Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching

Xin Liu, Yangqiu Song

Department of CSE, the Hong Kong University of Science and Technology
{xliucr, yqsong}@cse.ust.hk

Abstract

Graph neural networks (GNNs) and message passing neural networks (MPNNs) have been proven to be expressive for subgraph structures in many applications. Some applications in heterogeneous graphs require explicit edge modeling, such as subgraph isomorphism counting and matching. However, existing message passing mechanisms are not designed well in theory. In this paper, we start from a particular edge-to-vertex transform and exploit the isomorphism property in the edge-to-vertex dual graphs. We prove that searching isomorphisms on the original graph is equivalent to searching on its dual graph. Based on this observation, we propose dual message passing neural networks (DMPNNs) to enhance the substructure representation learning in an asynchronous way for subgraph isomorphism counting and matching as well as unsupervised node classification. Extensive experiments demonstrate the robust performance of DMPNNs by combining both node and edge representation learning in synthetic and real heterogeneous graphs. Code is available at https://github.com/HKUST-KnowComp/DualMessagePassing.

Introduction

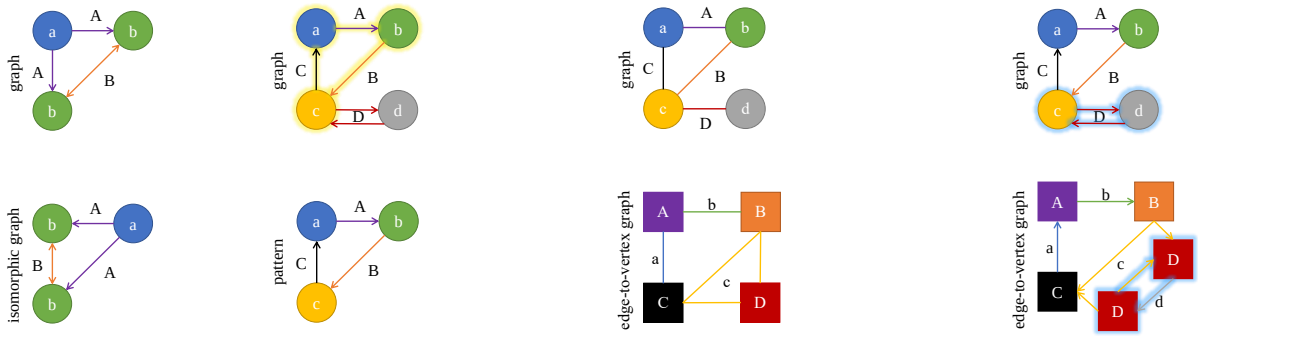
Graphs have been widely used in various applications across domains from chemoinformatics to social networks. The isomorphism is one of the important properties in graphs, and analysis on subgraph isomorphisms is useful in real applications. For example, we can determine the properties of compounds by finding functional group information in chemical molecules (Gilmer et al. 2017); some substructures in social networks are regarded as irreplaceable features in recommender systems (Ying et al. 2018). The challenge of finding subgraph isomorphisms requires the exponential computational cost. Particularly, finding and counting require global inference to oversee the whole graph. Existing counting and matching algorithms are designed for some query patterns up to a certain size (e.g., 5), and some of them cannot directly apply to heterogeneous graphs where vertices and edges are labeled with types (Bhattarai, Liu, and Huang 2019; Sun and Luo 2020).

There has been more attention to using deep learning to count or match subgraph isomorphisms. Liu et al. (2020) designed a general end-to-end framework to predict the number of subgraph isomorphisms on heterogeneous graphs, and Ying et al. (2020) combined node embeddings and voting to match subgraphs. They found that neural networks could speed up 10 to 1,000 times compared with traditional searching algorithms. Xu et al. (2019) and Morris et al. (2019) showed that graph neural networks (GNNs) based on message passing are at most as powerful as the WL test (Weisfeiler and Leman 1968), and Chen et al. (2020) further analyzed the upper-bound of message passing and k-WL for subgraph isomorphism counting. These studies show that it is theoretically possible for neural methods to count larger patterns in complex graphs. In heterogeneous graphs, edges play an important role in checking and searching isomorphisms because graph isomorphisms require taking account of graph adjacency and edge types. However, existing message passing mechanisms have not paid enough attention to edge representations (Gilmer et al. 2017; Schlichtkrull et al. 2018; Vashishth et al. 2020; Jin et al. 2021).

In this paper, we discuss a particular edge-to-vertex transform and find the one-to-one correspondence between subgraph isomorphisms of original graphs and subgraph isomorphisms of their corresponding edge-to-vertex dual graphs. This property suggests that searching isomorphisms on the original graph is equivalent to searching on its dual graph. Based on this observation and the theoretical guarantee, we propose new dual message passing networks (DMPNNs) to learn node and edge representations simultaneously in the aligned space. Empirical results show the effectiveness of DMPNNs on all homogenerous and heterogeneous graphs, synthetic data or real-life data.

Our main contributions are summarized as follows:

1. We prove that there is a one-to-one correspondence between isomorphisms of connected directed heterogeneous multi-graphs with reversed edges and isomorphisms between their edge-to-vertex dual graphs.
2. We propose dual message passing mechanism and design the DMPNN model to explicitly model edges and align node and edge representations in the same space.
3. We empirically demonstrate that DMPNNs can count subgraph isomorphisms more accurately and match isomorphic nodes more correctly. DMPNNs also surpass competitive baselines on unsupervised node classification, indicating the necessity of explicit edge modeling for general graph representation learning.



(a) Isomorphism (b) Subgraph isomorphism (c) Edge-to-vertex transform (undirected) (d) Edge-to-vertex transform (directed)

Figure 1: Examples of the isomorphism, subgraph isomorphism, and edge-to-vertex transforms.

Preliminaries

To be more general, we assume a graph is a directed heterogeneous multigraph. Let $\mathcal{G}$ be a graph with a vertex set $\mathcal{V}_\mathcal{G}$ and each vertex with a different *vertex id*, an edge set $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$, a label function $\mathcal{X}_\mathcal{G}$ that maps a vertex to a *vertex label*, and a label function $\mathcal{Y}_\mathcal{G}$ that maps an edge to a set of *edge labels*. As we regard each edge can be associated with a set of labels, we can merge multiple edges with the same source and the same target as one edge with multiple labels. A subgraph of $\mathcal{G}$, denoted as $\mathcal{G}_S$, is any graph with $\mathcal{V}_{\mathcal{G}_S} \subseteq \mathcal{V}_\mathcal{G}$, $\mathcal{E}_{\mathcal{G}_S} \subseteq \mathcal{E}_\mathcal{G} \cap (\mathcal{V}_{\mathcal{G}_S} \times \mathcal{V}_{\mathcal{G}_S})$ satisfying $\forall v \in \mathcal{V}_{\mathcal{G}_S}, \mathcal{X}_{\mathcal{G}_S}(v) = \mathcal{X}_\mathcal{G}(v)$ and $\forall e \in \mathcal{E}_{\mathcal{G}_S}, \mathcal{Y}_{\mathcal{G}_S}(e) = \mathcal{Y}_\mathcal{G}(e)$. To simplify the statement, we let $\mathcal{Y}_\mathcal{G}((u,v)) = \phi$ if $(u,v) \notin \mathcal{E}_\mathcal{G}$.

Isomorphisms and Subgraph Isomorphisms

Definition 1 (Isomorphism)**.** A graph $\mathcal{G}_1$ is *isomorphic* to a graph $\mathcal{G}_2$ if there is a bijection $f : \mathcal{V}_{\mathcal{G}_1} \to \mathcal{V}_{\mathcal{G}_2}$ such that:

- $\forall v \in \mathcal{V}_{\mathcal{G}_1}, \mathcal{X}_{\mathcal{G}_1}(v) = \mathcal{X}_{\mathcal{G}_2}(f(v))$,
- $\forall v' \in \mathcal{V}_{\mathcal{G}_2}, \mathcal{X}_{\mathcal{G}_2}(v') = \mathcal{X}_{\mathcal{G}_1}(f^{-1}(v'))$,
- $\forall (u,v) \in \mathcal{E}_{\mathcal{G}_1}, \mathcal{Y}_{\mathcal{G}_1}((u,v)) = \mathcal{Y}_{\mathcal{G}_2}((f(u), f(v)))$,
- $\forall (u',v') \in \mathcal{E}_{\mathcal{G}_2}, \mathcal{Y}_{\mathcal{G}_2}((u',v')) = \mathcal{Y}_{\mathcal{G}_1}((f^{-1}(u'), f^{-1}(v')))$.

We write $\mathcal{G}_1 \simeq \mathcal{G}_2$ for such isomorphic property and name f as an *isomorphism*. For example, there are two different isomorphisms between the two triangles in Figure 1a. As a special case, the isomorphism f between two empty graphs without any vertex is $\{\} \to \{\}$.

In addition, if a subgraph of $\mathcal{G}_1$ is isomorphic to another graph, then the corresponding bijection function is named as a *subgraph isomorphism*. The formal definition is:

Definition 2 (Subgraph isomorphism)**.** If a subgraph $\mathcal{G}_{1_S}$ of $\mathcal{G}_1$ is isomorphic to a graph $\mathcal{G}_2$ with a bijection f, we say $\mathcal{G}_1$ contains a subgraph isomorphic to $\mathcal{G}_2$ and name f as a *subgraph isomorphism*.

Subgraph isomorphism related problems commonly refer to two kinds of subgraphs: node-induced subgraphs and edge-induced subgraphs. In node-induced subgraph related problems, the possible subgraphs require that for each vertex in $\mathcal{G}_S$, the associated edges in $\mathcal{G}$ must appear in $\mathcal{G}_S$, i.e., $\mathcal{V}_{\mathcal{G}_S} \subseteq \mathcal{V}_\mathcal{G}, \mathcal{E}_{\mathcal{G}_S} = \mathcal{E}_\mathcal{G} \cap (\mathcal{V}_{\mathcal{G}_S} \times \mathcal{V}_{\mathcal{G}_S})$; in edge-induced subgraph related problems, the required subgraphs are restricted by associating vertices that are incident to edges, i.e., $\mathcal{E}_{\mathcal{G}_S} \subseteq \mathcal{E}_\mathcal{G}$, $\mathcal{V}_{\mathcal{G}_S} = \{u|(u,v) \in \mathcal{E}_{\mathcal{G}_S}\} \cup \{v|(u,v) \in \mathcal{E}_{\mathcal{G}_S}\}$. Node-induced subgraphs are specific edge-induced subgraphs when $\mathcal{G}$ is connected. Hence, we assume all subgraphs mentioned in the following are edge-induced for better generalization. Figure 1b shows an example of subgraph isomorphism that a graph with four vertices is subgraph isomorphic to the triangle pattern.

Edge-to-vertex Transforms

In graph theory, the line graph of an undirected graph $\mathcal{G}$ is another undirected graph that represents the adjacencies between edges of $\mathcal{G}$, e.g., Figure 1c. We extend line graphs to directed heterogeneous multigraphs.

Definition 3 (Edge-to-vertex transform)**.** A *line graph* (also known as *edge-to-vertex dual graph*) $\mathcal{H}$ of a graph $\mathcal{G}$ is obtained by associating a vertex $v' \in \mathcal{V}_\mathcal{H}$ with each edge $e = g^{-1}(v') \in \mathcal{E}_\mathcal{G}$ and connecting two vertices $u', v' \in \mathcal{V}_\mathcal{H}$ with an edge from u' to v' if and only if the destination of the corresponding edge $d = g^{-1}(u')$ is exact the source of $e = g^{-1}(v')$. Formally, we have:

- $\forall e = (u,v) \in \mathcal{E}_\mathcal{G}, \mathcal{Y}_\mathcal{G}(e) = \mathcal{X}_\mathcal{H}(g(e))$,
- $\forall v' \in \mathcal{V}_\mathcal{H}, \mathcal{X}_\mathcal{H}(v') = \mathcal{Y}_\mathcal{G}(g^{-1}(v'))$,
- $\forall d, e \in \mathcal{E}_\mathcal{G}, u' = g(d) \in \mathcal{V}_\mathcal{H}, v' = g(e) \in \mathcal{V}_\mathcal{H}$, $(d.\text{target} = e.\text{source} = v) \to (\mathcal{Y}_\mathcal{H}((u',v')) = \mathcal{X}_\mathcal{G}(v)))$,
- $\forall e' = (u',v') \in \mathcal{E}_\mathcal{H}, d = g^{-1}(u') \in \mathcal{V}_\mathcal{G}, e = g^{-1}(v') \in \mathcal{V}_\mathcal{G}$, $(d.\text{target} = e.\text{source}) \wedge (\mathcal{Y}_\mathcal{H}(e') = \mathcal{X}_\mathcal{H}(d.\text{target}))$.

We call the bijection $g : \mathcal{E}_\mathcal{G} \to \mathcal{V}_\mathcal{H}$ as the *edge-to-vertex map*, and write $\mathcal{H}$ as $L(\mathcal{G})$ where $L : \mathcal{G} \to \mathcal{H}$ corresponds to the *edge-to-vertex transform*. There are several differences between undirected line graphs and directed line graphs. As shown in Figure 1c and Figure 1d, except directions of edges, an edge with its inverse in the original graph will introduce two corresponding vertices and a pair of reversed edges in between in the line graph.

There are many properties in the edge-to-vertex graph. As the vertices of the line graph $\mathcal{H}$ corresponds to the edges of the original graph $\mathcal{G}$, some properties of $\mathcal{G}$ that depend only on adjacency between edges may be preserved as equivalent properties in $\mathcal{H}$ that depend on adjacency between vertices. For example, an independent set in $\mathcal{H}$ corresponds to a matching (also known as independent edge set) in $\mathcal{G}$. But the edge-to-vertex transform may lose the information of the original graph. For example, two different graphs may have the same line graph. We have one observation that if two

graphs are isomorphic, their line graphs are also isomorphic; nevertheless, the converse is not always correct. We will discuss the isomorphism and the edge-to-vertex transform in the next section.

Isomorphisms vs. Edge-to-vertex Transforms

The edge-to-vertex transform can preserve adjacency relevant properties of graphs. In this section, we discuss isomorphisms and the edge-to-vertex transform. Particularly, we analyze the symmetry of isomorphisms in special situations transforming edges to vertices, and we further extend all graphs into this particular kind of structure for searching.

Proposition 4. *If two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic with an isomorphism $f : \mathcal{V}_{\mathcal{G}_1} \to \mathcal{V}_{\mathcal{G}_2}$, then their line graphs $\mathcal{H}_1$ and $\mathcal{H}_2$ are also isomorphic with an isomorphism $f' : \mathcal{V}_{\mathcal{H}_1} \to \mathcal{V}_{\mathcal{H}_2}$ such that $\forall v \in \mathcal{V}_{\mathcal{H}_1}, \mathcal{X}_{\mathcal{H}_1}(v) = \mathcal{X}_{\mathcal{H}_2}(f'(v))$ and $\forall v' \in \mathcal{V}_{\mathcal{H}_2}, \mathcal{X}_{\mathcal{H}_2}(v') = \mathcal{X}_{\mathcal{H}_1}(f'^{-1}(v'))$.*

The proof is shown in Appendix A. Furthermore, we conclude that the dual isomorphism f' satisfies $\forall v \in \mathcal{V}_{\mathcal{H}_1}$, $f'(v) = g_2((f(g_1^{-1}(v).\text{source}), f(g_1^{-1}(v).\text{target})))$. We denote $\mathcal{G}_1 \simeq \mathcal{G}_2 \to L(\mathcal{G}_1) \simeq L(\mathcal{G}_2)$ for Proposition 4.

The relation between the isomorphism f and its dual f' is non-injective: two line graphs in Figure 2a are isomorphic but their original graphs are not, which also indicates f' may correspond to multiple different f (even f does not exist). That is to say, the edge-to-vertex transform L cannot remain all graph adjacency and guarantee isomorphisms in some situations.

Theorem 5 (Whitney isomorphism theorem)**.** *For connected simple graphs with more than four vertices, there is a one-to-one correspondence between isomorphisms of the graphs and isomorphisms of their line graphs.*

Theorem 5 (Whitney 1932) concludes the condition for simple graphs. Inspired by it, we add reversed edges associated with special labels for directed graphs so that graphs can be regarded as undirected (Figure 2b). Theorem 6 is the extension for directed heterogeneous multigraphs.

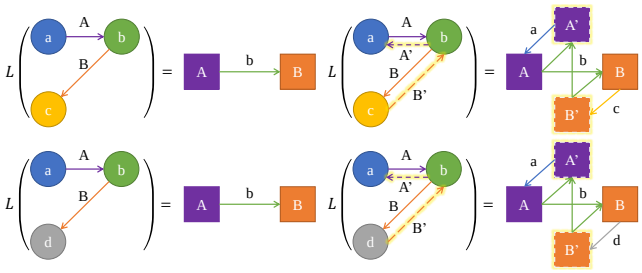
Theorem 6. *For connected directed heterogeneous multigraphs with reversed edges (the reverse of one self-loop is itself), there is a one-to-one correspondence between isomorphisms of the graphs and isomorphisms of their line graphs.*

The detailed proof is listed in Appendix B. Moreover, we have Corollary 7 for subgraph isomorphisms and their duals.

Corollary 7. *For connect directed heterogeneous multigraphs with reversed edges more than one vertex, there is a one-to-one correspondence between subgraph isomorphisms of the graphs and subgraph isomorphisms of their line graphs.*

Dual Message Passing Neural Networks

The edge-to-vertex transform and the duality property indicate that searching isomorphisms on the original graph is equivalent to searching on its line graph. Hence, we design the dual message passing to model nodes with original structure and model edges with the line graph structure. Moreover, we extend the dual message passing to heterogeneous multi-graphs.



(a) Non-injective case (b) Adding reversed edges

Figure 2: Non-isomorphic graphs and their line graphs.

Conventional Graph Convolutions

Kipf and Welling (2017) proposed parameterized conventional graph convolutions as the first-order approximation of spectral convolutions $\boldsymbol{\Theta} \star \boldsymbol{h} = \boldsymbol{U}\boldsymbol{\Theta}\boldsymbol{U}^\top \boldsymbol{h}$, where $\boldsymbol{\Theta}$ is the filter in the Fourier domain and $\boldsymbol{h} \in \mathbb{R}^n$ is the scalar feature vector for n vertices of $\mathcal{G}$. In practice, $\boldsymbol{\Theta}$ is a diagonal matrix as a function of eigenvalues of the (normalized) graph Laplacian. Considering the computational cost of eigendecomposition is $\mathcal{O}(n^3)$, it is approximated by shifted Chebyshev polynomials (Hammond, Vandergheynst, and Gribonval 2011):

$$\begin{aligned}\boldsymbol{\Theta} &\approx \sum_{k=0}^{K} T_k(\frac{2}{\lambda_{\mathcal{G}\max}}\boldsymbol{\Lambda}_{\mathcal{G}} - \boldsymbol{I}_n)\boldsymbol{\theta}_k \\ &\approx T_0(\frac{2}{\lambda_{\mathcal{G}\max}}\boldsymbol{\Lambda}_{\mathcal{G}} - \boldsymbol{I}_n)\boldsymbol{\theta}_0 + T_1(\frac{2}{\lambda_{\mathcal{G}\max}}\boldsymbol{\Lambda}_{\mathcal{G}} - \boldsymbol{I}_n)\boldsymbol{\theta}_1 \\ &= \boldsymbol{\theta}_0 + (\frac{2}{\lambda_{\mathcal{G}\max}}\boldsymbol{\Lambda}_{\mathcal{G}} - \boldsymbol{I}_n)\boldsymbol{\theta}_1, \end{aligned} \tag{1}$$

where $\boldsymbol{\Lambda}_{\mathcal{G}}$ is the diagonal matrix of eigenvalues, $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ is an identity matrix, $\lambda_{\mathcal{G}\max}$ is the largest eigenvalue so that the input of $T_k(\cdot)$ is located in $[-1, 1]$. Therefore, the convolution becomes to

$$\begin{aligned}\boldsymbol{\Theta} \star \boldsymbol{h} &= \boldsymbol{U}\boldsymbol{\Theta}\boldsymbol{U}^T\boldsymbol{h} \\ &\approx \boldsymbol{U}(\boldsymbol{\theta}_0 + (\frac{2}{\lambda_{\mathcal{G}\max}}\boldsymbol{\Lambda}_{\mathcal{G}} - \boldsymbol{I}_{n\times n})\boldsymbol{\theta}_1)\boldsymbol{U}^T\boldsymbol{h} \\ &= (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_1)\boldsymbol{h} + \frac{2\boldsymbol{\theta}_1}{\lambda_{\mathcal{G}\max}}\boldsymbol{L}_{\mathcal{G}}\boldsymbol{h}, \end{aligned} \tag{2}$$

where $\boldsymbol{L}_{\mathcal{G}}$ is the (normalized) graph Laplacian matrix. $\lambda_{\mathcal{G}\max}$ is bounded by $\max\{d_u + d_v | (u, v) \in \mathcal{E}_{\mathcal{G}}\}$ if the Laplacian $\boldsymbol{L}_{\mathcal{G}} = \boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}}$ or by 2 if the Laplacian is noramlzied as $\boldsymbol{D}_{\mathcal{G}}^{-\frac{1}{2}}(\boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}})\boldsymbol{D}_{\mathcal{G}}^{-\frac{1}{2}} = \boldsymbol{I}_n - \boldsymbol{D}_{\mathcal{G}}^{-\frac{1}{2}}\boldsymbol{A}_{\mathcal{G}}\boldsymbol{D}_{\mathcal{G}}^{-\frac{1}{2}}$, where $\boldsymbol{A}_{\mathcal{G}}$ is the adjacency matrix and a_{uv} corresponds to the number of edges from vertex u to vertex v, $\boldsymbol{D}_{\mathcal{G}}$ is the degree diagonal matrix and d_v is the (out-)degree of vertex v (Zhang 2011). Graph convolution networks have shown great success in many fields, including node classification, graph property prediction, graph isomorphism test, and subgraph isomorphism counting. Xu et al. (2019) and Liu et al. (2020) found that the sum aggregation is good at capturing structural information and solving isomorphism problems. Hence, we consider to use the unnormalized graph Laplacian $\boldsymbol{L}_{\mathcal{G}} = \boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}}$ and set $\lambda_{\mathcal{G}\max} = \max\{d_u + d_v | (u, v) \in \mathcal{E}_{\mathcal{G}}\}$.

Dual Message Passing Mechanism

This convolution can also apply on the line graph $\mathcal{H} = L(\mathcal{G})$, then convolutional operation in $\mathcal{H}$ is

$$\mathbf{\Gamma} \star \boldsymbol{z} \approx (\gamma_0 - \gamma_1)\boldsymbol{z} + \frac{2\gamma_1}{\lambda_{\mathcal{H}\text{max}}}\boldsymbol{L}_{\mathcal{H}}\boldsymbol{z}, \tag{3}$$

where $\mathbf{\Gamma}$ is the filter for $\mathcal{H}$, $\boldsymbol{z} \in \mathbb{R}^m$ is the scalar feature vector for m vertices of $\mathcal{H}$, and $\lambda_{\mathcal{H}\text{max}}$ is the largest eigenvalue of the Laplacian $L_{\mathcal{H}}$, which is no greater than $\max\{d_u + d_v | (u,v) \in \mathcal{E}_H\}$. We can use Eq. (3) to acquire the edge representations of $\mathcal{G}$ because Definition 3 and Corollary 7 show the line graph $\mathcal{H}$ also preserves the structural information of $\mathcal{G}$ for subgraph isomorphisms.

However, Eq. (3) results in a new problem: the computation cost is linear to $|\mathcal{E}_{\mathcal{H}}| = \frac{1}{2}\sum_{v\in\mathcal{G}} d_v^2 - n = \mathcal{O}(m^2)$ where $m = |\mathcal{E}_{\mathcal{G}}| = |\mathcal{V}_{\mathcal{H}}|$. To tackle this issue, we combine the two convolutions in an asynchronous manner in $\mathcal{O}(m)$.

Proposition 8. *If $\mathcal{G}$ is a directed graph with n vertices and m edges, then $\boldsymbol{A}_{\mathcal{G}} + \boldsymbol{A}_{\mathcal{G}}^\top = \boldsymbol{D}_{\mathcal{G}}^+ + \boldsymbol{D}_{\mathcal{G}}^- - \boldsymbol{B}_{\mathcal{G}}\boldsymbol{B}_{\mathcal{G}}^\top$, where $\boldsymbol{A}_{\mathcal{G}} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $\boldsymbol{D}_{\mathcal{G}}^+, \boldsymbol{D}_{\mathcal{G}}^- \in \mathbb{R}^{n\times n}$ are the out-degree and in-degree diagonal matrices respectively, and $\boldsymbol{B}_{\mathcal{G}} \in \mathbb{R}^{n\times m}$ is the oriented incidence matrix where $b_{ve} = 1$ if vertex v is the destination of edge e, $b_{ve} = -1$ if v is the source of e, $b_{ve} = 0$ otherwise. In particular, if $\mathcal{G}$ is with reversed edges, then we have $\boldsymbol{B}_{\mathcal{G}}\boldsymbol{B}_{\mathcal{G}}^\top = 2(\boldsymbol{D}_{\mathcal{G}} - \boldsymbol{A}_{\mathcal{G}}) = 2\boldsymbol{L}_{\mathcal{G}}$, where $\boldsymbol{L}_{\mathcal{G}} \in \mathbb{R}^{n\times n}$ is the Laplacian matrix.*

Proposition 9. *If $\mathcal{G}$ is a directed graph with n vertices and m edges and $\mathcal{H}$ is the line graph of $\mathcal{G}$, then $\boldsymbol{A}_{\mathcal{H}} + \boldsymbol{A}_{\mathcal{H}}^\top = \hat{\boldsymbol{B}}_{\mathcal{G}}^\top\hat{\boldsymbol{B}}_{\mathcal{G}} - 2\boldsymbol{I}_m$, where $\boldsymbol{A}_{\mathcal{H}} \in \mathbb{R}^{m\times m}$ is the adjacency matrix of $\mathcal{H}$, $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ is an identity matrix, and $\hat{\boldsymbol{B}}_{\mathcal{G}} \in \mathbb{R}^{n\times m}$ is the unoriented incidence matrix where $\hat{b}_{ve} = 1$ if vertex v is incident to edge e, $\hat{b}_{ve} = 0$ otherwise. In particular, if $\mathcal{G}$ is with reversed edges, then $\mathcal{H}$ is also with reversed edges and $\boldsymbol{A}_{\mathcal{H}} = \frac{1}{2}\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\hat{\boldsymbol{B}}_{\mathcal{G}} - \boldsymbol{I}_m$. Furthermore, we have $\boldsymbol{L}_{\mathcal{H}} = \boldsymbol{D}_{\mathcal{H}} - \boldsymbol{A}_{\mathcal{H}} = \boldsymbol{D}_{\mathcal{H}} + \boldsymbol{I}_m - \frac{1}{2}\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\hat{\boldsymbol{B}}_{\mathcal{G}}$, where $\boldsymbol{L}_{\mathcal{H}} \in \mathbb{R}^{m\times m}$ is the Laplacian matrix of $\mathcal{H}$.*

We use Proposition 8 to inspect the graph convolutions. The second term of Eq. (2) can be written as $\frac{\boldsymbol{\theta}_1}{\lambda_{\mathcal{G}\text{max}}}\boldsymbol{B}_{\mathcal{G}}\boldsymbol{B}_{\mathcal{G}}^\top\boldsymbol{h}$, and $\boldsymbol{B}_{\mathcal{G}}^\top\boldsymbol{h} \in \mathbb{R}^m$ corresponds to the computation $\{x_v - x_u | (u,v) \in \mathcal{E}_{\mathcal{G}}\}$ in the edge space. We can design a better filter to replace this subtraction operation so that $\mathbf{\Theta} \star \boldsymbol{h} \approx (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_1)\boldsymbol{h} + \frac{\boldsymbol{\theta}_1}{\lambda_{\mathcal{G}\text{max}}}\boldsymbol{B}_{\mathcal{G}}\boldsymbol{z}$, where z is the result of some specific computation in the edge space, which is straightforward to involve Eq. (3). We are able to generalize Eq. (3) by the same idea, but it does not help to reduce the complexity. The second term of Eq. (3) is equivalent to $\frac{2\gamma_1}{\lambda_{\mathcal{H}\text{max}}}(\boldsymbol{D}_{\mathcal{H}} + \boldsymbol{I}_m)\boldsymbol{z} - \frac{\gamma_1}{\lambda_{\mathcal{H}\text{max}}}\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\hat{\boldsymbol{B}}_{\mathcal{G}}\boldsymbol{z}$ obtained from Proposition 9. Moreover, $\hat{\boldsymbol{B}}_{\mathcal{G}}\boldsymbol{z} \in \mathbb{R}^n$ corresponds to the computation $\{\sum_{(u,v)\in\mathcal{E}_{\mathcal{G}}} \boldsymbol{z}_{uv} + \sum_{(v,u)\in\mathcal{E}_{\mathcal{G}}} \boldsymbol{z}_{vu} | v \in \mathcal{V}_{\mathcal{G}}\}$. We can also enhance this computation by introducing $\boldsymbol{h}$, e.g., $\mathbf{\Gamma} \star \boldsymbol{z} \approx (\gamma_0 - \gamma_1)\boldsymbol{z} + \frac{2\gamma_1}{\lambda_{\mathcal{H}\text{max}}}(\boldsymbol{D}_{\mathcal{H}} + \boldsymbol{I}_m)\boldsymbol{z} - \frac{\gamma_1}{\lambda_{\mathcal{H}\text{max}}}\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\boldsymbol{h}$. We can get the degree matrix $\boldsymbol{D}_{\mathcal{H}}$ without constructing the line graph $\mathcal{H}$ because it depends on the vertex degrees of $\mathcal{G}$: $\{d_{g(e)}^- = d_u^-, d_{g(e)}^+ = d_v^+ | e = (u,v) \in \mathcal{E}_{\mathcal{G}}\}$. We manually set $\lambda_{\mathcal{H}_{\text{max}}} = \max\{d_u + d_v | (u,v) \in \mathcal{E}_{\mathcal{H}}\} = \max\{d_u^- + d_v^+ | (u,v) \in \mathcal{E}_{\mathcal{G}}\}$.

Finally, the asynchronous updates are defined as follows:

$$\boldsymbol{h}^{(k)} \leftarrow (\boldsymbol{\theta}_0^{(k)} - \boldsymbol{\theta}_1^{(k)})\boldsymbol{h}^{(k-1)} + \frac{\boldsymbol{\theta}_1^{(k)}}{\lambda_{\mathcal{G}\text{max}}}\boldsymbol{B}_{\mathcal{G}}\boldsymbol{z}^{(k-1)}, \tag{4}$$

$$\boldsymbol{z}^{(k)} \leftarrow (\boldsymbol{\gamma}_0^{(k)} - \boldsymbol{\gamma}_1^{(k)})\boldsymbol{z}^{(k-1)} + \frac{2\boldsymbol{\gamma}_1^{(k)}}{\lambda_{\mathcal{H}\text{max}}}(\boldsymbol{D}_{\mathcal{H}} + \boldsymbol{I}_m)\boldsymbol{z}^{(k-1)} - \frac{\boldsymbol{\gamma}_1^{(k)}}{\lambda_{\mathcal{H}\text{max}}}\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\boldsymbol{h}^{(k-1)}, \tag{5}$$

where $\boldsymbol{\theta}_:^{(k)}$ and $\boldsymbol{\gamma}_:^{(k)}$ indicate the parameters at the k-th update and $\boldsymbol{h}^{(k)}$ and $\boldsymbol{z}^{(k)}$ are the updated results. The computation of $\boldsymbol{B}_{\mathcal{G}}\boldsymbol{z}^{(k)}$ and the computation of $\hat{\boldsymbol{B}}_{\mathcal{G}}^\top\boldsymbol{h}^{(k)}$ are linear to the number of edges m with the help of sparse representations for $\boldsymbol{B}_{\mathcal{G}}$ and $\hat{\boldsymbol{B}}_{\mathcal{G}}$.

Heterogeneous Multi-graph Extensions

Different relational message passing variants have been proposed to model heterogeneous graphs. Nevertheless, our dual message passing is natural to handle complex edge types and even edge features. Each edge not only carries the edge-level property, but also stores the local structural information in the corresponding line graph. However, Eq. (5) does not reflect the edge direction since $\hat{\boldsymbol{B}}_{\mathcal{G}}$ regards the source and the target of one edge as the same. Therefore, we extend Eqs. (4-5) and propose dual message passing neural networks (DMPNNs) to support the mixture of various properties:

$$\boldsymbol{H}^{(k)} = \boldsymbol{H}^{(k-1)}\boldsymbol{W}_{\theta_0}^{(k)} - (\hat{\boldsymbol{B}}_{\mathcal{G}} - \boldsymbol{B}_{\mathcal{G}})\boldsymbol{Z}^{(k-1)}\boldsymbol{W}_{\theta_1^-}^{(k)} + (\hat{\boldsymbol{B}}_{\mathcal{G}} + \boldsymbol{B}_{\mathcal{G}})\boldsymbol{Z}^{(k-1)}\boldsymbol{W}_{\theta_1^+}^{(k)}, \tag{6}$$

$$\boldsymbol{Z}^{(k)} = \boldsymbol{Z}^{(k-1)}\boldsymbol{W}_{\gamma_0}^{(k)} + 2(\boldsymbol{D}_{\mathcal{H}} + \boldsymbol{I}_m)\boldsymbol{Z}^{(k-1)}(\boldsymbol{W}_{\gamma_1^-}^{(k)} - \boldsymbol{W}_{\gamma_1^+}^{(k)}) - (\hat{\boldsymbol{B}}_{\mathcal{G}} - \boldsymbol{B}_{\mathcal{G}})^\top\boldsymbol{H}^{(k-1)}\boldsymbol{W}_{\gamma_1^-}^{(k)} + (\hat{\boldsymbol{B}}_{\mathcal{G}} + \boldsymbol{B}_{\mathcal{G}})^\top\boldsymbol{H}^{(k-1)}\boldsymbol{W}_{\gamma_1^+}^{(k)}, \tag{7}$$

where $\boldsymbol{H}^{(k)} \in \mathbb{R}^{n\times l^{(k)}}$ and $\boldsymbol{Z}^{(k)} \in \mathbb{R}^{m\times l^{(k)}}$ are $l^{(k)}$-dim hidden states of nodes and edges in the k-th DMPNN layer. $\boldsymbol{H}^{(0)}$ and $\boldsymbol{Z}^{(0)}$ are initialized with features, labels, and other properties, $\hat{\boldsymbol{B}}_{\mathcal{G}} - \boldsymbol{B}_{\mathcal{G}}$ eliminates out-edges, $\hat{\boldsymbol{B}}_{\mathcal{G}} + \boldsymbol{B}_{\mathcal{G}}$ filters out in-edges, and $\boldsymbol{W}_{\theta_:}^{(k)}$ and $\boldsymbol{W}_{\gamma_:}^{(k)} \in \mathbb{R}^{l^{(k-1)}\times l^{(k)}}$ are trainable parameters that are initialized bounded by $\frac{\sqrt{\frac{6}{l^{(k-1)}+l^{(k)}}}}{\lambda_{\mathcal{G}\text{max}}}$ and $\frac{\sqrt{\frac{6}{l^{(k-1)}+l^{(k)}}}}{\lambda_{\mathcal{H}\text{max}}}$, respectively. For the detailed explanations and reparameterization tricks, see Appendix C. After K updates, we finally get $l^{(K)}$-dim node and edge representations $\boldsymbol{H}^{(K)}$ and $\boldsymbol{Z}^{(K)}$ in the aligned space.

Experiments

We evaluate DMPNNs on the challenging subgraph isomorphism counting and matching tasks. Besides, we also learn embeddings and classify nodes without any label or attribute

on heterogeneous graphs to verify the generalization and the necessity of explicit edge modeling. Training and testing of DMPNNs and baselines were conducted on single NVIDIA V100 GPU under PyTorch (Paszke et al. 2019) and DGL (Wang et al. 2019a) frameworks.

Subgraph Isomorphism Counting and Matching

DMPNNs are designed based on the duality of isomorphisms so that evaluation on isomorphism related tasks is the most straightforward. Given a pair of pattern $\mathcal{P}$ and graph $\mathcal{G}$, subgraph isomorphism counting aims to count all different subgraph isomorphisms in $\mathcal{G}$, and matching aims to seek out which nodes and edges belong to those isomorphic subgraphs. We report the root mean square error (RMSE) and the mean absolute error (MAE) between global counting predictions and the ground truth, and evaluate graph edit distance (GED) between predicted subgraphs and all isomorphic subgraphs. However, computing GED is NP-hard, so we consider the lower-bound of GED in contiguous space. We use DMPNN and baselines to predict the possible frequency of each node or edge appearing in isomorphic subgraphs. For example, models are expected to return $[2, 2, 2]$ for nodes and $[2, 2, 2]$ for edges given the pair in Figure 1a, and return $[1, 1, 1, 0]$ for nodes and $[1, 1, 1, 0, 0]$ for edges given Figure 1b. MAE between node predictions and node frequencies or the MAE between edge predictions and edge frequencies is regarded as the lower-bound of GED. We run experiments on three different seeds and report the best.

Models We compare with three sequence models and three graph models, including CNN (Kim 2014), LSTM (Hochreiter and Schmidhuber 1997), TXL (Dai et al. 2019), RGCN (Schlichtkrull et al. 2018), RGIN (Liu et al. 2020), and CompGCN (Vashishth et al. 2020). Sequence models embed edges, and we calculate the MAE over edges as the GED. On the contrary, graph models embed nodes so that we consider the MAE over nodes. We jointly train counting and matching prediction modules of DMPNN and other graph baselines:

$$\mathcal{J} = \frac{1}{|\mathcal{D}|} \sum_{(\mathcal{P},\mathcal{G})\in\mathcal{D}} \Big((c_{\mathcal{P},\mathcal{G}} - p_{\mathcal{P},\mathcal{G}})^2 + \sum_{v\in\mathcal{V}_\mathcal{G}} (w_{\mathcal{P},v} - p_{\mathcal{P},v})^2 \Big), \quad (8)$$

where $\mathcal{D}$ is the dataset containing pattern-graph pairs, $c_{\mathcal{P},\mathcal{G}}$ indicates the ground truth of number of subgraph isomorphisms between pattern $\mathcal{P}$ and graph $\mathcal{G}$, $c_{\mathcal{P},v}$ indicates the frequency of vertex v appearing in isomorphisms, $p_{\mathcal{P},\mathcal{G}}$ and $p_{\mathcal{P},v}$ are the corresponding predictions. For sequence models, we jointly minimize the MSE of counting predictions and the MSE of edge predictions. We follow the same setting of Liu et al. (2020) to combine multi-hot encoding and message passing to embed graphs and use pooling operations to make predictions:

$$\begin{aligned}
\boldsymbol{h}_v^{(0)} &= \text{Concat}(\text{MultiHot}(v), \text{MultiHot}(\mathcal{X}(v)))\boldsymbol{W}_{\text{vertex}}, \\
\boldsymbol{z}_e^{(0)} &= \text{MultiHot}(\mathcal{Y}(e))\boldsymbol{W}_{\text{edge}}, \\
\boldsymbol{H}^{(K)}, \boldsymbol{Z}^{(K)} &= \text{DMPNN}^{(K)}(\cdots(\text{DMPNN}^{(1)}(\boldsymbol{H}^{(0)}, \boldsymbol{Z}^{(0)}))), \\
\boldsymbol{p} &= \sum_{v\in\mathcal{P}} \boldsymbol{h}_{\mathcal{P}v}^{(K)}, \quad \boldsymbol{g} = \sum_{v\in\mathcal{G}} \boldsymbol{h}_{\mathcal{G}v}^{(K)},
\end{aligned}$$

	Erdős-Renyi		Regular		Complex		MUTAG	
#train	6,000		6,000		358,512		1,488	
#valid	4,000		4,000		44,814		1,512	
#test	10,000		10,000		44,814		1,512	
	Max	Avg	Max	Avg	Max	Avg	Max	Avg
$\lvert\mathcal{V}_\mathcal{P}\rvert$	4	3.8±0.4	4	3.8±0.4	8	5.2±2.1	4	3.5±0.5
$\lvert\mathcal{E}_\mathcal{P}\rvert$	10	7.5±1.7	10	7.5±1.7	8	5.9±2.0	3	2.5±0.5
$\lvert\mathcal{X}_\mathcal{P}\rvert$	1	1±0	1	1±0	8	3.4±1.9	2	1.5±0.5
$\lvert\mathcal{Y}_\mathcal{P}\rvert$	1	1±0	1	1±0	8	3.8±2.0	2	1.5±0.5
$\lvert\mathcal{V}_\mathcal{G}\rvert$	10	10±0	30	18.8±7.4	64	32.6±21.2	28	17.9±4.6
$\lvert\mathcal{E}_\mathcal{G}\rvert$	48	27.0±6.1	90	62.7±17.9	256	73.6±66.8	66	39.6±11.4
$\lvert\mathcal{X}_\mathcal{G}\rvert$	1	1±0	1	1±0	16	9.0±4.8	7	3.3±0.8
$\lvert\mathcal{Y}_\mathcal{G}\rvert$	1	1±0	1	1±0	16	9.4±4.7	4	3.0±0.1

Table 1: Statistics of datasets on subgraph isomorphism experiments. $\mathcal{P}$ and $\mathcal{G}$ corresponds to patterns and graphs.

$$\begin{aligned}
p_{\mathcal{P},v} &= \text{FC}_{\text{matching}}(\text{Concat}(\boldsymbol{h}_{\mathcal{G}v}^{(K)}, \boldsymbol{p}, \boldsymbol{h}_{\mathcal{G}v}^{(K)} - \boldsymbol{p}, \boldsymbol{h}_{\mathcal{G}v}^{(K)} \odot \boldsymbol{p})), \\
p_{\mathcal{P},\mathcal{G}} &= \text{FC}_{\text{counting}}(\text{Concat}(\boldsymbol{g}, \boldsymbol{p}, \boldsymbol{g} - \boldsymbol{p}, \boldsymbol{g} \odot \boldsymbol{p})),
\end{aligned}$$

where $\boldsymbol{W}_{\text{vertex}}$ and $\boldsymbol{W}_{\text{edge}}$ are trainable matrices to align id and label representations to the same dimension. We also consider the more powerful Deep-LRP (Chen et al. 2020) and add local relational pooling behind dual message passing for node representation learning, denoted as DMPNN-LRP. For a fair comparison, we use 3-layer networks and set the embedding dimensions, hidden sizes, and numbers of filters as 64 for all models. We follow the original setting of Deep-LRP to use 3-truncated BFS. Considering the quadratic computation complexity of TXL, we set the segment size and memory size as 128. All models are trained using AdamW (Loshchilov and Hutter 2019) with a learning rate 1e-3 and a decay 1e-5.

Datasets Table 1 shows the statistics of two synthetic homogeneous datasets with 3-stars, triangles, tailed triangles, and chordal cycles as patterns (Chen et al. 2020), one synthetic heterogeneous dataset with 75 random patterns,[1] and one mutagenic compound dataset *MUTAG* with 24 patterns (Liu et al. 2020). In traditional algorithms, adding reversed edges increases the search space dramatically, but it does not take too much extra time on neural methods. Thus, we also conduct experiments on patterns and graphs with reversed edges associated with specific edge labels , which doubles the number of edges and the number of edge labels.

Results Counting and matching results are reported in Table 2. We find graph models perform better than sequence models, and DMPNN almost surpasses all message passing based networks in counting and matching. RGIN extends RGCN with the sum aggregator followed by an MLP to makes full use of the neighborhood information, and it improves the original RGCN significantly. CompGCN is designed to leverage vertex-edge composition operations to predict the potential links, which is contrary to the goal of accurate matching. On the contrary, DMPNN learns both node embeddings and edge embeddings in aligned space but from different but dual structures. We also observe local re-

[1]This *Complex* dataset corresponds to the *Small* dataset in the original paper. But we found some ground truth counts are not correct because VF2 does not check self-loops. We removed all self-loops from patterns and graphs and got the correct ground truth.

Models	Homogeneous						Heterogeneous					
	Erdős-Renyi			Regular			Complex			MUTAG		
	RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED	RMSE	MAE	GED
Zero	92.532	51.655	201.852	198.218	121.647	478.990	68.460	14.827	86.661	16.336	6.509	15.462
Avg	121.388	131.007	237.349	156.515	127.211	576.476	66.836	23.882	156.095	14.998	10.036	27.958
CNN	20.386	13.316	NA	37.192	27.268	NA	41.711	7.898	NA	1.789	0.734	NA
LSTM	14.561	9.949	160.951	14.169	10.064	234.351	30.496	6.839	88.739	1.285	0.520	3.873
TXL	10.861	7.105	116.810	15.263	10.721	208.798	43.055	9.576	98.124	1.895	0.830	4.618
RGCN	9.386	5.829	28.963	14.789	9.772	70.746	28.601	9.386	64.122	0.777	0.334	1.441
RGIN	6.063	3.712	**22.155**	13.554	8.580	56.353	20.893	4.411	56.263	0.273	0.082	0.329
CompGCN	6.706	4.274	25.548	14.174	9.685	64.677	22.287	5.127	57.082	0.300	0.085	0.278
DMPNN	**5.062**	**3.054**	23.411	**11.980**	**7.832**	**56.222**	**17.842**	**3.592**	**38.322**	**0.226**	**0.079**	**0.244**
Deep-LRP	0.794	0.436	2.571	1.373	0.788	5.432	27.490	5.850	56.772	0.260	0.094	0.437
DMPNN-LRP	**0.475**	**0.287**	**1.538**	**0.617**	**0.422**	**2.745**	**17.391**	**3.431**	**35.795**	**0.173**	**0.053**	**0.190**

Table 2: Performance on subgraph isomorphism counting and matching.

Models		Complex			MUTAG		
		RMSE	MAE	GED	RMSE	MAE	GED
CNN	w/o rev	**41.711**	**7.898**	NA	**1.789**	**0.734**	NA
	w/ rev	47.467	10.128	NA	2.073	0.865	NA
LSTM	w/o rev	**30.496**	**6.839**	**88.739**	**1.285**	**0.520**	**3.873**
	w/ rev	32.178	7.575	90.718	1.776	0.835	5.744
TXL	w/o rev	43.055	9.576	98.124	**1.895**	**0.830**	**4.618**
	w/ rev	**37.251**	**9.156**	**95.887**	2.701	1.175	6.436
RGCN	w/o rev	28.601	9.386	64.122	0.777	0.334	**1.441**
	w/ rev	**26.359**	**7.131**	**49.495**	**0.511**	**0.200**	1.628
RGIN	w/o rev	20.893	4.411	56.263	0.273	**0.082**	**0.329**
	w/ rev	**20.132**	**4.126**	**39.726**	**0.247**	0.091	0.410
CompGCN	w/o rev	22.287	5.127	57.082	0.300	0.085	0.278
	w/ rev	**19.072**	**4.607**	**40.029**	**0.268**	**0.072**	**0.266**
DMPNN	w/o rev	18.974	3.922	56.933	0.232	0.088	0.320
	w/ rev	**17.842**	**3.592**	**38.322**	**0.226**	**0.079**	**0.244**
Deep-LRP	w/o rev	27.490	5.850	**56.772**	**0.260**	**0.094**	**0.437**
	w/ rev	**26.297**	**5.725**	61.696	0.290	0.108	0.466
DMPNN-LRP	w/o rev	20.425	4.173	42.200	0.196	0.062	0.210
	w/ rev	**17.391**	**3.431**	**35.795**	**0.173**	**0.053**	**0.190**

Table 3: Performance comparison after introducing reversed edges on heterogeneous data.

Models		MUTAG		Regular		Complex	
		RMSE	MAE	RMSE	MAE	RMSE	MAE
LSTM	MTL	1.285	0.520	14.169	10.064	30.496	6.839
	STL	-0.003	+0.030	+0.159	-0.029	-1.355	-0.096
TXL	MTL	1.895	0.830	14.306	10.143	37.251	9.156
	STL	-0.128	-0.041	+1.487	+1.211	-5.671	-2.067
RGCN	MTL	0.511	0.200	14.652	9.911	26.359	7.131
	STL	+0.202	+0.090	+0.348	-0.269	+1.686	+0.460
RGIN	MTL	0.247	0.091	13.128	8.412	20.132	4.126
	STL	+0.053	+0.004	+1.119	+1.019	+1.804	+0.068
CompGCN	MTL	0.268	0.072	14.174	9.685	19.072	4.607
	STL	+0.088	+0.086	+0.252	+0.738	+3.625	+0.260
DMPNN	MTL	0.226	0.079	11.980	7.832	17.842	3.592
	STL	+0.011	+0.001	+0.318	+0.097	+3.604	+0.865
Deep-LRP	MTL	0.260	0.094	1.275	0.731	26.297	5.725
	STL	+0.099	+0.044	+0.036	+0.035	+3.753	+0.886
DMPNN-LRP	MTL	0.173	0.053	0.617	0.422	17.391	3.431
	STL	+0.040	+0.020	+0.513	+0.252	+4.263	+0.928

Table 4: Performance comparison in multi-task training (MTL) and single-task training (STL) on subgraph isomorphism counting. We report best results of whether adding reversed edges or not, and error increases are underlined.

lational pooling can significantly decrease errors on homogeneous data by explicitly permuting neighbor subsets. But Deep-LRP is designed for patterns within three nodes and simple graphs so that it cannot handle multi-edges in nature, let along complex structures in randomly generated data and real-life data. One advantage of DMPNN is to model heterogeneous nodes and edges in the same space. We can see the success of DMPNN-LRP in three datasets with the maximum pattern size 4. But it struggles on the *Complex* dataset where patterns contain at most 8 nodes.

We also evaluate baselines with additional reversed edges on *Complex* and *MUTAG* datasets. From results in Table 3, we see graph convolutions consistently reduce errors with reversed edges, but sequence models usually become worse. LRP is designed for simple graphs so that it cannot handle heterogeneous edges in nature, but DMPNN makes it generalized. This observation also indicates that one of the challenges on neural subgraph isomorphism counting and matching is the complex graph local structure instead of the number of edges in graphs; otherwise, revised edges were toxic. We compare the efficiency in Appendix D.

In the joint learning, we hope models can learn the mutual supervision that node weights determine the global count and the global count is the upper bound of node weights. We also conduct experiments on single task learning to examine whether models can benefit from this mutual supervision. As shown in Table 4, graph models consistently achieve further performance gains from multi-task learning, while sequence models cannot. Moreover, improvement is more notable if the dataset is more complicated, e.g., patterns with more edges and graphs with non-trivial structures.

Unattributed Unsupervised Node Classification

Unattributed unsupervised node classification focuses on local structures instead of node features and attributes. Node embeddings are learned with the link prediction loss, then linear support vector machines are trained based on 80% of labeled node embeddings to predict the remaining 20%. We report the average Macro-F1 and Micro-F1 on five runs.

Models We follow the setting of RGCN and CompGCN: graph neural networks first learn the node representations, and then DistMult models (Yang et al. 2015) take pairs of node hidden representations to produce a score for a triplet $\langle u, y, v \rangle$, where u, y, v are the source, the edge type, and the target, respectively. Eq. (9) is the objective function, where

Dataset	$\lvert\mathcal{V}_\mathcal{G}\rvert$	$\lvert\mathcal{E}_\mathcal{G}\rvert$	$\lvert\mathcal{Y}_\mathcal{G}\rvert$	#Label type	#Labeled node
PubMed	63,109	244,986	10	8	454
Yelp	82,465	30,542,675	4	16	7,417

Table 5: Statistics of two real-life heterogeneous networks on unattributed unsupervised node classification.

$\mathcal{D} = \{\langle u, y, v\rangle | (u, v) \in \mathcal{E}_\mathcal{G}, y \in \mathcal{Y}_\mathcal{G}((u, v))\}$ is the triplet collection of graph $\mathcal{G}$, $s_y(u, v)$ is the score for $\langle u, y, v\rangle$, and $\langle u'_t, y, v'_t\rangle$ is one of the T negative triplets sampled from $\mathcal{G}$ by replacing u with u'_t or v with v'_t uniformly:

$$\mathcal{J} = -\frac{1}{|\mathcal{D}|}\sum_{\langle u,y,v\rangle\in\mathcal{D}}\Big(\log\sigma(s_y(\boldsymbol{h}_u,\boldsymbol{h}_v)) - \frac{1}{T}\sum_{t=1}^{T}\log(1-\sigma(s_y(\boldsymbol{h}_{u'_t},\boldsymbol{h}_{v'_t})))\Big). \quad (9)$$

We report the results of KG embedding models, proximity-preserving based embedding methods, graph convolutional networks, and graph attention networks for comparison. We use the same parameter setting as Yang et al. (2020).

Datasets Yang et al. (2020) collected and processed two heterogeneous networks to evaluate graph embedding algorithms. PubMed is a biomedical network constructed by text mining and manual processing where nodes are labeled as one of eight types of diseases; Yelp is a business network where nodes may have multiple labels (businesses, users, locations, and reviews). Statistics are summarized in Table 5.

Results In Table 6, we observe low F1 scores on both datasets and the difficulty of this task. Traditional KG embedding methods perform very similarly, but graph neural networks vary dramatically. RGCN and RGIN adapt the same relational transformations, but RGIN surpasses RGCN because of sum aggregation and MLPs. HAN and MAGNN explicitly learn the node representations from meta-paths and meta-path neighbors, but these models are evidently easy to overfit to training data because they predict the connectivity with the leaky edge type information. On the contrary, CompGCN and HGT obtain better scores since CompGCN incorporates semantics by node-relation composition, and HGT captures semantic relations and injects edge dependencies by relation-specific matrices. Our DMPNN outperforms all baselines by asynchronously learning node embeddings and edge representations in the same aligned space. Even for the challenging 16-way multi-label classification, DMPNN also works without any node attributes.

Models	PubMed		Yelp	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
TransE‡ (Bordes et al. 2013)	11.40	15.16	5.05	23.03
DistMult‡ (Yang et al. 2015)	11.27	15.79	5.04	23.00
ConvE‡ (Dettmers et al. 2018)	13.00	14.49	5.09	23.02
metapath2vec‡ (Dong, Chawla, and Swami 2017)	12.90	15.51	5.16	23.32
HIN2vec‡ (Fu, Lee, and Lei 2017)	10.93	15.31	5.12	23.25
HEER‡ (Shi et al. 2018)	11.73	15.29	5.03	22.92
RGCN‡ (Schlichtkrull et al. 2018)	10.75	12.73	5.10	23.24
RGIN (Liu et al. 2020)	12.22	15.41	5.14	23.82
CompGCN (Vashishth et al. 2020)	13.89	21.13	5.09	23.96
HAN‡ (Wang et al. 2019b)	9.54	12.18	5.10	23.24
MAGNN‡ (Fu et al. 2020)	10.30	12.60	5.10	23.24
HGT‡ (Hu et al. 2020)	11.24	18.72	5.07	23.12
DMPNN	**16.54**	**23.13**	**12.74**	**29.12**

Table 6: F1 scores (%) on unattributed unsupervised node classification. Results of ‡ are taken from (Yang et al. 2020).

Related Work

The isomorphism search aims to find all bijections between two graphs. The subgraph isomorphism search is more challenging, and it has been proven to be an NP-complete problem. Most subgraph isomorphism algorithms are based on backtracking or graph-index (Ullmann 1976; He and Singh 2008). However, these algorithms are hard to be applied to complex patterns and large data graphs. The search space of backtracking methods grows exponentially, and the latter requires a large quantity of disk space to index. Some methods introduce weak rules to reduce search space in most cases, such as candidate region filtering, partial matching enumeration, and ordering (Carletti et al. 2018). On the other hand, there are many approximate techniques for subgraph counting, such as path sampling (Jha, Seshadhri, and Pinar 2015) and color coding (Bressan, Leucci, and Panconesi 2019). But most approaches are hard to generalize to complex heterogeneous multi-graphs (Sun and Luo 2020).

In recent years, graph neural networks (GNNs) and message passing networks (MPNNs) have achieved success in graph data modeling. There are also some discussions about isomorphisms. Xu et al. (2019) and Morris et al. (2019) showed that neighborhood-aggregation schemes are as stronger as Weisfeiler-Leman (1-WL) test. Chen et al. (2020) proved that k-WL cannot count all patterns more than k nodes accurately, but the bound of T iterations of k-WL grows quickly to $(k + 1)2^T$. These conclusions encourage researchers to empower message passing and explore the possibilities of neural subgraph counting. Empirically, Liu et al. (2020) combined graph encoding and dynamic memory networks to count subgraph isomorphisms in an end-to-end way. They showed the memory with linear-complexity read-write operations can significantly improve all encoding models. A more challenging problem is subgraph isomorphism matching. NeuralMatch (Ying et al. 2020) utilizes neural methods and a voting method to detect subgraph matching. However, it only returns whether one pattern is included in the data graph instead of specific isomorphisms. Neural subgraph matching is still under discussion. Besides, graph learning also applies on maximum common subgraph detection (Bai et al. 2021), providing another possible solution for isomorohisms.

Conclusion

In this paper, we theoretically analyze the connection between the edge-to-vertex transform and the duality of isomorphisms in heterogeneous multi-graphs. We design dual message passing neural networks (DMPNNs) based on the equivalence of isomorphism searching over original graphs and line graphs. Experiments on subgraph isomorphism counting and matching as well as unsupervised node classification support our theoretical exposition and demonstrate effectiveness. We also see huge performance boost in small patterns by stacking dual message passing and local relational pooling. We defer a better integration as future work.

Acknowledgements

The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20) and the GRF (16211520) from RGC of Hong Kong, the MHKJFS (MHP/001/19) from ITC of Hong Kong with special thanks to HKMAAC and CUSBLT, and the Jiangsu Province Science and Technology Collaboration Fund (BZ2021065). We thank Dr. Xin Jiang for his valuable comments and the Gift Fund from Huawei Noah's Ark Lab.

References

Bai, Y.; Xu, D.; Sun, Y.; and Wang, W. 2021. GLSearch: Maximum Common Subgraph Detection via Learning to Search. In *ICML*, volume 139, 588–598.

Bhattarai, B.; Liu, H.; and Huang, H. H. 2019. CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching. In *SIGMOD*, 1447–1462.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*, 2787–2795.

Bressan, M.; Leucci, S.; and Panconesi, A. 2019. Motivo: Fast Motif Counting via Succinct Color Coding and Adaptive Sampling. *VLDB*, 12(11): 1651–1663.

Carletti, V.; Foggia, P.; Saggese, A.; and Vento, M. 2018. Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3. *TPAMI*, 40(4): 804–818.

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J. 2020. Can Graph Neural Networks Count Substructures? In *NeurIPS*.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *ACL*, 2978–2988.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*, 1811–1818.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*.

Fu, T.; Lee, W.; and Lei, Z. 2017. HIN2Vec: Explore Metapaths in Heterogeneous Information Networks for Representation Learning. In *CIKM*, 1797–1806.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*, 2331–2341.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, volume 70, 1263–1272.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2011. Wavelets on graphs via spectral graph theory. *ACHA*, 30(2): 129–150.

He, H.; and Singh, A. K. 2008. Graphs-at-a-time: query language and access methods for graph databases. In *SIGMOD*, 405–418. ACM.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous Graph Transformer. In *WWW*, 2704–2710.

Jha, M.; Seshadhri, C.; and Pinar, A. 2015. Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts. In *WWW*, 495–505.

Jin, M.; Chang, H.; Zhu, W.; and Sojoudi, S. 2021. Power up! Robust Graph Convolutional Network via Graph Powering. In *AAAI*, 8004–8012.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, 1746–1751.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Liu, X.; Pan, H.; He, M.; Song, Y.; Jiang, X.; and Shang, L. 2020. Neural Subgraph Isomorphism Counting. In *KDD*, 1959–1969.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.

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 *AAAI*, 4602–4609.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Schlichtkrull, M. S.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*, volume 10843, 593–607.

Shi, Y.; Zhu, Q.; Guo, F.; Zhang, C.; and Han, J. 2018. Easing Embedding Learning by Comprehensive Transcription of Heterogeneous Information Networks. In *SIGKDD*, 2190–2199.

Sun, S.; and Luo, Q. 2020. In-Memory Subgraph Matching: An In-depth Study. In *SIGMOD*, 1083–1098.

Ullmann, J. R. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1): 31–42.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. P. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

Wang, M.; Zheng, D.; Ye, Z.; Gan, Q.; Li, M.; Song, X.; Zhou, J.; Ma, C.; Yu, L.; Gai, Y.; Xiao, T.; He, T.; Karypis, G.; Li, J.; and Zhang, Z. 2019a. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019b. Heterogeneous Graph Attention Network. In *WWW*, 2022–2032.

Weisfeiler, B.; and Leman, A. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9): 12–16.

Whitney, H. 1932. Congruent Graphs and the Connectivity of Graphs. *AJMA*, 54(1): 150–168.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *TKDE*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*, 974–983.

Ying, R.; Lou, Z.; You, J.; Wen, C.; Canedo, A.; and Leskovec, J. 2020. Neural Subgraph Matching. *CoRR*, abs/2007.03092.

Zhang, X.-D. 2011. The Laplacian eigenvalues of graphs: a survey. *arXiv preprint arXiv:1111.2897*.