance of WLKS-$\mathbb{K}$ by $\mathbb{K}$ We highlight the importance of selecting the appropriate $\mathbb{K}$ in Table 4. Specifically, the performance of WLKS-$\mathbb{K}$ varies significantly depending on the choice of $\mathbb{K}$. WLKS-$\{0, D\}$, which combines kernels of 0 and D, consistently delivers strong results across datasets. WLKS-$\{0\}$ and WLKS-$\{D\}$ perform well independently in certain datasets, but their combination makes the better performance. This is clearly demonstrated in the Coreness experiment for predicting the average core number of nodes within a subgraph. This benchmark requires modeling internal structures, global structures, and subgraph positions. A significant improvement when combining kernels on Coreness aligns with our research motivation of capturing arbitrary interactions

Table 4: Mean performance of WLKS-$\mathbb{K}$ in micro F1-score by $\mathbb{K}$: $\{0\}$, $\{1\}$, $\{2\}$, $\{D\}$, and $\{0, D\}$. The standard deviations are omitted (all 0). The higher the performance, the darker the blue color.

Model	PPI-BP	HPO-Neuro	HPO-Metab	EM-User	Density	Cut-Ratio	Coreness	Component
WLKS-$\{0, D\}$	64.8	65.3	57.9	91.8	96.0	60.0	91.3	100.0
WLKS-$\{0\}$	34.0	31.4	26.4	67.3	96.0	36.0	87.0	100.0
WLKS-$\{1\}$	39.0	OOM	OOM	79.6	68.0	56.0	39.1	100.0
WLKS-$\{2\}$	64.2	OOM	OOM	89.8	68.0	56.0	39.1	100.0
WLKS-$\{D\}$	64.2	65.1	57.9	89.8	68.0	56.0	39.1	100.0

Table 5: Mean performance in micro F1-score of WLKS variants integrated with continuous features over 10 runs. Used S2N models are based on GCNII (Chen et al., 2020). The higher the performance, the darker the blue color.

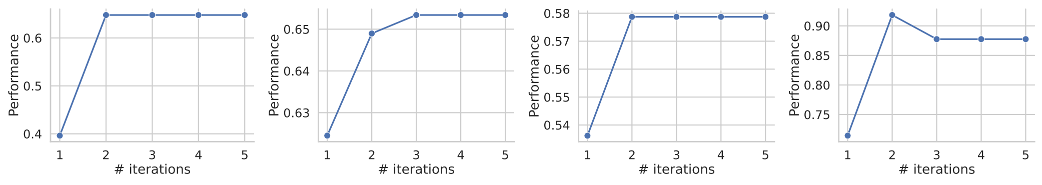
Model	PPI-BP	HPO-Neuro	HPO-Metab	EM-User	Density	Cut-Ratio	Coreness	Component
WLKS-$\{0, D\}$	$64.8_{\pm 0.0}$	$65.3_{\pm 0.0}$	$57.9_{\pm 0.0}$	$91.8_{\pm 0.0}$	$96.0_{\pm 0.0}$	$60.0_{\pm 0.0}$	$91.3_{\pm 0.0}$	$100.0_{\pm 0.0}$
WLKS-$\{0, D\}$ + Kernel on $\boldsymbol{X}$	$64.8_{\pm 0.0}$	$65.8_{\pm 0.0}$	$59.1_{\pm 0.0}$	$91.8_{\pm 0.0}$	$96.0_{\pm 0.0}$	$64.0_{\pm 0.0}$	$91.3_{\pm 0.0}$	$100.0_{\pm 0.0}$
WLKS-$\{0, D\}$ + Cont. WL Kernel on $\boldsymbol{X}$	$64.8_{\pm 0.0}$	$66.0_{\pm 0.0}$	$59.6_{\pm 0.0}$	$93.9_{\pm 0.0}$	$96.0_{\pm 0.0}$	$64.0_{\pm 0.0}$	$91.3_{\pm 0.0}$	$100.0_{\pm 0.0}$
WLKS-$\{0, D\}$ for S2N+0	$64.8_{\pm 1.5}$	$66.3_{\pm 0.6}$	$62.4_{\pm 1.1}$	$86.5_{\pm 2.4}$	$92.0_{\pm 0.0}$	$51.2_{\pm 3.9}$	$69.6_{\pm 1.9}$	$100.0_{\pm 0.0}$
WLKS-$\{0, D\}$ for S2N+A	$65.4_{\pm 2.4}$	$68.4_{\pm 1.1}$	$62.9_{\pm 1.9}$	$90.0_{\pm 3.3}$	$95.6_{\pm 2.8}$	$48.0_{\pm 0.0}$	$87.4_{\pm 4.1}$	$100.0_{\pm 0.0}$

between and within subgraph structures. This ability is necessary when multiple k-hop neighborhoods are associated with the labels of the subgraph, and the performance can be improved from the complementary nature of WLKS capturing different k-hop structures.

The selection of k is analogous to determining the number of layers in GNNs and can be treated as a hyperparameter optimized for specific tasks. Intermediate values of k may capture important substructures in datasets with large diameters. However, the model empirically performs well even when k is substantially smaller than the graph's diameter D. For instance, PPI-BP's largest component has a diameter of 8, yet $k = 2$ performs as well as $k = D$. In addition, our empirical results showed no clear relation between the size of the graph, its global density, or the average density of subgraphs (as presented in Table 1) and task performance with different k values. This suggests that such structural properties are not key factors in determining the optimal k. Instead, the nature of the task and its structural requirements should guide the selection of k.

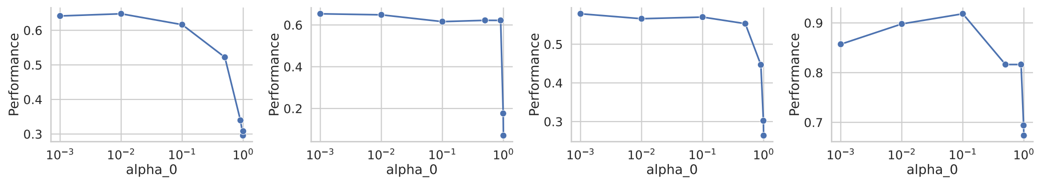
Performance of WLKS Variants Integrated with Continuous Features Table 5 presents the results of WLKS variants combining WLKS-$\{0, D\}$ and continuous features. The performance of WLKS-$\{0, D\}$ for S2N improves over vanilla WLKS-$\{0, D\}$ on PPI-BP, HPO-Neuro, and HPO-Metab. However, performance decreases on EM-User, Density, Cut-Ratio, and Coreness. Applying GNNs to the WLKS kernel matrix requires kernel sparsification, which leads to some loss of structural information. The enhanced features provided by deep neural networks can mitigate this trade-off. We interpret that the former set of benchmarks prioritizes features over structure, while the latter relies more on structural information. Using kernels on continuous features improves performance on HPO-Neuro, HPO-Metab, EM-User, and Cut-Ratio. Notably, on EM-User, it achieves the best performance of 93.9 among all methods. For other datasets, no changes in performance are observed. Unlike the combination of GNN-based models, which can leverage the feature processing of neural networks, kernels on features seem to provide limited performance gains from features.

Sensitivity Analysis of Hyperparameters Figures 3 and 4 demonstrate the performance sensitivity of the WLKS-$\{0, D\}$ with respect to the number of iterations T and the kernel mixing coefficient α_0. The best performance is achieved at iterations of $T = 2$ or $T = 3$, beyond which the WL coloring stabilizes and no further improvement is observed. For α_0, the WLKS-$\{0, D\}$ is best-performed between 10^{-3} and 10^{-1}, while performance drops sharply as α_0 approaches 1. Since $\alpha_D = 1 - \alpha_0$ is larger than α_0 in this range, this suggests that the subgraph labels rely more on global structures ($k = D$) than internal ones ($k = 0$).



(a) PPI-BP (b) HPO-Neuro (c) HPO-Metab (d) EM-User

Figure 3: Performance of WLKS-$\{0, D\}$ by the number of iterations T.



(a) PPI-BP (b) HPO-Neuro (c) HPO-Metab (d) EM-User

Figure 4: Performance of WLKS-$\{0, D\}$ by the coefficient α_0 in $\alpha_0 \boldsymbol{K}^0_{\texttt{WLS}} + (1 - \alpha_0)\boldsymbol{K}^D_{\texttt{WLS}}$.

7 Conclusion

We proposed WLKS, a simple but powerful model for subgraph-level tasks that generalizes the Weisfeiler-Lehman (WL) kernel on induced k-hop neighborhoods. WLKS can enhance expressiveness by linearly combining kernel matrices from multiple k-hop levels, capturing richer structural information without redundant neighborhood sampling. Through extensive experiments on eight real-world and synthetic benchmarks, WLKS outperformed state-of-the-art methods on five datasets with reduced training times—ranging from $\times 0.01$ to $\times 0.53$ compared to existing models. Furthermore, WLKS does not need pre-computation, pre-training, GPUs, or extensive hyperparameter tuning.

Our method offers a promising and accessible alternative to GNN-based approaches for subgraph representation learning, but some tasks can still benefit from deep neural networks. We leave as future work the seamless integration of WLKS with GNNs to leverage the expressive power of both structures and features.

References

Emily Alsentzer, Samuel G Finlayson, Michelle M Li, and Marinka Zitnik. Subgraph neural networks. *Proceedings of Neural Information Processing Systems, NeurIPS*, 2020.

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

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

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

Radin Hamidi Rad, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. Subgraph representation learning for team mining. In *Proceedings of the 14th ACM Web Science Conference 2022*, pp. 148–153, 2022.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Shweta Ann Jacob, Paul Louis, and Amirali Salehi-Abari. Stochastic subgraph neighborhood pooling for subgraph classification. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 3963–3967, 2023.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 321–328, 2003.

Dongkwan Kim and Alice Oh. Translating subgraphs to nodes makes simple GNNs strong and efficient for subgraph representation learning. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=xSizvCoI79`.

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

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 291–298, 2012.

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

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Shufei Li, Pai Zheng, Shibao Pang, Xi Vincent Wang, and Lihui Wang. Self-organising multiple human–robot collaboration: A temporal subgraph reasoning-based method. *Journal of Manufacturing Systems*, 68:304–312, 2023.

Chang Liu, Yuwen Yang, Zhe Xie, Hongtao Lu, and Yue Ding. Position-aware subgraph neural networks with data-efficient learning. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pp. 643–651, 2023.

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

Paridhi Maheshwari, Hongyu Ren, Yanan Wang, Rok Sosic, and Jure Leskovec. Timegraphs: Graph-based temporal reasoning. *arXiv preprint arXiv:2401.03134*, 2024.

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

Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*, 2016.

Shiyu Ouyang, Qianlan Bai, Hui Feng, and Bo Hu. Bitcoin money laundering detection via subgraph contrastive learning. *Entropy*, 26(3):211, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Nino Shervashidze and Karsten Borgwardt. Fast subtree kernels on graphs. *Advances in neural information processing systems*, 22, 2009.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in neural information processing systems*, 32, 2019.

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

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

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

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

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021.

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

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. URL `https://openreview.net/forum?id=Mspk_WYKoEH`.

A Formal Comparison with Representative Related Work

In this section, we compare WLKS with highly related prior work including Subgraph Neural Network (SubGNN; Alsentzer et al., 2020), GNN with LAbeling trickS for Subgraph (GLASS; Wang & Zhang, 2022), Subgraph-To-Node Translation (S2N; Kim & Oh, 2024), and GNN As Kernel (GNN-AK; Zhao et al., 2022).

While SubGNN employs message-passing within subgraphs, its reliance on ad hoc patch sampling and its separation of hand-crafted channels (e.g., position, neighborhood, structure) introduces complexity and potential sub-optimality in information aggregation. Without requiring handcrafted patch designs or sampling strategies, WLKS captures a unified and expressive structural representation based on the theoretical rationale that structures at multiple levels are important.

GLASS uses separate message-passing for node labels that distinguish internal and global structures. This can enhance expressiveness by mixing representations from local and global structures similar to WLKS. However, GLASS has a limited ability to handle multiple labels in batched subgraphs; thus, a small batch size is required for GLASS. WLKS provides a generalized framework to represent fine-grained levels of structures around subgraphs, which can process multiple subgraphs efficiently by leveraging kernel methods.

Conceptually, node labeling based on membership (Zhang et al., 2021; Wang & Zhang, 2022), distances (Zhang & Chen, 2018; Li et al., 2020) or structures (Dwivedi et al., 2022) can be integrated as additional features into the kernel methods. However, our current design may not benefit from existing node labeling techniques. Node labeling encodes information about a node's membership within a subgraph (e.g., GLASS) or its distance to the target structure (distance-based). When aggregated into histograms, GLASS's label-histograms capture distributions of sizes of the subgraph and its neighborhoods, and distance-based label-histograms capture the distribution of distances with the target within an enclosing subgraph. While such distributions are somewhat informative, size and distances represent coarser structural summaries compared to the fine-grained subtree patterns encoded by WL color histograms. The expressive power of WLKS arises from its ability to capture these detailed structural features across multiple k-hop neighborhoods, obviating the need for additional node-level labeling.

S2N efficiently learns subgraph representations by compressing the global graph. However, this compression results in a loss of structural information and expressiveness in tasks where the global structure is important. In particular, since the approximation bound of S2N depends on how many subgraphs are spread out in the global graph, we cannot always expect a robust approximation. In contrast, WLKS does not rely on lossy compression and can yield informative representations using efficient kernel methods.

GNN-AK generates a graph representation by aggregating the information of each node's locally induced encompassing subgraph. Although there are local and global interactions, there are fundamental differences between WLKS. First, GNN-AK is designed for graph-level tasks, so the interactions between the graph itself and its local subgraphs are modeled. However, dealing with subgraph-level tasks is more challenging since modeling both the inside and the outside of the subgraph is required. WLKS encodes them by using multiple k-hop kernels. Second, GNN-AK has a large complexity that depends on the total number of neighbors and the sum of edges between neighbors, so it cannot be applied to a large global graph, unlike WLKS. In fact, the average number of nodes covered by the GNN-AK paper is much smaller, ranging from 25 to 430. In these perspectives, our study takes a complementary approach to GNN-AK, addressing aspects not covered in their work.

B Discussion on Relations between WL Isomorphism Test and WL Kernels

In this section, we provide a detailed discussion to clarify the distinctions between Weisfeiler-Lehman (WL) isomorphism test and WL kernels, elaborating on how our work builds upon these foundational concepts.

The WL algorithm is recognized for testing graph isomorphism by iteratively refining node labels to capture the structural similarity between graphs. Its ability to distinguish non-isomorphic graphs

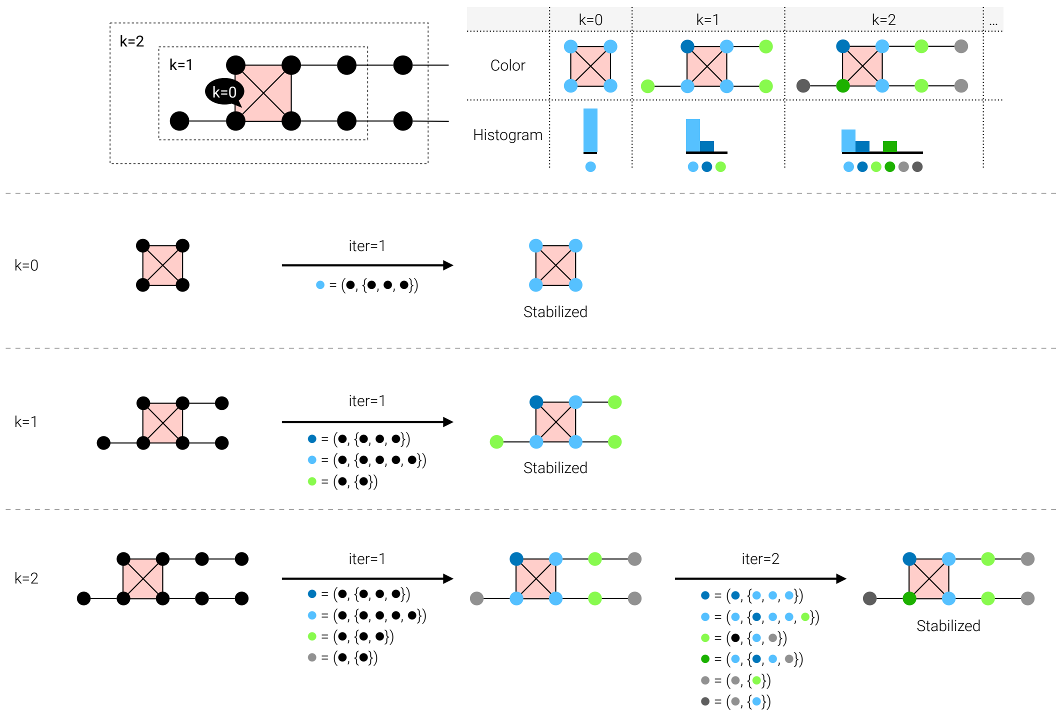


Figure 5: A step-by-step visualization of $\texttt{WLS}^k$ algorithm (Algorithm 2) for $k \in \{0, 1, 2\}$ using an example in Figure 1.

is often considered a benchmark for evaluating the expressiveness of graph representation methods. While isomorphism distinguishability is theoretically significant, it can be overly restrictive in practical applications where the exact topological equivalence of graphs is not the primary concern. For example, many real-world tasks involve identifying structural similarities between graphs that may not be strictly isomorphic but share functional or semantic similarities.

Our work aligns more closely with the WL kernel framework (Shervashidze & Borgwardt, 2009; Shervashidze et al., 2011), which extends the application of the WL algorithm beyond isomorphism testing. WL kernels compute graph similarity based on histogram representations of subtree patterns generated by the WL algorithm. These histograms serve as compact summaries of graph structure, allowing for the comparison of graphs even when they are not isomorphic. In this context, WL kernels prioritize capturing similarities between graphs over distinguishing isomorphic structures. This broader perspective makes WL kernels particularly suitable for various graph-structured data, where the goal is to quantify structural resemblance rather than to test for isomorphism.

Building upon the WL kernel framework, our work introduces the WLKS method, which leverages WL histograms as measures of subgraph similarity. The key insight here is that WL histograms provide a rich representation of subtree patterns within graphs, enabling a nuanced comparison of internal and external structures of subgraphs.

C Step-by-Step Visualization of $\texttt{WLS}^k$ Algorithm

In Figure 5, we visualize each iteration of $\texttt{WLS}^k$ algorithm using an example of Figure 1.

D Using Kernels of Distance or Structural Encoding

It is well-known that additional structural features (often called labeling tricks, distance encoding, or structural encoding) can enhance the expressiveness of message-passing mechanisms under certain conditions (Zhang & Chen, 2018; Li et al., 2020; Zhang et al., 2021; Dwivedi et al., 2022; Wang & Zhang, 2022).

We argue that these approaches can have different effectiveness for subgraph-level tasks and kernel-based methods:

- Zero-one labeling (Zhang et al., 2021; Wang & Zhang, 2022): This binary labeling (assigning 0 to internal nodes and 1 to external nodes) shows limited expressiveness when aggregating labels to histograms as kernel inputs. Its histogram is represented as a length-2 vector (0 or 1), which only counts the number of nodes inside and outside the subgraph, thereby omitting finer structural details.
- SEAL's Double-radius node labeling (Zhang & Chen, 2018): SEAL computes distances with respect to target structures (e.g., links) and can be applicable to k-hop neighborhoods of subgraphs but computationally challenging. While efficient for link prediction tasks due to the smaller size of enclosing subgraphs, extending this approach to general subgraphs becomes infeasible due to the computational overhead of calculating all pairwise distances.
- Distance Encoding (DE) (Li et al., 2020) and Random Walk Structural Encoding (RWSE) (Dwivedi et al., 2022): DE uses landing probabilities of random walks from nodes in the node set to a given node, and RWSE uses diagonal elements of random walk matrices. In this line of work, random walk matrices are shown to encode structures in an expressive way efficiently even on large-scale graphs.

We linearly combine WLKS-$\{0, D\}$ with an inner product kernel of RWSE (the walk length of 1 – 64) sum-aggregated per subgraph, yielding a significant performance boost on the Cut-Ratio dataset (from 60.0 to 96.0). However, this improvement was not shown across the other seven benchmarks we tested. We leave investigating which specific node labeling methods are effective, which aggregations of node labels are effective, and which kernels (e.g., linear, polynomial, RBF) best complement the specific node labeling as future work.