Random Walk Conformer: Learning Graph Representation from Long and Short Range

Pei-Kai Yeh, Hsi-Wen Chen, Ming-Syan Chen

National Taiwan University
{pei-kai, hwchen}@arbor.ee.ntu.edu.tw, mschen@ntu.edu.tw

Abstract

While graph neural networks (GNNs) have achieved notable success in various graph mining tasks, conventional GNNs only model the pairwise correlation in 1-hop neighbors without considering the long-term relations and the high-order patterns, thus limiting their performances. Recently, several works have addressed these issues by exploring the motif, i.e., frequent subgraphs. However, these methods usually require an unacceptable computational time to enumerate all possible combinations of motifs. In this paper, we introduce a new GNN framework, namely *Random Walk Conformer (RWC)*, to exploit global correlations and local patterns based on the random walk, which is a promising method to discover the graph structure. Besides, we propose *random walk encoding* to help RWC capture topological information, which is proven more expressive than conventional spatial encoding. Extensive experiment results manifest that RWC achieves state-of-the-art performance on graph classification and regression tasks. The source code of RWC is available at **https://github.com/b05901024/RandomWalkConformer**.

Introduction

Graph Neural networks (GNNs) (Kipf and Welling 2017; Veličković et al. 2018; Xu et al. 2019) serve as a promising tool for several graph mining problems, e.g., social network analysis (Dwivedi et al. 2020), and molecular design (Hu et al. 2021). Specifically, GNNs derive the node (or graph) embedding by iteratively aggregating the neighbors' features with the uniform weight (Kipf and Welling 2017) or the attention mechanism (Veličković et al. 2018) and utilize the learned representations for the downstream tasks. Another line of studies argued the existence and the correctness of the given graph topology, thus generalizing the GNN by Transformer (Rong et al. 2020; Ying et al. 2021), which has achieved celebrated success in modeling structured data in various domains, including natural language (Devlin et al. 2019), speech (Gulati et al. 2020), and computer vision (Dosovitskiy et al. 2021). Instead of only aggregating the node features along with the linked nodes, Transformer utilizes a self-attention mechanism, i.e., calculating the pairwise relation between every node, which is more powerful than conventional GNNs (Ying et al. 2021).

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

While the studies mentioned above mainly target the pairwise correlation between nodes, i.e., one-to-one, the high order pattern in graphs, i.e., many-to-many (Zhao et al. 2022), however, have not been fully discovered. Generally, most GNNs expressiveness is bounded by the first order Weisfeiler-Leman (1-WL) isomorphism test (Xu et al. 2019), which can not effectively solve some fundamental graph mining tasks, e.g., counting cycles or triangles (Chen et al. 2020). Therefore, several recent works empower GNN with motifs, i.e., frequent subgraphs (Zhou et al. 2021). Morris et al. (Morris et al. 2019) incorporate K-WL in the design of more powerful GNNs, and Zhao et al. (Zhao et al. 2022) leverage star graph as subgraph patterns in their GNN framework. However, enumerating all possible motifs is computationally extensive, which usually grows exponentially, i.e., $O(|G|^k)$ where $|G|$ is the size of the graph, and k is the size of the motif.[1] Meanwhile, the random walk serves as a promising technique to estimate the motif by checking only a tiny fraction of all the induced subgraphs (Han and Sethu 2016). In this paper, we aim to incorporate the random walk for motif discovery and encode these high-order patterns into GNN.

To effectively capture nodes correlation and estimate the motifs, we propose *Random Walk Conformer (RWC)*, including two major components, *Random Walk Self Attention Network (RW-SAN)* and the *Random Walk Convolution (RW-Conv)*. By processing nodes sampled by the random walk, RW-SAN measures the long-range correlation between the node pairs to aggregate the topological information in the whole random walk sequence. Then, RW-Conv encodes the short-range local structure by processing the convolutional module on the random walk sequence, which can be regarded as the graph kernels (Toenshoff et al. 2021).In contrast, conventional transformer ignores the high-order patterns and cannot correctly analyze the graph structure (Zhang and Li 2021; Zhao et al. 2022). Furthermore, we propose *Random Walk Encoding* to extract both topological and contextual information between two nodes based on the random walk sequences. In contrast, several previous works employ the shortest path to analyze the global structural information (Ying et al. 2021; Vaswani et al. 2017) to empower the capability of Trans-

[1] Counting all the subgraphs with size k has an upper bound of $O(|\mathcal{E}|^k)$ and a lower bound of $O(|\mathcal{V}|c^{k-1})$, where $\mathcal{V}, \mathcal{E}$ are the set of nodes and edges in the graph, and c is the average number of neighbors for each node (Itzhack, Mogilevski, and Louzoun 2007).

former for modeling the graph data. However, as there are probably many paths between these two nodes on the graph, a fixed shortest path may not correctly represent the relation. We also theoretically prove that by sampling random walk with enough length, random walk encoding is strictly more powerful than 1-WL and conventional spatial encoding based on the shortest path.

The contributions are summarized as follows.

- We propose a new graph learning framework, *Random Walk Conformer (RWC)*, which extracts the advantages from both Transformer-based and motif-based models with the graph conformer and random walk encoding.
- We theoretically prove that random walk encoding is strictly more powerful than 1-WL and spatial encoding based on the shortest path.
- Extensive experiments on eight benchmark datasets manifest that RWC achieves the best results on the leaderboards compared to several state-of-the-art methods on graph classification and regression tasks.

Related Work

Recently, Graph Neural Networks (GNNs) (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017) have gained significant attention from research fields, which derive node embedding by aggregating its own and neighboring node features for several graph learning tasks. While several recent works (Dwivedi and Bresson 2021; Ying et al. 2021) point out the existence and the correctness of the graph topology, several works learn the graph representation by Transformer, which encodes the entire node set for aggregation. Dwivedi et al. (Dwivedi and Bresson 2021) fuse node positional features using Laplacian eigenvectors, and they suggest that the attention mechanism should only aggregate the information from neighbors. Kreuzer et al. (Kreuzer et al. 2021) input the eigenfunctions of a graph and project them into a learned positional encoding, which allows the network to use up to the entire Laplace spectrum of each graph. Ying et al. (Ying et al. 2021) use encodings based on the shortest path in the graph to capture the relationship for node pairs and represent the graph. However, the studies mentioned above focus on the pair-wise correlation and ignore the high-order pattern on a graph, i.e., motif. Therefore, several works extend GNNs to exploit motifs to capture more complex subgraph structures (Morris et al. 2019; Maron et al. 2019b). Sankar et al. (Sankar, Zhang, and Chang 2017) and Lee et al., (Lee et al. 2019) employ convolution and attention mechanisms to integrate features extracted from predefined motifs. While the scalability of motif-based method GNNs is limited by exploring all motifs on the graph motifs (Han and Sethu 2016), Toenshoff et al. (Toenshoff et al. 2021) employ the random walk sequence and process them with 1D CNN to discover the sequential pattern on the graph. However, there are little works that extract both long-range relations of graphs and structural information from motifs.

Preliminary

In the following, we present our problem formulation following the conventional custom of notation. Scalars or elements of a set are denoted by italic lowercase letters, e.g., x and x_i, while vectors are denoted by boldface lowercase letters, e.g., $\mathbf{x}$. Matrices are represented by boldface capital letters, e.g., $\mathbf{X}$, and sets are denoted by calligraphic letters, e.g., $\mathcal{X}$.

Graph Classification/ Regression Tasks

Let $G = (\mathcal{V}, \mathcal{E})$ denote a graph, and $\mathbf{x}_i$ is the feature vector of node $v_i \in \mathcal{V}$. Given a set of graphs $\mathcal{G} = \{G_i\}_{i\in[1,N]}$ and their labels (for classification) or ground-truth values (for regression) $\mathcal{Y} = \{y_i\}_{i\in[1,N]}$, we aim to learn model parameters θ as follows.

$$\theta^* = \arg\min_{\theta} \sum_{i\in[1,N]} \mathcal{L}(f_\theta(G_i), y_i),$$

where $\mathcal{L}$ is a task-oriented loss function, e.g., cross entropy for classification or mean absolute error for regression.

Graph Neural Network

GNNs use the graph G and associated node features $\mathbf{X}$ to learn a representation vector of a node, $\mathbf{h}_v$, or the entire graph, $\mathbf{h}_G$. Modern GNNs follow a learning schema that iteratively updates the representation of a node by aggregating representations of its first or higher-order neighbors. We denote $\mathbf{h}_i^{(k)}$ as the representation of v_i at the k-th layer and initialize $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. Formally, the k-th layer of a GNN is

$$\begin{aligned} \mathbf{a}_i^{(k)} &= \text{AGGREGATE}^{(k)}\Big(\Big\{\mathbf{h}_j^{(k-1)}\Big| v_j \in \mathcal{N}(v_i)\Big\}\Big), \\ \mathbf{h}_i^{(k)} &= \text{COMBINE}^{(k)}\Big(\mathbf{h}_i^{(k-1)}, \mathbf{a}_i^{(k)}\Big), \end{aligned} \tag{1}$$

where $\mathcal{N}(v_i)$ denotes the set of neighbors of v_i. The AGGREGATE function is used to gather information from neighbors. Different architectures for AGGREGATE have been proposed, including uniform weight (Kipf and Welling 2017; Xu et al. 2019) or neural networks (Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018). The COMBINE function, which is usually a concatenation followed by a linear mapping, fuses the information from neighbors into the node representation.

Transformer

Limited to the existence and the correctness of graph topology, recent efforts utilize Transformer (Vaswani et al. 2017), which aggregates the node features along with every node by the self attention mechanism.

$$\begin{aligned} \mathbf{a}_i^{(k)} &= \frac{\sum_{v_j\in\mathcal{V}} exp(\mathbf{q}_i^{(k-1)\top}\mathbf{k}_j^{(k-1)})\mathbf{v}_j^{(k-1)}}{\sum_{v_j\in\mathcal{V}} exp(\mathbf{q}_i^{(k-1)\top}\mathbf{k}_j^{(k-1)})}, \\ \mathbf{h}_i^{(k)} &= \mathbf{a}_i^{(k)} + \alpha\mathbf{h}_i^{(k-1)} \end{aligned}$$

where the node embedding of v_i at the k-th layer is aggregated along with all nodes on the graph. $\mathbf{h}_i^{(k-1)}$ is projected by three matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ as the query $\mathbf{q}_i^{(k-1)}$, key $\mathbf{k}_i^{(k-1)}$, and value $\mathbf{v}_i^{(k-1)}$ vector, respectively. Note that the α is a trade-off parameter for the residual connection to prevent over-smoothing problem (Hamilton, Ying, and

Leskovec 2017). With self attention mechanism, Transformer can derive the node embedding v_i by considering every node $\mathcal{V}$ without the graph topology, i.e., $\mathcal{N}(v_i)$.

For graph classification or regression tasks, the READOUT function aggregates node features $\mathbf{h}_i^{(K)}$ from the final iteration to obtain the entire graph's representation $\mathbf{h}_G$.

$$\mathbf{h}_G = \text{READOUT}\Big(\Big\{\mathbf{h}_i^{(K)}\Big|v_i \in \mathcal{V}\Big\}\Big) \tag{2}$$

Since nodes are unordered in the graph, READOUT is a permutation invariant graph-level pooling function, e.g., summation (Xu et al. 2019) or attention (Baek, Kang, and Hwang 2021; Ying et al. 2018). Then, an additional classifier is trained to predict the label/value according to the embedding vector $\mathbf{h}_G$.

Method

In this section, we introduce a new GNN framework, namely *Random Walk Conformer (RWC)*, to capture the global pairwise relations and consider the local subgraph structures efficiently. First, we introduce the random walk generator with a novel spatial encoding to discover motifs. Then, we detail the design of the graph conformer. Fig. 1 illustrates the overall architecture of RWC.

Random Walk Encodings

While the output of GNN (Transformer) is permutations invariant to the input data, it is fundamental to encode the auxiliary topological information by spatial encoding (Ying et al. 2021). Therefore, previous works introduce spatial encoding to help the models exploit the global structure in local aggregation, which can be categorized into absolute spatial encoding (Vaswani et al. 2017) and relative spatial encoding (Ke, He, and Liu 2021). Absolute spatial encoding adds a vector based on sinusoidal functions to each of the input features by nodes' order, which is also sensitive to the sequence length (Shaw, Uszkoreit, and Vaswani 2018). Besides, nodes are permutations invariant in the graph. Thus, relative position encoding is proposed later (Ying et al. 2021) by adding the information depending on the relative distance of each node pair in the input sequence to the attention logits.

However, previous relative spatial encodings (Ying et al. 2021) are typically based on the shortest path between two nodes, which only considers a fixed single path on the graph. Therefore, we introduce *Random Walk Encodings*, which exploit the relative position between two nodes by multiple paths from the random walk. Generally, the random walk sequence $\mathcal{S} = \{s_i\}_{i=1}^{l}$ is a l-length node sequence sampled from the graph $G = \{\mathcal{V}, \mathcal{E}\}$, where $s_i \in \mathcal{V}$. A random walk sequence in a graph is generated by starting at some initial node s_1 and iteratively sampling the next node s_{i+1} randomly from the neighbors $\mathcal{N}(s_i)$ of the current node s_i. Since the nodes on a sparse graph are with a small degree, the uniform sampling on the neighborhood set tends to backtrack often and is challenging to discover long-range patterns. To explore further nodes, the random walk is sampled by the non-backtracking method, i.e., excluding the previous node unless it is the only neighbor. Thus, each node's predecessor and successor are forced to be different. [2]

Given the random walk sequence $\mathcal{S}$, the random walk distance $D(v_i, v_j)$ is defined as follows.

$$D(v_i, v_j) = \begin{cases} \min\Big(\Big\{|m-n|\Big|s_m = v_i \wedge s_n = v_j,\Big\}\Big), & \text{if } v_i, v_j \in \mathcal{S} \\ \infty, & \text{otherwise} \end{cases}, \tag{3}$$

which represents the shortest distance between v_i and v_j in the random walk sequence. Thus, we can find different paths by sampling multiple random walks to discover more diverse patterns between a node pair. Note that shortest path encoding can be regarded as a special case of the random walk encoding with infinity walk length l to cover the whole graph. Similar to other spatial encodings, we trained the embedding function $\rho^s_{i,j} = \psi^s(D(v_i, v_j))$, which maps the distance between v_i and v_j in Eq. (3) to a trainable parameter as our random walk encoding.[3]

By the random walk encoding, we can convert a random walk sequence, i.e., a simple line graph to a motif by utilizing the global topological information. While the random walk encoding can explore some specific substructure on the graph by comparing multiple random walks, the shortest path encoding only focus on the fixed pattern of graphs. For example, we can explore the substructures formed by nodes v_i and v_j such as k-cycles in molecular networks, where $D(v_i, v_j)$ is 0 in one sequence and k in another. However, it cannot be detected by the conventional relative spatial encoding with a single shortest path. We conclude the proof as follows.

Theorem 1 (Proof in Appendix)**.** *With a sufficient number of layers, random walk encoding is strictly more expressive than that based on a fixed shortest path.*

We also theoretically prove that random walk encoding is more powerful than 1-WL test.

Theorem 2 (Proof in Appendix)**.** *With a sufficient number of layers, random walk encoding is strictly more expressive than 1-WL test.*

Since the distance equal to 1 implies that the two nodes are their mutual 1-hop neighbor, with a sufficient number of layers or length of the random walk such that all the 1-hop neighbors are explored for each node, 1-WL test can be viewed as a special case of the random walk encoding. Theorem 1 implies that random walk encoding can provide the structural information to the Transformer-based model while current relative encoding methods cannot. Furthermore, since the expressiveness of most GNNs is bounded by 1-WL, Theorem 2 shows that random walk encoding is more powerful than most modern GNNs, which illustrates the expressiveness of random walk encoding.

[2]The expected length of the random walk to discover all nodes in the graph is $O(4|\mathcal{V}||\mathcal{E}|)$ (Abdullah 2012). Hence, for RWC, the random walk length l should be much larger than $|\mathcal{V}|$ to capture enough information from the nodes.

[3]Note that each layer can either train its own embedding function or share the parameter across all blocks.

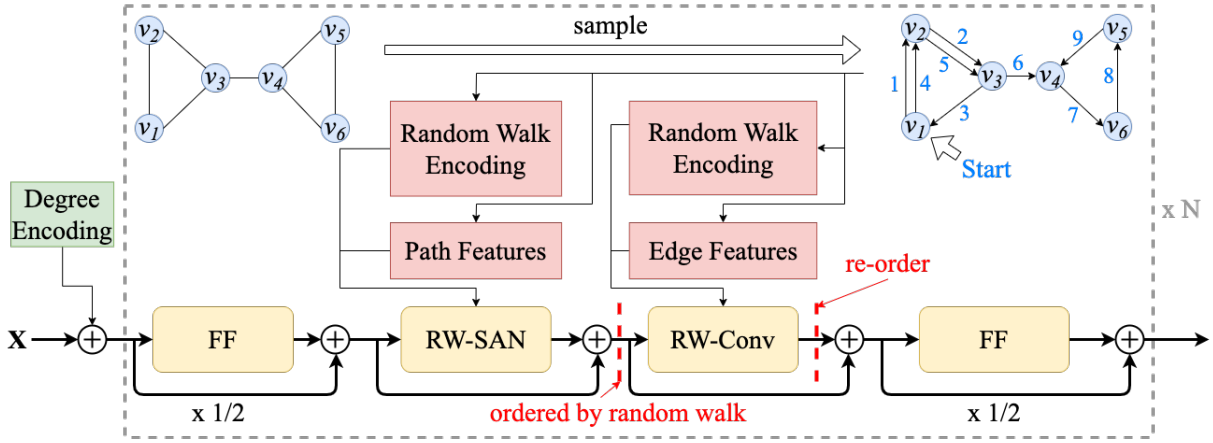

Figure 1: Random Walk Conformer (RWC). The model is composed of several Random Walk Conformer blocks. Each block consists of a Feed Forward Module (FF), a *Random Walk Self Attention Network (RW-SAN)*, a *Random Walk Convolution (RW-Conv)*, and a second FF. With random walk encoding, the structural information is injected into RWC.

Random Walk Conformer

While Transformer is good at modeling the global structure, they are less capable of extracting fine-grained local patterns. Therefore, Conformer (Gulati et al. 2020) is proposed by employing the convolution module to vanilla Transformer. Unlike the sequence data in NLP tasks that can easily pass the architecture stacked by attention module and convolution module, it is non-trivial to apply the Conformer architecture on graphs that usually have specific structures, i.e., motifs, rather than a simple sequence. To gather global structure and discover local motifs efficiently, we introduce a new GNN framework, namely *Random Walk Conformer (RWC)*, which consists of two major components *Random Walk Self Attention Network (RW-SAN)* to capture the global information among graphs by self attention and *Random Walk Convolution (RW-Conv)* to exploits the motifs.

Random Walk Self Attention Network Compared to conventional GNNs (Xu et al. 2019; Hamilton, Ying, and Leskovec 2017), Transformer (Vaswani et al. 2017) generalizes the GNNs by carefully analyzing the pairwise correlation between every node to deal with the limited graph topology (Dwivedi and Bresson 2021). By a single self attention layer, Transformer can potentially allow all the nodes of an input graph to communicate; however, the price to pay is that this core component does not take any topological structure of the graph into account. Besides, the time complexity of the Transformer is $O(|\mathcal{V}|^2 d)$, where d is the dimension of the feature, thus limiting the applicability of the Transformer, especially on a large graph. Therefore, for a large graph, one can utilize the random walk to sample a subset of nodes instead of considering the entire graph. *Random Walk Self Attention Network (RW-SAN)* is formulated as:

$$\mathbf{a}_i^{(k)} = \frac{\sum_{v_j \in \mathcal{S}} f_j \, exp(\mathbf{q}_i^{(k-1)\top} \mathbf{k}_j^{(k-1)} + \rho_{i,j}^s + \rho_{i,j}^p) \mathbf{v}_j^{(k-1)}}{\sum_{v_j \in \mathcal{S}} f_j \, exp(\mathbf{q}_i^{(k-1)\top} \mathbf{k}_j^{(k-1)} + \rho_{i,j}^s + \rho_{i,j}^p)}. \tag{4}$$

Note that f_j is the weight for the attention mechanism, we can either set it to 1 or reweight the node based on the occurrence of each node, which is the number of node v_j in the random walk sequence over the walk length l. Therefore, RW-SAN is able to estimate the statistics in the graph, i.e., the occurrence of a node in the random walk reflects its importance.

Since it is challenging to model the topological information by Transformer, we utilize the random walk encoding $\rho_{i,j}^s = \psi^s(v_i, v_j)$ which maps the distance between a sampled random walk $D(v_i, v_j)$ to a scalar. Furthermore, the edges may also consist of rich contextual features (Dwivedi et al. 2020; Hu et al. 2021), e.g., edges represent the bonds between node pairs and have features to describe the type of the bonds in a molecular graph. Similar to random walk encoding, we choose the shortest path in the random walk instead of the fixed shortest path in the graph for each node pair. Given the shortest path $\mathcal{P}_{i,j}$ for v_i to v_j in the random walk, which represents the set of edges linked v_i to v_j in the random walk sequence $\mathcal{S}$, we defined the path features as follows.

$$\rho_{i,j}^p = \frac{1}{|\mathcal{P}_{i,j}|} \sum_{e \in \mathcal{P}_{i,j}} \mathbf{w}^{p\top} \mathbf{e}, \tag{5}$$

where $\mathbf{e}$ and $\mathbf{w}^p$ denote the features of the edge e and the trainable vector for weighted sum, respectively. The random walk encoding $\rho_{i,j}^s$ and path features $\rho_{i,j}^p$ serve as a bias term in the self attention mechanism, which can utilize both topological and contextual information from the given random walk path. Since there are residual connections between all modules, if the node does not exist in the random walk sequence, it will not be updated in this layer. In other words, the embeddings of those unsampled nodes are the same in the next layer. Therefore, to reduce the chance of missing some nodes, we sample the random walk with a large enough length l.

Random Walk Convolution Compared to Transformer, which models the global interactions, CNN exploits local features and is used as the de-facto computational block with

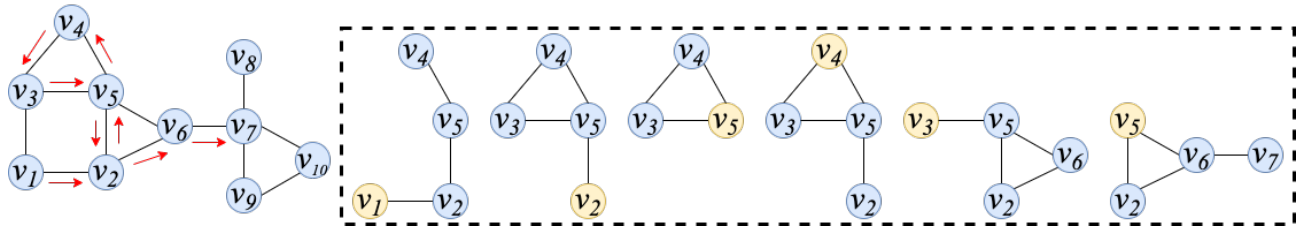

Figure 2: Assume the kernel size is 4, by sampling the random walk (red path) and processing the line graph with convolution operation, RWC can discover several motifs in the graph. We color the starting node in yellow for each motif.

a shared kernel. Inspired by Toenshoff et al. (Toenshoff et al. 2021), we adopt *Random Walk Convolution (RW-Conv)* to iteratively update node embeddings from the random walk sequence. In the k-th layer, RW-Conv first samples a sequence of nodes with the associated hidden features $\mathbf{S}^{k-1}$ from RW-SAN. Similar to RW-SAN, we also leverage the topological and contextual information from random walk encoding and edge features. For the random walk encoding $\mathbf{r}_n^{k-1} \in \mathbb{R}^w$, RW-Conv looks up the w predecessors of s_n, storing their distances to s_n, which is derived from Eq. (3), and mapping each distance to a scalar similar to RW-SAN. In addition to node features, we also utilize the edge features to explore the contextual information by RW-Conv. For each node s_n, we concatenate random walk encoding, and the edge features of predecessor $\mathbf{e}_{m,n}$ with s_m and the successor $\mathbf{e}_{n,o}$ with s_o to the original node features. The final input of the RW-Conv is defined as follows.

$$\hat{\mathbf{s}}_n^{(k-1)} = \mathbf{s}_n^{(k-1)} \,||\, \mathbf{r}_n^{k-1} \,||\, \mathbf{e}_{m,n} \,||\, \mathbf{e}_{n,o}, \tag{6}$$

where $||$ is the concatenation operation. In contrast, RW-SAN exploits the information to the attention map because the attention map represents the global pairwise relation while convolution is used to detect local structural information formed as motifs (see Figure 2).

Then, we process hidden features of the random walk sequence $\hat{\mathbf{S}}^{k-1}$ by 1D CNN with kernel size w.

$$\mathbf{s}_n^{(k)} = \sum_{i \in \lfloor -\frac{w}{2}, \frac{w}{2} \rfloor} \mathbf{w}_i^{(k)} \odot \hat{\mathbf{s}}_{n+i}^{(k-1)} + \mathbf{b}^{(k)}, \tag{7}$$

where $\odot$ denotes the Hadamard product between the input features $\hat{\mathbf{s}}^{k-1}$ and the kernel vector $\mathbf{w}_i$ with the bias $\mathbf{b}^{(k)}$. Therefore, RW-Conv gathers motif information by performing a convolution operation over the given random walk sequence, where the kernel in the CNN acts like a graph kernel and slides over the walk to extract features from each segment as a motif. Since a node v_i may occur multiple times in the random walk sequence, we take the average of the hidden features and reorder them to node features matrix $\mathbf{H}^{(k)}$ as the input of the next layer.

Readout To derive the whole graph embedding, one straightforward method is to sum the embedding of every node (Lee et al. 2019); however, direct summation treats each node equally, thus leading to the over-squashing problem (Alon and Yahav 2020). In contrast, we employ the virtual node, which is a special node that connects to all the nodes in the graph and can help nodes to exchange information effectively, and thus improve the performance (Gilmer et al. 2017; Li, Cai, and He 2017). Furthermore, since the virtual node aggregates information from every node in the graph, we adopt the hidden feature of the virtual node as the whole graph embedding and train an additional classifier for the downstream task. Since RWC depends on the random walk, one may concern about the robustness of RWC. Sampling different random walks can be viewed as data augmentation (You et al. 2020) that provides various motifs in each training iteration, which helps RWC become more robust. During our experiments, we notice that the learning curve can converge after several iterations.

Algorithm 1: Random Walk Conformer (RWC)

Adding degree encoding to node features.
for $k \leftarrow 1$ to K **do**
  Sample the random walk sequence $\mathcal{S}$ with random walk encodings $\boldsymbol{\rho}^s$
  $\mathbf{H}'^{(k-1)} = \mathbf{H}^{(k-1)} + \frac{1}{2}\text{FF}(\mathbf{H}^{(k-1)})$
  $\mathbf{H}''^{(k-1)} = \mathbf{H}'^{(k-1)} + \text{RW-SAN}(\mathbf{H}'^{(k-1)}, \boldsymbol{\rho}^s)$
  Construct the random walk features $\mathbf{S}^{(k-1)}$ from $\mathbf{H}''^{(k-1)}$ by the order of $\mathcal{S}$.
  $\mathbf{S}^{(k)} = \mathbf{S}^{(k-1)} + \text{RW-Conv}(\mathbf{S}^{(k-1)}, \boldsymbol{\rho}^s)$
  Rearrange the value in $\mathbf{S}^{(k-1)}$ to the corresponding node index in $\mathbf{H}''^{(k)}$.
  $\mathbf{H}'^{(k)} = \mathbf{H}''^{(k)} + \frac{1}{2}\text{FF}(\mathbf{H}''^{(k)})$
  $\mathbf{H}^{(k)} = \text{LayerNorm}(\mathbf{H}'^{(k)})$
Output the final embedding of the virtual node

Architecture

Fig. 1 illustrates the architecture of RWC. In the k-th layer, we first generate the random walk sequence $\mathcal{S}$ with associated random walk encodings. Then, RWC constructs the hidden features $\mathbf{S}^{(k-1)}$ by the lookup function on the previous layer $\mathbf{H}^{(k-1)}$ and processes the features by a conformer block. The conformer block sandwiches the proposed RW-SAN and RW-Conv modules by two Feed Forward networks (FFs). Furthermore, the residual connection is adopted within each module in a conformer block. Finally, we obtain the output features $\mathbf{H}^{(k)}$ of each node by rearranging the processed node embeddings from the random walk sequence w.r.t to the node index. If a node is sampled multiple times in the random walk sequence, we average the node features as the final output.

In addition, RWC can still obtain the node embedding that does not occur in the random walk sequence through residual connection, i.e., the embedding passes through two FFs.

The total complexity of RWC is $\mathcal{O}(|\mathcal{V}|^2 d + |\mathcal{V}|k(d+w))$, where k is kernel size, and w is window size for RW-Conv. Generally, $|\mathcal{V}| \gg k, d, w$. Hence, the complexity can be simplified to $\mathcal{O}(|\mathcal{V}|^2 d)$, the same as Transformer. In comparison, the complexity of K-WL is $\mathcal{O}(|\mathcal{V}|^{K+1} d^2 \log |\mathcal{V}|)$ (Immerman and Sengupta 2019), which is much larger than RWC. Detailed pseudocode is provided in Algorithm 1.

Experiments

Here, we evaluate RWC on graph classification with six datasets and graph regression with two datasets, compared to both graph kernel methods and state-of-the-art GNNs.

Experiment Setup

Datasets and Evaluation For graph classification, we test RWC on six TUDataset benchmarks (Morris et al. 2020) from various domains, including one biology (i.e., *PROTEINS*), three chemistry (i.e., *MUTAG*, *PTC*, and *NCI1*), and two social (i.e., *IMDB-B* and *IMDB-M*) datasets, measured by the accuracy score. For graph regression, we further conduct experiments on two public molecular benchmarks. *ZINC-500k* (Dwivedi et al. 2020) is a molecular dataset with $12K$ molecules that aims to predict the strained solubility. Besides, *PCQM4Mv2* (Hu et al. 2021) is currently the most extensive graph regression dataset with $3.7M$ molecules, which aims at predicting the HOMO-LUMO energy gap. The node label is the atomic number, and the edge labels specify the bond type for both datasets. The mean absolute error (MAE) measures the performance graph regression. We adopt the standard split for all benchmarks.

Implementation Details For all experiments, we use a linear learning rate scheduler with the peak learning rate $1e-3$ and the end learning rate $1e-9$. The optimizer of RWC is AdamW with the weight decay in 0.01. To avoid exploding gradient, we set a gradient clipping value to 5. Following Toenshoff et al. (Toenshoff et al. 2021), the random walk length for training is 50 for all experiments. Note that the lengths of random walks for validating and testing are longer than that for training to prevent missing some nodes while evaluating. The random walk length for testing is set to 150 for TUDatasets and ZINC-500k and 100 for PCQM4Mv2. Besides, the window size w and kernel size are set to 8 and 9, respectively. Due to the parameter constraint of Benchmarking GNN leaderboard (Dwivedi et al. 2020), we keep the total parameters of RWC approximate to $500K$ for ZINC-500k. Detailed hyper-parameter settings are reported in Appendix.

Graph Classification

Table 1 summarizes the quantitative results of the graph classification by comparing graph kernel methods and GNNs. For chemistry datasets, RWC outperforms all baselines by at least 2% on MUTAG, and 6.2% on PTC in terms of accuracy. Similarly, RWC improves the accuracy by 2.2% on the biological dataset PROTEINS, and the social networks by 1% on IMDB-B and 0.2% on IMDB-M. Specifically, RWC can outperform all graph kernel methods except K-WL method on NCI1 since there are 37 discrete labels in this dataset, and K-WL achieves better performance with a large number $K = 37$. However, the computation time of K-WL is unacceptable for evaluating all combinations of the k-order subgraphs. Note that motif-based GNNs outperform conventional GNNs that merely aggregate the information from neighbors. Specifically, CIN explores the graph's cellular structures, which is suitable for biology and chemistry graph data. Furthermore, GSN shows similar results in biology and chemistry datasets, achieving impressive performance on social networks. RWC employs the RW-SAN to measure the global pairwise correlations and RW-Conv to carefully analyze the local substructures of graphs.

Graph Regression

The results of the graph regression task are shown in Table 2. RWC achieves state-of-the-art performance by outperforming the best motif-based GNNs (CIN) by 14% and the best Transformer-based method (GRPE) by 28%. In ZINC-500k, we follow the budget constraints with $500K$ parameters for a fair comparison (Dwivedi et al. 2020). The results manifest that the RWC can achieve the best performance even with limited parameters by effectively discovering the motif by the random walk sequences with RW-SAN and RW-Conv. For PCQM4Mv2, since the test set is unavailable, we compare the result to the validation set with the best performance of 0.0837 in terms of MAE, which outperforms current state-of-the-art methods on the official leaderboard. Similar to the graph classification task, the motif-based GNNs, i.e., CIN and CRaWl, possess the motifs and thus beat those conventional GNNs on the chemistry datasets because the molecules consist of rich substructures, e.g., rings. On the other hand, Transformer-based methods, i.e., Graphormer, EGT, and GRPE, also outperform other GNNs since it models global relations and leverages spatial encoding to exploit the topological information on graphs, especially on a large dataset PCQM4Mv2. By leveraging the strengths of Transformer-based and motif-based GNNs, RWC can outperform all baselines significantly.

Ablation Studies

We conduct ablation studies to evaluate the importance of different modules in RWC on graph classification (NCI1) and graph regression (ZINC-500k) in Table 3. We first validate the importance of RW-SAN by comparing RW-SAN with the weighted and unweighted version, i.e., calculating the occurrence of each node f_j in the random walk in Eq. (4). The experimental results manifest that adopting RW-SAN can boost RWC's performance, and assigning weight, i.e., occurrence, to the attention mechanism according to the occurrence in the random walk can also improve the performance. Then, we investigate the effectiveness of RW-Conv, which exploits the motifs to strengthen RWC's performance significantly. Last, we compare the spatial encoding based on the shortest path (SP) and the random walk (RW). Random walk encoding is more powerful than shortest path encoding on ZINC-500k. We also theoretically prove that our random walk is more expressive than the shortest path in Theorem 1. While random

Method	MUTAG ↑	PTC ↑	PROTEINS ↑	NCI1 ↑	IMDB-B ↑	IMDB-M ↑
GK (k = 3) (Shervashidze et al. 2009)	81.4 ± 1.7	55.7 ± 0.5	71.4 ± 0.3	62.5 ± 0.3	-	-
PK (Neumann et al. 2016)	76.0 ± 2.7	59.5 ± 2.4	73.7 ± 0.7	82.5 ± 0.5	-	-
WL kernel (Shervashidze et al. 2011)	90.4 ± 5.7	59.9 ± 4.3	75.0 ± 3.1	$\mathbf{86.0 \pm 1.8}$	73.8 ± 3.9	50.9 ± 3.8
DGCNN (Zhang et al. 2018)	85.8 ± 1.8	58.6 ± 2.5	75.5 ± 0.9	74.4 ± 0.5	70.0 ± 0.9	47.8 ± 0.9
IGN (Maron et al. 2019b)	83.9 ± 13.0	58.5 ± 6.9	76.6 ± 5.5	74.3 ± 2.7	72.0 ± 5.5	48.7 ± 3.4
GIN (Xu et al. 2019)	89.4 ± 5.6	64.6 ± 7.0	76.2 ± 2.8	82.7 ± 1.7	75.1 ± 5.1	52.3 ± 2.8
PPGNs (Maron et al. 2019a)	90.6 ± 8.7	66.2 ± 6.6	77.2 ± 4.7	83.2 ± 1.1	73.0 ± 5.8	50.5 ± 3.6
Natural GN (de Haan, Cohen, and Welling 2020)	89.4 ± 1.6	66.8 ± 1.7	71.7 ± 1.0	82.4 ± 1.3	73.5 ± 2.0	51.3 ± 1.5
GSN (Bouritsas et al. 2020)	92.2 ± 7.5	68.2 ± 7.2	76.6 ± 5.0	83.5 ± 2.0	77.8 ± 3.3	54.3 ± 3.3
RWNN (Nikolentzos and Vazirgiannis 2020)	89.2 ± 4.3	65.8 ± 5.5	74.7 ± 3.3	73.9 ± 1.3	70.8 ± 4.8	48.8 ± 2.9
CRaWl (Toenshoff et al. 2021)	88.2 ± 5.6	63.9 ± 4.9	74.1 ± 4.4	82.0 ± 2.0	72.7 ± 2.8	47.8 ± 3.9
CIN (Bodnar et al. 2021)	92.7 ± 6.1	68.2 ± 5.6	77.0 ± 4.3	83.6 ± 1.4	75.6 ± 3.7	52.7 ± 3.1
GNN-AK (Zhao et al. 2022)	91.3 ± 7.0	67.7 ± 8.8	77.1 ± 5.7	85.0 ± 2.0	75.0 ± 4.2	52.7 ± 4.8
RWC	$\mathbf{94.7 \pm 3.7}$	$\mathbf{74.4 \pm 4.4}$	$\mathbf{79.4 \pm 4.7}$	82.2 ± 1.5	$\mathbf{78.8 \pm 3.1}$	$\mathbf{54.5 \pm 2.9}$

Table 1: Results on graph classification.

Method	ZINC ↓	PCQM4Mv2 ↓
GCN (Kipf and Welling 2017)	0.3671	0.1153
GraphSAGE (Hamilton, Ying, and Leskovec 2017)	0.3982	0.1199
GIN (Xu et al. 2019)	0.5264	0.1083
PNA (Corso et al. 2020)	0.1421	0.1231
GT (Dwivedi and Bresson 2021)	0.2265	0.1002
CRaWl (Toenshoff et al. 2021)	0.0851	0.0902
Graphormer (Ying et al. 2021)	0.1228	0.0865
EGT (Hussain, Zaki, and Subramanian 2021)	0.1085	0.0869
GRPE (Park et al. 2022)	0.0942	0.0890
CIN (Bodnar et al. 2021)	0.0790	0.0871
GNN-AK (Zhao et al. 2022)	0.0801	0.0881
RWC	**0.0678**	**0.0837**

Table 2: Results on graph regression.

RW-SAN	RW-Conv	Enc.	ZINC MAE ↓	NCI1 Acc. (%) ↑
w/o weighted	✓	SP	0.082	82.4
w/o weighted	✓	RW	0.070	82.6
w/o weighted	-	SP	0.141	79.1
w/o weighted	-	RW	0.120	76.2
-	✓	SP	0.090	80.6
-	✓	RW	0.077	81.2
w/ weighted	✓	SP	0.081	82.6
w/ weighted	✓	RW	0.068	82.2
w/ weighted	-	SP	0.130	75.7
w/ weighted	-	RW	0.116	75.8

Table 3: Ablation studies.

	training walk length			
Sampling method	20	50	100	200
Non-backtracking	0.191	0.068	0.065	0.077
Backtracking	0.299	0.131	0.091	0.085

Table 4: Sensitivity test on ZINC dataset. (MAE ↓)

walk encoding show relatively weak performance in some case of NCI1, our potential explanation is that random walk encoding requires a larger number of layers, and there are 16 layers for ZINC-500k and 4 layers for NCI1.

Sensitivity Test

As a key component of RWC, we evaluate the impact of various parameter of random walk.

Effect of Random Walk Sampling Method We follow the random sampling method from CRaWl (Toenshoff et al. 2021) and Node2Vec (Grover and Leskovec 2016), which conducts random walks without backtracking to exploit the high-order neighborhood more effectively. We also compare the performance of non-backtracking and backtracking in Table 4. Non-backtracking can significantly outperform backtracing since it helps RWC to discover further nodes.

Effect of Training Random Walk Length We also conduct experiments for different training walk lengths on ZINC dataset (with a test length 150) in Table 4. While RWC can exploit more diverse patterns in the graph with a longer walk sequence, there is a tradeoff between performance and efficiency. Furthermore, since random walk encoding is more expressive than shortest path spatial encoding, and if the length becomes a large number, random walk encoding acts like shortest path spatial encoding, thus the walk length indeed affects the model performance.

Conclusion

We have proposed a novel GNN, namely Random Walk Conformer (RWC), which benefits from its attention mechanism to extract global pairwise relations and random walk convolution operation to model the local structure from motifs. Furthermore, we propose novel random walk encoding to exploit motifs on graphs, which is theoretically proven to be more expressive than conventional spatial encoding. Experiment results manifest that RWC achieves state-of-the-art performance in various domains, including biological, chemistry, and social datasets for graph classification and regression.

Acknowledgments

This work is supported by MOST Project No. 111-2221-E-002-135-MY3 and No. 111-2223-E-002-006, Taiwan.

References

Abdullah, M. 2012. The Cover Time of Random Walks on Graphs. *CoRR*, abs/1202.5569.

Alon, U.; and Yahav, E. 2020. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*.

Baek, J.; Kang, M.; and Hwang, S. J. 2021. Accurate Learning of Graph Representations with Graph Multiset Pooling. In *International Conference on Learning Representations*.

Bodnar, C.; Frasca, F.; Otter, N.; Wang, Y. G.; Liò, P.; Montúfar, G.; and Bronstein, M. 2021. Weisfeiler and Lehman Go Cellular: CW Networks. In *Advances in Neural Information Processing Systems*, volume 34.

Bouritsas, G.; Frasca, F.; Zafeiriou, S.; and Bronstein, M. M. 2020. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*.

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J. 2020. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33: 10383–10395.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal Neighbourhood Aggregation for Graph Nets. In *Advances in Neural Information Processing Systems*.

de Haan, P.; Cohen, T. S.; and Welling, M. 2020. Natural graph networks. *Advances in Neural Information Processing Systems*, 33: 3636–3646.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Dwivedi, V. P.; and Bresson, X. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

Dwivedi, V. P.; Joshi, C. K.; Laurent, T.; Bengio, Y.; and Bresson, X. 2020. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y. W., eds., *ICML*, volume 70 of *Proceedings of Machine Learning Research*, 1263–1272. PMLR.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*.

Gulati, A.; Qin, J.; Chiu, C.-C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; and Pang, R. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. In Meng, H.; Xu, B.; and Zheng, T. F., eds., *INTERSPEECH*, 5036–5040. ISCA.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *NIPS*, 1024–1034.

Han, G.; and Sethu, H. 2016. Waddling Random Walk: Fast and Accurate Mining of Motif Statistics in Large Graphs. In Bonchi, F.; Domingo-Ferrer, J.; Baeza-Yates, R.; Zhou, Z.-H.; and Wu, X., eds., *ICDM*, 181–190. IEEE Computer Society.

Hu, W.; Fey, M.; Ren, H.; Nakata, M.; Dong, Y.; and Leskovec, J. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *arXiv preprint arXiv:2103.09430*.

Hussain, M. S.; Zaki, M. J.; and Subramanian, D. 2021. Edge-augmented Graph Transformers: Global Self-attention is Enough for Graphs. *arXiv preprint arXiv:2108.03348*.

Immerman, N.; and Sengupta, R. 2019. The k-Dimensional Weisfeiler-Leman Algorithm. *CoRR*, abs/1907.09582.

Itzhack, R.; Mogilevski, Y.; and Louzoun, Y. 2007. An optimal algorithm for counting network motifs. *Physica A: Statistical Mechanics and its Applications*, 381: 482–490.

Ke, G.; He, D.; and Liu, T.-Y. 2021. Rethinking Positional Encoding in Language Pre-training. In *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34.

Lee, J. B.; Rossi, R. A.; Kong, X.; Kim, S.; Koh, E.; and Rao, A. 2019. Graph convolutional networks with motif-based attention. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 499–508.

Li, J.; Cai, D.; and He, X. 2017. Learning Graph-Level Representation for Drug Discovery. *CoRR*, abs/1709.03741.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019a. Provably Powerful Graph Networks. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E. B.; and Garnett, R., eds., *NeurIPS*, 2153–2164.

Maron, H.; Ben-Hamu, H.; Shamir, N.; and Lipman, Y. 2019b. Invariant and Equivariant Graph Networks. In *International Conference on Learning Representations*.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4602–4609.

Neumann, M.; Garnett, R.; Bauckhage, C.; and Kersting, K. 2016. Propagation kernels: efficient graph kernels from propagated information. *Mach. Learn.*, 102(2): 209–245.

Nikolentzos, G.; and Vazirgiannis, M. 2020. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33: 16211–16222.

Park, W.; Chang, W.-G.; Lee, D.; Kim, J.; and seung-won hwang. 2022. GRPE: Relative Positional Encoding for Graph Transformer. In *ICLR2022 Machine Learning for Drug Discovery*.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33: 12559–12571.

Sankar, A.; Zhang, X.; and Chang, K. C.-C. 2017. Motif-based Convolutional Neural Network on Graphs. *arXiv preprint arXiv:1711.05697*.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In Walker, M. A.; Ji, H.; and Stent, A., eds., *NAACL-HLT (2)*, 464–468. Association for Computational Linguistics. ISBN 978-1-948087-29-2.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.*, 12: 2539–2561.

Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. M. 2009. Efficient graphlet kernels for large graph comparison. In Dyk, D. A. V.; and Welling, M., eds., *AISTATS*, volume 5 of *JMLR Proceedings*, 488–495. JMLR.org.

Toenshoff, J.; Ritzert, M.; Wolf, H.; and Grohe, M. 2021. Graph Learning with 1D Convolutions on Random Walks. *arXiv preprint arXiv:2102.08786*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proceedings of the 7th International Conference on Learning Representations*, ICLR '19, 1–17.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do Transformers Really Perform Badly for Graph Representation? In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W. L.; and Leskovec, J. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, 4805–4815.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In McIlraith, S. A.; and Weinberger, K. Q., eds., *AAAI*, 4438–4445. AAAI Press.

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

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N. 2022. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *International Conference on Learning Representations*.

Zhou, D.; Zhang, S.; Yildirim, M. Y.; Alcorn, S.; Tong, H.; Davulcu, H.; and He, J. 2021. High-Order Structure Exploration on Massive Graphs: A Local Graph Clustering Perspective. *ACM Trans. Knowl. Discov. Data*, 15(2).