A Simple and Expressive Graph Neural Network Based Method for Structural Link Representation

Veronica Lachi [1] **Francesco Ferrini** [2] **Antonio Longa** [2] **Bruno Lepri** [1] **Andrea Passerini** [2]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

Graph Neural Networks (GNNs) have achieved state-of-the-art results in tasks like node classification, link prediction, and graph classification. While much research has focused on their ability to distinguish graphs, fewer studies have addressed their capacity to differentiate links, a complex and less explored area. This paper introduces SLRGNN, a novel, theoretically grounded GNN-based method for link prediction. SLRGNN ensures that link representations are distinct if and only if the links have different structural roles within the graph. Our approach transforms the link prediction problem into a node classification problem on the corresponding line graph, enhancing expressiveness without sacrificing efficiency. Unlike existing methods, SLRGNN computes link probabilities in a single inference step, avoiding the need for individual subgraph constructions. We provide a formal proof of our method's expressiveness and validate its superior performance through experiments on real-world datasets. The code is publicly available[1].

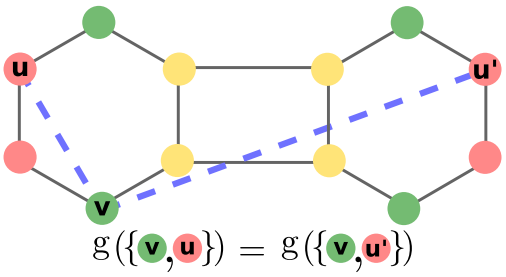

Figure 1. Links (u, v) and (u', v) are not set-isomorphic. Nevertheless, if we use a GNN to learn individual node representations and then aggregate them to represent a link, both (u, v) and (u', v) will receive identical predictions. Indeed, a GNN that is as powerful as the WL test will assign the same output (denoted by colors) to nodes u and v. Therefore, no matter how powerful the aggregation function g is, since the inputs are identical, the representations of the two links will collapse.

1. Introduction

Graph Neural Networks (GNNs) (Scarselli et al., 2008; Micheli, 2009; Kipf & Welling, 2016a) have proven to be a powerful tool, consistently achieving state-of-the-art performance in several tasks such as node classification (Hamilton et al., 2017; Veličković et al., 2017; Kipf & Welling, 2016a; Gasteiger et al., 2018; Wu et al., 2019), link prediction (Zhang & Chen, 2018; Zhang et al., 2021), subgraph classification (Alsentzer et al., 2020; Besta et al., 2022; Liu et al., 2022), and graph classification (Errica et al., 2019; Xu et al., 2019; Samoaa et al., 2022). Driven by the increasing application of GNNs across various domains, significant theoretical research has been conducted on their expressive power (Sato, 2020). In literature, the term expressiveness of GNNs refers to their ability to distinguish different inputs. The first challenge in this direction is understanding when inputs are different. Indeed, when dealing with graph-structured data, the underlying topological structure makes it non-trivial to assess whether two nodes, two links, or two graphs are different. Most research has focused on the ability of GNNs to distinguish between entire graphs, a well-studied problem known as graph isomorphism problem (Grohe & Schweitzer, 2020). Determining if two graphs are isomorphic is a problem for which a polynomial-time solution is still lacking (Babai, 2016). An efficient alternative is the Weisfeiler-Lehman (WL) test (Weisfeiler & Leman,

[*]Equal contribution [1]Fondazione Bruno Kessler, Trento, Italy [2]University of Trento, Trento, Italy. Correspondence to: Veronica Lachi <vlachi@fbk.eu>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at the 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024.
Copyright 2024 by the author(s).

[1]https://github.com/francescoferrini/SLRGNN-Structural-Link-Representation-Graph-Neural-Network

1968), an iterative algorithm that approximates graph isomorphism testing. It has been shown that GNNs are at most as poweful as the WL test in distinguishing graphs (Xu et al., 2019; Morris et al., 2019; Bianchi & Lachi, 2024). Additionally, many alternatives have been proposed to go beyond the WL test's capabilities, and several methods to enhance the expressive power of GNNs have been suggested (Abboud et al., 2020; Bevilacqua et al., 2021; Morris et al., 2019; Lachi et al., 2023).

Far fewer studies have been conducted on the ability of GNNs to distinguish between pairs or, more generally, subsets of nodes (Srinivasan & Ribeiro, 2019; Zhang et al., 2021). Intuitively, the ideal scenario would be for a GNN to produce identical outputs if and only if subsets of nodes are structurally equivalent. However, the concept of structural equivalence is less intuitive and less studied compared to graph isomorphism, deserving thorough investigation and formalization. Specifically, subsets of nodes within the same graph share the same structural role if an automorphism of the graph maps one onto the other. Our paper begins by revisiting the basic concepts and definitions necessary for comparing subsets of nodes based on their structural roles in the graph. These definitions are fundamental for assessing the expressive capabilities of methods that produce representations for graph subsets.

Among all problems related to node subsets classification, link prediction is undoubtedly the most popular. GNN-based methods for link prediction have been widely used in many applications, such as friend recommendation in social network (Adamic & Adar, 2003), movie recommendation (Bennett et al., 2007) and protein-protein interaction (Jha et al., 2022). There are two main classes of such approaches: pure GNN methods like Graph AutoEncoder (GAE) and Variational AutoEncoder (VGAE) (Kipf & Welling, 2016b) and Labeling Trick (LT) methods like SEAL (Zhang et al., 2021). Pure GNN methods first apply a GNN to the entire network to compute a representation for each node. The candidate link is then predicted based on the aggregated representation of its two end nodes. While efficient, as prediction can be made for any potential link in a single inference step, pure GNN methods fail to distinguish pairs of links that are not structurally identical, like links (u, v) and (u', v) in Figure 1. To address this issue, LT methods apply a GNN to an enclosing subgraph around each link, where nodes in the subgraph are labeled differently according to their distances to the two end nodes before applying the GNN. Methods in this category are more expressive than pure GNN methods (Zhang et al., 2021), but they have a drawback: a new subgraph must be constructed for each link whose existence is to be predicted, leading to efficiency problems.

In this article, we propose a GNN-based method, called SLRGNN, that is intuitive, theoretically grounded, and more expressive than pure GNN methods, meaning it produces identical outputs for two links if and only if they have the same structural role. It is also more efficient than LT methods, because it allows to compute predictions for any pair of nodes in a single inference step, avoiding the need to calculate a subgraph for each link. Our proposed method relies on transforming a link prediction problem on a graph into a node classification problem on the corresponding line graph. As demonstrated by a formal proof, using an expressive GNN for node classification on the line graph results in link representations that are distinct if and only if their structural roles within the graph differ. Furthermore, leveraging the inductiveness of GNNs, predictions can be made for any new node on the line graph, corresponding to any possible link in the original graph. Our main contributions are the following:

- We provide a theoretical framework with formal concepts and definitions necessary to understand the problem of the expressive power of GNNs regarding distinguishing links or, more generally, subsets of nodes;
- We analyze existing GNN-based link prediction methods, highlighting their strengths and weaknesses;
- We propose a new method, SLRGNN, based on the construction of the line graph that is simple and allows to perform link prediction on any pair of nodes in a single inference step;
- We formally prove the expressiveness of SLRGNN;
- We show through a set of experiments that SLRGNN achieves better performance than competitors on most real-world datasets.

2. Preliminaries

This section outlines basic definitions and foundational concepts that will be used in the rest of the paper, including node permutation, set-isomorphism and structural representation.

Definition 2.1 (*graph*)**.** A **graph** is a tuple $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}_G^0)$ where $V_G = \{1, \dots, n\}$ is a set of nodes, $E_G \subseteq V_G \times V_G$ is a set of $m = |E_G|$ edges, $\mathbf{A}_G \in \{0,1\}^{n \times n}$ is the adjacency matrix, with $\mathbf{A}_{G_{i,j}} = 1$ if and only if $(i, j) \in E_G$ and $\mathbf{X}_G^0 \in \mathbb{R}^{n \times d}$ is the node features matrix. In our analysis, we consider just simple and undirected graphs.

For ease of understanding, we restrict ourselves to analysis within a single graph G; however, the following definitions and considerations can be easily extended to comparisons between different graphs.

Definition 2.2 (*node permutation*)**.** A **node permutation** $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ is a bijective function that

assigns a new index to each node of the graph. All the $n!$ possible node permutations constitute the permutation group Π_n. Given a subset of nodes $S \subseteq V_G$, we define the permutation π on S as $\pi(S) := \{\pi(i) | i \in S\}$. Additionally, we define $\pi(\mathbf{A}_G)$ as the matrix $\mathbf{A}_G$ with rows and columns permutated based on π, i.e., $\pi(\mathbf{A}_G)_{\pi(i),\pi(j)} = \mathbf{A}_{G_{i,j}}$.

Definition 2.3 (*automorphism*)**.** An **automorphism** on the graph $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}_G^0)$ is a permutation $\pi \in \Pi_n$ such that $\pi(\mathbf{A}_G) = \mathbf{A}_G$. All the possible automorphisms on a graph constitute the automorphism group Σ_n.

Generally, not all node permutations are also automorphisms. This is exclusively true for complete graphs, where every permutation of nodes maintains the graph's adjacency relationships and so, Σ_n is isomorphic to the symmetry group.

Definition 2.4 (*set-isomorphism*)**.** Given a graph $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}_G^0)$, two node subsets $S, S' \subseteq V_G$ are **set-isomorphic** ($S \simeq_G S'$) if there exists an automorphism $\sigma \in \Sigma_n$ such that $\sigma(S) = S'$.

The concept of set-isomorphism is pivotal for analyzing the discriminative power of functions that generate representations of node subsets. In graph learning, a desirable property for learned functions is their ability to produce distinct outputs for different inputs. However, assessing the diversity of inputs with a topological structure is challenging. When it comes to comparing graphs, this is precisely the issue of graph isomorphism, which has been extensively studied (Grohe & Schweitzer, 2020; Babai, 2016; Babai & Kucera, 1979; Kobler et al., 2012). In contrast, when comparing nodes or subsets of nodes, the problem is less clear and less explored (Morris et al., 2023; Srinivasan & Ribeiro, 2019; Zhang et al., 2021). Thanks to Definition 2.4, we can assert that a function that learns representations for subsets (of any cardinality) of nodes should ideally generate identical representations if and only if the subsets are set-isomorphic.

Definition 2.5 (*structural representation*)**.** Let $S \subseteq V_G$ and $f(S, G)$ be a permutation invariant function that aims at learning a representation for the node set S, given G. $f(S, G)$ is said to be a **structural representation** if, for all $S, S' \subseteq V_G$ $f(S, G) = f(S', G)$ if and only if $S \simeq_G S'$.

The definition applies generally to any cardinality of S. However, a particularly relevant case for graph representation learning is link prediction, where $|S| = 2$. Here, we refer to it as **structural link representation**. Additionally, the scenario $|S| = 1$, corresponding to node feature learning, is also significant for this paper and will be called **structural node representation**.

3. Link Prediction with GNNs

3.1. Link Prediction Task Formulation

Link prediction aims to predict the likelihood of a link existing between two nodes given the structural and feature information. The link prediction task can be formalized as learning a function that maps a node pair to a probability score, i.e.,

$$f_{LP}(\{u, v\}, G) = p \in [0, 1]. \tag{1}$$

Traditionally, p was estimated via non-learnable heuristic methods (Liben-Nowell & Kleinberg, 2003; Menon & Elkan, 2011). More recently, methods that use learnable parameters have gained popularity (Zhang & Chen, 2018; Chamberlain et al., 2022). These methods attempt to estimate p via a learnable function

$$f_{LP_\Theta}(\{u, v\}, G; \Theta) = p \in [0, 1]. \tag{2}$$

where Θ represents a set of learnable parameters. A common choice of f_{LP_Θ} are GNN-based methods.

A model for link prediction is effective if it learns a function that is a structural link representation, as defined in the Definition 2.5.
In the following subsections, we first recall how GNNs work; then, different ways of using GNNs for link prediction are discussed, with a particular focus on whether they learn structural link representations.

3.2. Graph Neural Networks

GNNs are a class of neural network architectures specifically designed to process and analyze graphs structured data; GNNs rely on the so called message passing mechanism, which implements a local computational scheme to process graphs (Gilmer et al., 2017). Specifically, each feature vector of each node is updated by combining the features of the neighboring nodes. After l iterations, $\mathbf{x}_v^l$ embeds both the structural information and the content of the nodes in the l-hop neighborhood of v. More rigorously, the output of the l-th layer of a GNN is:

$$\mathbf{x}_v^l = \text{COMB}^{(l)}(\mathbf{x}_v^{l-1}, \text{AGG}^{(l)}(\{\mathbf{x}_u^{l-1},\, u \in N[v]\})) \tag{3}$$

where $\text{AGG}^{(l)}$ is a function that aggregates the node features from the neighborhood $N[v]$ at the $(l-1)$–th iteration, and $\text{COMB}^{(l)}$ is a function that combines the own features with those of the neighbors. The initial feature is the v-th row of the node feature matrix, i.e., $\mathbf{x}_v^0 = \mathbf{X}_{v,:}^0$.
This type of GNN implements permutation-invariant feature aggregation functions and the information propagation is isotropic (Tailor et al., 2021).
In node classification/regression tasks, a readout function

typically transforms the feature vector from the last layer L to produce the final output:

$$\mathbf{o}_v = \text{READ}(\mathbf{x}_v^L). \tag{4}$$

Depending on the specific task, the READ function can be the identity function, a sigmoid function, or a more sophisticated function (Buterez et al., 2022).
In graph classification/regression tasks, a global readout function transforms the output vectors to produce the final global output:

$$\mathbf{o} = \text{READ}(\{\mathbf{o}_v,\ v \in V_G\}). \tag{5}$$

The READ is implemented as the sum, mean, or the maximum of all node features, or by more elaborated functions (Bruna et al., 2013; Yuan & Ji, 2020; Khasahmadi et al., 2020).

Expressive Power of GNNs In the literature, studying the expressive power of GNNs refers to evaluating their ability to distinguish between different graphs. This capability is inherently linked to the graph isomorphism problem, a widely studied issue believed to lack a polynomial-time solution (Babai, 2016). The WL test (Weisfeiler & Leman, 1968) provides a computationally efficient and effective method for distinguishing a broad range of graphs. The algorithm assigns a color to each node based on the multiset of features of its neighbors and its own color. At each iteration, the colors of the nodes are updated until convergence is reached. If the color multisets of two graphs differ, the graphs are not isomorphic. Conversely, if the color multisets are identical, there are no guarantees that the two graphs are isomorphic. It has been proven that GNNs are at most as effective as the WL test in distinguishing between graphs (Xu et al., 2019). Thus, we can readily identify as failures of GNNs those cases where non-isomorphic graphs receive the same output from the WL test. From now on, we will refer to GNN models which are as powerful as the WL test as WL-GNNs. Relatively few studies have focused on the expressive capabilities of GNNs concerning representations of subsets of nodes and, particularly, links, which are of significant interest due to the widespread application of GNNs in link prediction. Ideally, a GNN must generate the same representation for two subsets of nodes if and only if they are set-isomorphic, but, as shown in the following, this is not always the case.

3.3. GNN-based approaches for link prediction

There are two fundamental approaches for performing link prediction using methods based on GNNs. In this section, we will explore these two categories, identifying the methods that fall into each one and discussing if they learn structural link representation along with their strengths and weaknesses.

Pure GNN Methods In the pure GNN Methods, a GNN is first applied to the entire graph to learn an embedding vector for each node. Then, the embeddings of pairs of nodes are aggregated. Formally, these methods learn a function:

$$f_{LP_\Theta}(\{u,v\}, G; \Theta) = g(\text{GNN}(u, G; \Theta), \text{GNN}(v, G; \Theta)) \tag{6}$$

where g is an aggregation function. In principle, any type of GNN can be used, and the function g can be modeled using an MLP over any aggregation function over the feature vectors. In practice, the most commonly used pure GNN model is GAE (Kipf & Welling, 2016b). This method employs a GCN (Kipf & Welling, 2016a) to generate node representations. These representations are then aggregated using the inner product, followed by the application of a sigmoid function to produce the final output.
As proved in (Zhang et al., 2021), pure GNN Methods cannot learn a structural link representation, even equipped with a GNN that learns a structural node representation. As an example of failure of pure GNN Methods, consider the graph in Figure 1. Node u and u' have identical structural roles within the graph, meaning there exists an automorphism that maps one to the other. As a result, a GNN that learns structural node representations will produce identical outputs for these nodes (as shown by the colors in the figure). Thus, when aggregating the representations to form link representations, even with a very powerful aggregation function, the representation of the link (u, v) will be identical to that of the link (u', v). Thus, pure GNN methods assign the same probability to these two potential links, even though $\{u, v\}$ is not set-isomorphic to $\{u', v\}$.

Labeling Trick Methods Another category of GNN-based approaches comprises models that employ the so-called Labeling Trick (LT) (Zhang et al., 2021). The most frequently used model in this category is SEAL (Zhang & Chen, 2018; Li et al., 2020). This method labels nodes according to their relationship to the target link, followed by applying a GIN (Xu et al., 2019) to the labeled graph. The representations of the target nodes generated within the labeled graph are then aggregated to form the link representation. This mechanism has been shown to significantly enhance the link discrimination capability of the models, as they learn structural link representations effectively.
In particular, the SEAL model operates by extracting an h-hop enclosing subgraph centered around the two target nodes and bases its link prediction on the topology of this subgraph. Consequently, the task of link prediction is transformed into a graph classification problem, with the model treating the enclosing subgraph as the input to determine the presence of a link between the nodes.
While this approach offers substantial improvements to the link discrimination power of GNNs, it also presents certain challenges. It requires repeatedly applying GNNs to the la-

beled subgraph for each link prediction, unlike approaches like GAE, which can process the entire graph in a single step and simultaneously learn representations for all target links. Consequently, methods that use the LT are less efficient because they cannot generate all link predictions in a single GNN inference step.

Inspired by the SEAL approach, (Cai et al., 2021) suggests to directly learning the features of the target link rather than extracting features from the entire enclosing subgraph. Specifically, they transform the original enclosing subgraph into its corresponding line graph, allowing graph convolution layers to be directly applied to learn the node embeddings within the line graph. These embeddings are then used as features for the edges in the original graph to predict the existence of a link. Thus, the link prediction task is effectively viewed as a node classification problem.

However, this method encounters the same limitation as the SEAL model: it necessitates applying GNNs to a line graph for each link prediction, preventing it from generating all link predictions in a single GNN inference step. Furthermore, while the use of line graphs is effective for achieving a structural representation of links as demonstrated in Section 4.2, constructing line graphs solely for subgraphs may not be sufficient to capture the structural role of links, as shown in the example in Figure 2. Indeed, usually the chosen hop length is $h = 2$. The 2-hop subgraphs centered around links (u, v) and (u', v') are identical, and their corresponding line graphs are also equal; however $(u, v) \not\simeq_G (u', v')$. Thus, the two links will have the same representations despite not being set-isomorphic.

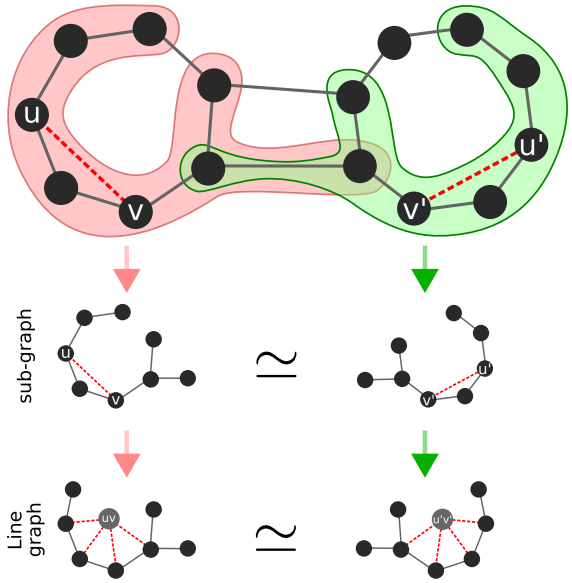

Figure 2. Links (u, v) and (u', v') are not set-isomorphic. However, their 2-hop subgraphs are identical, resulting in their line graphs also being identical.

4. Structural Link Representation using Line Graphs

4.1. Method

The aim of our approach is to learn a structural link representation while efficiently processing the entire graph in a single step to simultaneously learn representations for all potential target links.

We propose a new GNN-based method, called SLRGNN, which not only provides structural link representation but also boosts efficiency by generating all link predictions in a single GNN inference step. The proposed method consists of three steps, which are explained in detail below.

Line Graph Construction The initial step of SLRGNN entails constructing the line graph of the graph for which link prediction is desired. The formal definition of a line graph is as follows:

Definition 4.1 (*line graph*)**.** Given a graph $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}_G^0)$, its **line graph** is the graph $L_G = (V_{L_G}, E_{L_G}, \mathbf{A}_{L_G}, \mathbf{X}_{L_G}^0)$ such that $uv \in V_{L_G}$ if and only if $(u, v) \in E_G$ and $(uv, wz) \in E_{L_G}$ if and only if (u, v) and (w, z) are incident edges in G. Moreover, $\mathbf{X}_{L_G}^0 \in \mathbb{R}^{m \times m}$ is obtained starting from $\mathbf{X}_G^0$ using some node features aggregation method.

Contrary to the LGLP approach proposed in (Cai et al., 2021), SLRGNN constructs the line graph for the entire graph rather than for individual subgraphs centered around each link of interest. This methodology enables the simultaneous learning of representations for all potential target links. Furthermore, as proved in Section 4.2, SLRGNN effectively learns a structural representation of the links.

The line graph construction method serves as a preprocessing step that is performed only at the beginning of the method. The computational cost of constructing the entire line graph is $O(m^2)$ where m is the number of edges in the original graph. This complexity arises from the need to check every possible pair of edges to establish connections in the line graph; this step is the most computationally intensive, thereby defining the overall complexity.

While the topological structure of a line graph is uniquely defined by the line graph's definition, the node features can be constructed using different approaches. We detail our chosen methodology in the subsequent section.

Features Aggregation Once a line graph is constructed, the new features of the nodes in the line graph need to be determined based on the features of the nodes in the original graph. Some aggregation function must be used to aggregate the features of pairs of nodes that form a node in the line graph. In (Zaheer et al., 2017), it has been proven that there exists a function which, if applied to the addends of a sum, makes the sum injective. In practice, this function can

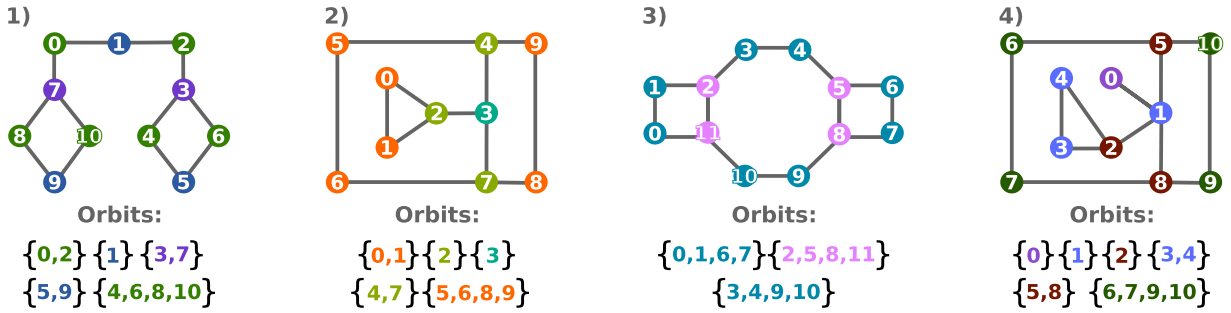

Figure 3. The 4 graphs out of 202,579 in the Alchemy dataset where the WL test and the automorphism group induce different partitions on the node set. Node colors represent the WL test output, while the orbits are indicated below each graphs.

be approximated using an MLP, as done in (Xu et al., 2019), since it has been shown that MLPs are universal approximators (Hornik et al., 1989). Using a similar approach, for each node $uv \in V_{LG}$, the initial feature $\mathbf{x}^0_{uv}$ is calculated as:

$$\mathbf{x}^0_{uv} = \text{MLP}_\theta(\mathbf{x}^0_v) + \text{MLP}_\theta(\mathbf{x}^0_u), \tag{7}$$

where $\mathbf{x}^0_u$ and $\mathbf{x}^0_v$ are respectively the u-th and v-th rows of the original feature matrix $\mathbf{X}^0_G \in \mathbb{R}^{n\times d}$.
The presence of the MLP guarantees the injectivity of the sum; therefore, two nodes $uv, u'v' \in V_{LG}$ in the line graph will have different features if and only if $\{\mathbf{x}^0_u, \mathbf{x}^0_v\} \neq \{\mathbf{x}^0_{u'}, \mathbf{x}^0_{v'}\}$. In the case where the original dataset does not have initial node features, it is possible to use node topological features such as the degree, the clustering coefficient or some centrality measure. An in-depth discussion on the selection of initial features is provided in Appendix D.

Training and Testing Once the input graph is transformed into its corresponding line graph, the link prediction problem can be formulated as a node classification one, i.e.,

$$f_{LP}(\{u, v\}, G) = f_{NC}(uv, L_G). \tag{8}$$

In our approach, this function is learned by training a GNN:

$$\mathbf{o}_{uv} = \text{GNN}(uv, L_G; \Theta). \tag{9}$$

Specifically as GNN layer, GIN (Xu et al., 2019) is employed, which is the most used WL-GNN:

$$\mathbf{x}^l_{uv} = \text{MLP}^{(l)}\left(\left(1+\epsilon^l\right)\cdot \mathbf{x}^{l-1}_{uv} + \sum_{wz\in\mathcal{N}_{(uv)}} \mathbf{x}^{l-1}_{wz}\right). \tag{10}$$

In the context of link prediction, negative sampling is used to introduce edges to the graph that the network should learn to classify as negative. Specifically, in SLRGNN, the negative links are added in the original graph between nodes that are at most h-hops apart, where h is an hyperparameter of the model. Negative edges are then transformed into negative nodes in the line graph. Consequently, the model is trained on the line graph to differentiate between nodes labeled as 0 (representing negative edges in the original graph) and nodes labeled as 1 (representing existing edges in the original graph). The training process employs the Binary Cross-Entropy loss function:

$$\mathcal{L}_{BCE} = -(\mathbf{y}_{uv}\log(\mathbf{o}_{uv}) + (1-\mathbf{y}_{uv})\log(1-\mathbf{o}_{uv})), \tag{11}$$

where $\mathbf{y}_{uv}$ is the label of node uv, i.e., $\mathbf{y}_{uv} = 1$ if the link (u, v) exists, $\mathbf{y}_{uv} = 0$ otherwise.

The MLP employed to ensure the injectiveness of the features aggregation and the GNN are trained in an end-to-end fashion.
After training, the GNN can be evaluated on unseen nodes by leveraging its inductive capabilities. Specifically, nodes representing links to be predicted are added to the line graph, connecting them to the existing nodes based on their topological relationships in the original graph. The trained model is then tested on these new nodes.
Formally, given the optimized parameters Θ^*, the likelihood of the target link (u^*, v^*) is:

$$\mathbf{o}_{u^*v^*} = \text{GNN}(u^*v^*, L_{G\cup(u^*,v^*)}; \Theta^*), \tag{12}$$

in which $G \cup (u^*, v^*) := (V_G, E_G \cup (u^*, v^*), A_G + A_{(u^*,v^*)}, \mathbf{X}^0_G)$, where $A_{(u^*,v^*)}$ is a $n \times n$ matrix such that $A_{(u^*,v^*)_{u^*,v^*}} = A_{(u^*,v^*)_{v^*,u^*}} = 1$ and all the other entries are equal to zero. Thus, the proposed method addresses the issue of (Cai et al., 2021; Zhang et al., 2021), enabling link prediction for any pair of nodes in a single inference step. Furthermore, as proved in the following section, the proposed method learns a structural link representation.

4.2. Theoretical Analysis

The following theorem shows that, under certain sufficient conditions, SLRGNN learns a structural link representation. These sufficient conditions are detailed in the subsequent discussion and in the theorem's proof.

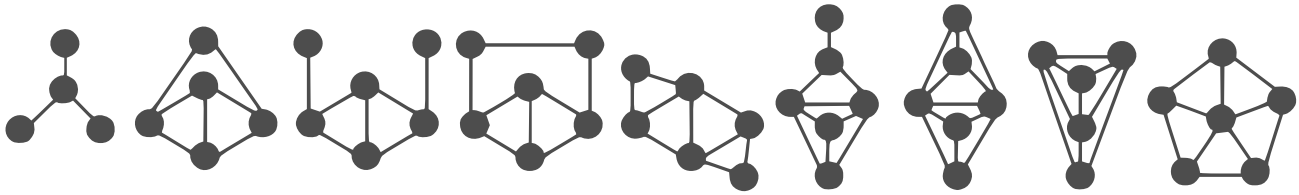

Figure 4. According to Beineke's theorem, a graph is a line graph if and only if it does not contain any of these specific graphs as induced subgraphs.

Before presenting the theorem, we recall that K_n denotes the complete graph with n nodes, and $K_{n,k}$ denotes the complete bipartite graph with n nodes in the first partition and k nodes in the second partition. For example, K_3 is a triangle, while $K_{1,3}$ is a complete bipartite graph, also known as a star graph. It consists of two sets of nodes, where one set has one node (the center) and the other set has three nodes. The center node is connected to each of the three outer nodes, but the outer nodes are not connected to each other. $K_{1,3}$ is the first graph in Figure 4.

Theorem 4.2. *Let $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}^0_G)$, $L_G = (V_{L_G}, E_{L_G}, \mathbf{A}_{L_G}, \mathbf{X}^0_{L_G})$ its line graph and $f_{LP}(\{u,v\}, G) = GNN(uv, L_{G\cup(u,v)}; \Theta)$. If*

1. *$G \cup (u,v)$ is connected and $G \cup (u,v) \neq K_3, K_{1,3}$ $\forall (u,v) \notin E_G$ with $u, v \in V_G$,*
2. *GNN is a node structural representation,*

then $f_{LP}(\{u,v\}, G)$ is a structural link representation.

Theorem 4.2 guarantees that, when the two conditions are met, the method proposed in Section 4.1 learns a structural link representation. Condition 1 is widely satisfied in most real and synthetic graph datasets, as it is met by all connected graphs with more than three nodes. On the other hand, Condition 2 is more complex and requires a more detailed analysis. This condition imposes that the employed GNN must be a node structural representation, meaning that for any two nodes, the GNN produces identical outputs if and only if there is an automorphism between the two nodes, i.e., if and only if the nodes are set-isomorphic. While the definition of set-ismorphism (Def. 2.4) can be applied to node sets of any cardinality, there is a specific definition for the case when $|S| = 1$:

Definition 4.3 (*similar nodes*)**.** Given a graph $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}^0_G)$ two nodes $u, v \in V_G$ are **similar** if there exists an automorphism $\sigma \in \Sigma_n$ such that $\sigma(u) = v$.

Node similarity induces an equivalence relation on V_G, where each equivalence class is an orbit.

Definition 4.4 (*orbit*)**.** The **orbit** of a vertex $v \in V_G$ under the action of the automorphism group Σ_n is defined as the set of nodes of V_G to which v can be mapped via an automorphism $\sigma \in \Sigma_n$. Formally:

$$\text{Orb}(v) = \{\sigma(v) \mid \sigma \in \Sigma_n\}. \tag{13}$$

The set of all the orbits of V_G with respect to the action of Σ_n is the partition of V_G in equivalence classes induced by Σ_n, i.e.,

$$V_{G_{/\Sigma_n}} = \{\text{Orb}(v) \mid v \in V_G\}. \tag{14}$$

A natural question that arises is whether a WL-GNN, such as GIN, is a node structural representation, i.e., whether it produce different outputs if and only if the nodes are not similar. The answer is provided in the following.

Proposition 4.5. *If WL-GNN assigns different outputs to $u, v \in V_G$, then u, v are non-similar.*

The proof can be found in (Morris et al., 2023). The converse implications is not true, i.e., there exist non-similar nodes for which the WL-GNN assigns the same output. Some examples are reported in Figure 3. Indeed, in each of these graphs, the set of the orbits is not equal to the partition of nodes induced by the WL test. For example, in graph 1), $V_{G_{/\Sigma_n}} = \{\{0,2\},\{4,6,8,10\},\{3,7\},\{5,9\},\{1\}\}$ while the partition induced by the WL test (denote by the colors in the figure) is $V_{G_{/WL}} = \{\{3,7\},\{0,2,4,6,8,10\},\{1,5,9\}\}$. Therefore, the output of a WL-GNN for node 1 is equal to the one for node 9, but these nodes are not similar. Thus, while a WL-GNNs ensure that similar nodes always have the same representation, they can fail in providing different representations for non-similar nodes.

Although there is no characterization of the failures of WL-GNNs in *distinguishing nodes*, these failures are fewer and less well-known compared to the failures of WL-GNNs in *distinguishing graphs*. It is said that WL-GNNs can give different representations to **almost all** non-similar nodes (Babai & Kucera, 1979). For instance, among the 202,579 graphs in the Alchemy dataset (Chen et al., 2019), WL test fails to distinguish similar nodes just in the four graphs of Figure 3.

Furthermore, in the context of our method, the WL-GNN is applied not to any arbitrary graph but specifically to a line

	BUP	C.ele	USAir	SMG	EML	NSC	YST	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	87	85	92	86	88	98	81	74	88	94	95	90	93	97
PR	90	89	94	89	89	98	81	75	92	94	96	91	93	98
SR	85	76	79	78	87	97	74	71	77	83	89	84	93	95
N2V	80	80	85	78	83	96	77	77	83	79	92	81	94	94
NRI	95	90	96	91	92	**100**	92	82	93	95	96	92	97	97
GAE	90	84	92	86	87	99	77	70	84	91	94	85	91	95
SEAL	93	87	95	92	92	**100**	91	84	93	95	**97**	93	**98**	98
LGLP	95	90	**97**	93	92	**100**	**92**	**85**	**94**	96	**97**	94	**98**	98
SLRGNN	**97**	**96**	94	**95**	**98**	83	**92**	84	**94**	**98**	96	**97**	**98**	**99**

Table 1. Averaged AUC results of proposed model and baselines. Best results are in bold. For standard deviations check Table 3 Appendix B

graph. According to Beineke's theorem (Beineke & Wilson, 2004), a graph is a line graph if and only if it does not contain any of the graphs shown in Figure 4 as induced subgraphs. Among these, the first one, i.e., $K_{1,3}$ is a structure commonly found in graphs. For instance, all four examples of WL-GNN failures depicted in Figure 3 contain $K_{1,3}$ as an induced subgraph, and thus cannot be line graphs.
To conclude, in practice, there are currently no known examples of graphs where WL-GNNs fail to distinguish similar nodes while also being line graphs, making condition 2 a reasonable assumption in real-world settings.

5. Experiments

We evaluate SLRGNN on 14 real-world datasets, namely BUP, Celegans, HPD, YST, SMG, NSC, KHN, GRQ, LDG, ZWL, USAir, EML, Power, and ADV (Watts & Strogatz, 1998; Newman, 2001). These datasets span various application domains and differ in the number of nodes and links, as detailed in Table 2 (Appendix B). We compare our results with three high-order heuristic methods: Katz (Katz, 1953), PageRank (PR) (Brin & Page, 2012), and SimRank (SR) (Jeh & Widom, 2002). Additionally, we benchmark against the graph embedding method node2vec (N2V) (Grover & Leskovec, 2016), neural relational inference (NRI) (Kipf et al., 2018) and the gnn-based methods (GAE) (Kipf & Welling, 2016b), SEAL (Zhang & Chen, 2018) and LGLP (Cai et al., 2021).

5.1. Experimental setting

To assess the performance of SLRGNN, we generated negative links in the original graph to ensure a balance between positive and negative links. The maximum number of hops between any two nodes selected for negative links in the original graph is set to $h = 4$. For graphs without node features, we created feature vectors using the clustering coefficients, the betweenness centrality, and the closeness centrality. The use of such topological features is valid because, if two nodes are similar, their topological features will be the same. Further elaboration on this is provided in Appendix D.
To optimize model performance, we explored a range of hyperparameters using grid search. Specifically learning rates ranging from 0.0001 to 0.01, weight decay from 10^{-6} to 10^{-5}, number of GNN layers from 1 to 4, embedding dimensions from 16 to 64 and number of MLP layers from 1 to 3. SLRGNN is trained on a NVIDIA GeForce GTX TITAN X gpu.

5.2. Experimental results

Table 1 presents the AUC results for SLRGNN and the baseline methods across various datasets. For the complete results with Standard Deviations see Table 3 in Appendix C. SLRGNN, consistently outperforms other methods, achieving the highest AUC scores on most datasets. Notably, SLRGNN achieves an AUC of 97 or higher on datasets such as BUP, C.ele, EML, ADV, HPD, GRQ, and ZWL, showcasing its robustness and generalization capability across different domains. LGLP and SEAL methods also demonstrate strong performance, with LGLP achieving a perfect AUC score of 100 on the NSC dataset and high scores on others. However, these methods require the computation of line graphs or subgraphs for each link prediction task, which can be computationally intensive and reduce efficiency. Heuristic methods and embedding-based methods, while performing well in certain cases, generally exhibit lower AUC scores compared to gnn-based methods. Overall, SLRGNN not only achieves superior performance in terms of AUC compared to other methods but also balances performance and efficiency, making it an effective method for link prediction tasks across diverse datasets.

6. Conclusions

This paper presents a novel and theoretically grounded approach for link prediction using GNNs, which consists of transforming the link prediction problem on a graph into a node classification task on the corresponding line graph. Our method ensures that link representations are distinct if and only if the links have different structural roles within

the graph, thereby addressing a significant limitation of existing GNN-based link prediction techniques. By avoiding the need for constructing individual subgraphs for each link and enabling single-step inference for link probabilities, our approach combines efficiency with enhanced expressiveness. The formal proof of expressiveness and the superior performance on real-world datasets underscore the effectiveness of our proposed method. Future research can build on this foundation to explore how to build theoretically grounded expressive GNNs for motif and subgraphs classification/regression, contributing to the broader field of graph representation learning.

Acknowledgements

Veronica Lachi and Bruno Lepri acknowledge funding by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120237 (ELIAS). Bruno Lepri, Andrea Passerini and Antonio Longa acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU. Bruno Lepri and Andrea Passerini also acknowledge funding by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120763 - TANGO. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them.

References

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

Adamic, L. A. and Adar, E. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

Alsentzer, E., Finlayson, S., Li, M., and Zitnik, M. Subgraph neural networks. *Advances in Neural Information Processing Systems*, 33:8017–8029, 2020.

Babai, L. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697, 2016.

Babai, L. and Kucera, L. Canonical labelling of graphs in linear average time. In *20th annual symposium on foundations of computer science (sfcs 1979)*, pp. 39–46. IEEE, 1979.

Beineke, L. W. and Wilson, R. J. *Topics in algebraic graph theory*, volume 102. Cambridge University Press, 2004.

Bennett, J., Lanning, S., et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, pp. 35. New York, 2007.

Besta, M., Grob, R., Miglioli, C., Bernold, N., Kwasniewski, G., Gjini, G., Kanakagiri, R., Ashkboos, S., Gianinazzi, L., Dryden, N., et al. Motif prediction with graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 35–45, 2022.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2021.

Bianchi, F. M. and Lachi, V. The expressive power of pooling in graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Brin, S. and Page, L. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Buterez, D., Janet, J. P., Kiddle, S. J., Oglic, D., and Liò, P. Graph neural networks with adaptive readouts. *Advances in Neural Information Processing Systems*, 35:19746–19758, 2022.

Cai, L., Li, J., Wang, J., and Ji, S. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

Chamberlain, B. P., Shirobokov, S., Rossi, E., Frasca, F., Markovich, T., Hammerla, N. Y., Bronstein, M. M., and Hansmire, M. Graph neural networks for link prediction with subgraph sketching. In *The eleventh international conference on learning representations*, 2022.

Chen, G., Chen, P., Hsieh, C.-Y., Lee, C.-K., Liao, B., Liao, R., Liu, W., Qiu, J., Sun, Q., Tang, J., Zemel, R., and Zhang, S. Alchemy: A quantum chemistry dataset for benchmarking ai models. *arXiv preprint arXiv:1906.09427*, 2019.

Cuypers, H. Whitney's theorem for line graphs of multigraphs. *arXiv preprint arXiv:2105.08610*, 2021.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2019.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grohe, M. and Schweitzer, P. The graph isomorphism problem. *Communications of the ACM*, 63(11):128–134, 2020.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.

Harary, F. *Graph theory (on Demand Printing of 02787)*. CRC Press, 2018.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jeh, G. and Widom, J. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 538–543, 2002.

Jha, K., Saha, S., and Singh, H. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360, 2022.

Jung, H. Zu einem isomorphiesatz von h. whitney für graphen. *Mathematische Annalen*, 164(3):270–271, 1966.

Katz, L. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

Khasahmadi, A. H., Hassani, K., Moradi, P., Lee, L., and Morris, Q. Memory-based graph networks. In *International Conference on Learning Representations*, 2020.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *International conference on machine learning*, pp. 2688–2697. PMLR, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Kobler, J., Schöning, U., and Torán, J. *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media, 2012.

Lachi, V., Moallemy-Oureh, A., Roth, A., and Welke, P. Graph pooling provably improves expressivity. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Liben-Nowell, D. and Kleinberg, J. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Liu, Y., Ma, J., and Li, P. Neural predicting higher-order patterns in temporal networks. In *Proceedings of the ACM Web Conference 2022*, pp. 1340–1351, 2022.

Menon, A. K. and Elkan, C. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II 22*, pp. 437–452. Springer, 2011.

Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

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

Morris, M., Grau, B. C., and Horrocks, I. Orbit-equivariant graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.

Newman, M. E. The structure of scientific collaboration networks. *Proceedings of the national academy of sciences*, 98(2):404–409, 2001.

Samoaa, H. P., Longa, A., Mohamad, M., Chehreghani, M. H., and Leitner, P. Tep-gnn: Accurate execution time prediction of functional tests using graph neural networks. In *International Conference on Product-Focused Software Process Improvement*, pp. 464–479. Springer, 2022.

Sato, R. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Srinivasan, B. and Ribeiro, B. On the equivalence between positional node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*, 2019.

Tailor, S. A., Opolka, F., Lio, P., and Lane, N. D. Do we need anisotropic graph neural networks? In *International Conference on Learning Representations*, 2021.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wasserman, S. and Faust, K. Social network analysis: Methods and applications. 1994.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

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

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks?, 2019.

Yuan, H. and Ji, S. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

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

Zhang, M., Li, P., Xia, Y., Wang, K., and Jin, L. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems*, 34:9061–9073, 2021.

Appendix

A. Proof

Theorem A.1. *Let $G = (V_G, E_G, \mathbf{A}_G, \mathbf{X}^0_G)$, $L_G = (V_{L_G}, E_{L_G}, \mathbf{A}_{L_G}, \mathbf{X}^0_{L_G})$ its line graph and $f_{LP}(\{u,v\}, G) =$ GNN$(uv, L_{G\cup(u,v)}; \Theta)$. If*

1. *$G \cup (u,v)$ is connected and $G \cup (u,v) \neq K_3, K_{1,3}$ $\forall (u,v) \notin E_G$ with $u, v \in V_G$,*
2. *GNN is a node structural representation,*

then $f_{LP}(\{u,v\}, G)$ is a structural link representation.

Proof. Let $u, v, u', v' \in V_G$ and $(u,v), (u',v') \notin E_G$. In order to prove that $f(\{u,v\}, G)$ is a structural link representation, by definitions 2.2,2.5,2.4 we need to prove:

$$
\begin{gathered}
f(\{u,v\}, G) = f(\{u',v'\}, G) \\
\iff \\
\exists \pi \in \Pi_n \text{ s.t. } \{u',v'\} = \pi(\{u,v\}) \text{ and } \mathbf{A}_G = \pi(\mathbf{A}_G).
\end{gathered}
$$

Let $f(\{u,v\}, G) = \text{GNN}(uv, L_{G\cup(u,v)}; \Theta)$ as stated in the theorem statement. Then:

$$
\begin{gathered}
\text{GNN}(uv, L_{G\cup(u,v)}; \Theta) = \text{GNN}(u'v', L_{G\cup(u',v')}, \Theta) \\
\iff \text{Cond. 2} \\
\exists \pi_2 \in \Pi_m \text{ s.t. } u'v' = \pi_2(uv) \qquad (15) \\
\text{and} \\
\mathbf{A}_{L_{G\cup(u',v')}} = \pi_2(\mathbf{A}_{L_{G\cup(u,v)}}). \qquad (16)
\end{gathered}
$$

Equations 15 and 16 indicate that the two line graphs $L_{G\cup(u,v)}$ and $L_{G\cup(u',v')}$ are isomorphic, and, in particular, the isomorphism send node $uv \in V_{L_{G\cup(u,v)}}$ to node $u'v' \in V_{L_{G\cup(u',v')}}$. Clearly, an isomorphism between two graphs always induces an isomorphism between the respective line graphs (Cuypers, 2021). The converse is not always true as the Whitney's theorem states: two connected graphs with isomorphic line graphs are isomorphic unless one is K_3 and the other one is $K_{1,3}$ (Jung, 1966). Following these results, we have:

$$
\begin{gathered}
\exists \pi_2 \in \Pi_m \text{ s.t. } u'v' = \pi_2(uv) \\
\text{and} \\
\mathbf{A}_{L_{G\cup(u',v')}} = \pi_2(\mathbf{A}_{L_{G\cup(u,v)}}) \\
\iff \text{(Harary, 2018) and Cond. 1} \\
\exists \pi \in \Pi_n \text{ s.t. } \mathbf{A}_{G\cup(u,v)} = \pi(\mathbf{A}_{G\cup(u',v')}) \\
\text{with} \\
\pi_2(ab) = \pi(a)\pi(b) \quad \forall ab \in E_{L_{G\cup(u,v)}} \\
\text{and} \\
\pi_2^{-1}(a'b') = \pi^{-1}(a')\pi^{-1}(b') \quad \forall a'b' \in E_{L_{G\cup(u',v')}}.
\end{gathered}
$$

From this and from Equation 15, it follows that:

$$\begin{gathered} u'v' = \pi_2(uv) = \pi(u)\pi(v) \\ \iff \\ u' = \pi(u) \text{ and } v' = \pi(v) \\ \text{or} \\ v' = \pi(u) \text{ and } u' = \pi(v) \\ \iff \\ \{u', v'\} = \pi(\{u, v\}). \end{gathered}$$

Thus, it holds:

$$\begin{gathered} \exists \pi \in \Pi_n \text{ s.t. } \{u', v'\} = \pi(\{u, v\}) \\ \text{and} \\ \mathbf{A}_{G \cup (u,v)} = \pi(\mathbf{A}_{G \cup (u',v')}), \end{gathered}$$

meaning that the graphs $G \cup (u, v)$ and $G \cup (u', v')$ are isomorphic. Additionally, when given two isomorphic graphs, adding or removing corresponding edges will result in graphs that remain isomorphic through the same isomorphism. Therefore, we can conclude:

$$\begin{gathered} \exists \pi \in \Pi_n \text{ s.t. } \{u', v'\} = \pi(\{u, v\}) \\ \text{and} \\ \mathbf{A}_{G \cup (u,v)} = \pi(\mathbf{A}_{G \cup (u',v')}), \\ \iff \\ \exists \pi \in \Pi_n \text{ s.t. } \{u', v'\} = \pi(\{u, v\}) \\ \text{and} \\ \mathbf{A}_G = \pi(\mathbf{A}_G), \end{gathered}$$

which concludes the proof. □

B. Datasets

The number of nodes, links and average node degree are provided for each dataset.

Name	# nodes	# links	Degree
BUP	105	441	8.4
C.ele	297	2148	14.46
USAir	332	2126	12.81
SMG	1024	4916	9.6
EML	1133	5451	9.62
NSC	1461	2742	3.75
YST	2284	6646	5.82
Power	4941	6594	2.669
KHN	3772	12718	6.74
ADV	5155	39285	15.24
GRQ	5241	14484	5.53
LDG	8324	41532	9.98
HPD	8756	32331	7.38
ZWL	6651	54182	16.29

Table 2. Datasets Summary for experiments

C. Results

The AUC results with standard deviation calculated on 5 different seeds are reported in the following Table.

	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	87 ± 03	85 ± 02	92 ± 01	86 ± 01	88 ± 01	98 ± 00	81 ± 01
PR	90 ± 02	89 ± 01	94 ± 01	89 ± 01	89 ± 01	98 ± 00	81 ± 01
SR	85 ± 03	76 ± 02	79 ± 02	78 ± 01	87 ± 01	97 ± 00	74 ± 01
N2V	80 ± 06	80 ± 02	85 ± 01	78 ± 12	83 ± 01	96 ± 01	77 ± 00
NRI	95 ± 01	90 ± 01	96 ± 00	91 ± 00	92 ± 00	$\mathbf{100 \pm 00}$	92 ± 00
GAE	90 ± 02	84 ± 01	92 ± 01	86 ± 01	87 ± 01	99 ± 00	77 ± 00
SEAL	93 ± 01	87 ± 01	95 ± 01	92 ± 00	92 ± 00	$\mathbf{100 \pm 00}$	91 ± 00
LGLP	95 ± 01	90 ± 01	$\mathbf{97 \pm 03}$	93 ± 02	92 ± 03	$\mathbf{100 \pm 00}$	$\mathbf{92 \pm 00}$
SLRGNN	$\mathbf{97 \pm 01}$	$\mathbf{96 \pm 03}$	94 ± 02	$\mathbf{95 \pm 02}$	$\mathbf{98 \pm 01}$	83 ± 10	$\mathbf{92 \pm 03}$
	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	74 ± 01	88 ± 00	94 ± 00	95 ± 00	90 ± 00	93 ± 00	97 ± 00
PR	75 ± 01	92 ± 00	94 ± 00	96 ± 00	91 ± 00	93 ± 00	98 ± 00
SR	71 ± 01	77 ± 01	83 ± 00	89 ± 01	84 ± 00	93 ± 00	95 ± 00
N2V	77 ± 01	83 ± 01	79 ± 01	92 ± 01	81 ± 01	94 ± 00	94 ± 00
NRI	82 ± 01	93 ± 00	95 ± 00	96 ± 00	92 ± 00	97 ± 00	97 ± 00
GAE	70 ± 01	84 ± 01	91 ± 00	94 ± 00	85 ± 00	91 ± 00	95 ± 00
SEAL	84 ± 01	93 ± 00	95 ± 00	$\mathbf{97 \pm 00}$	93 ± 00	$\mathbf{98 \pm 00}$	98 ± 00
LGLP	$\mathbf{85 \pm 01}$	$\mathbf{94 \pm 00}$	96 ± 00	$\mathbf{97 \pm 00}$	94 ± 00	$\mathbf{98 \pm 00}$	98 ± 00
SLRGNN	84 ± 01	$\mathbf{94 \pm 01}$	$\mathbf{98 \pm 00}$	96 ± 01	$\mathbf{97 \pm 01}$	$\mathbf{98 \pm 00}$	$\mathbf{99 \pm 00}$

Table 3. AUC results with std deviation calculated on 5 different seeds.

D. Features Aggregation

For each node $uv \in V_{LG}$, the initial feature $\mathbf{x}^0_{uv}$ is calculated as:

$$\mathbf{x}^0_{uv} = \text{MLP}_\theta(\mathbf{x}^0_v) + \text{MLP}_\theta(\mathbf{x}^0_u), \tag{17}$$

where $\mathbf{x}^0_u$ and $\mathbf{x}^0_v$ are respectively the u-th and v-th rows of the original features matrix $\mathbf{X}^0_G \in \mathbb{R}^{n \times d}$.

The presence of the MLP guarantees the injectivity of the sum; therefore, two nodes $uv, u'v' \in V_{LG}$ in the line graph will have different features if and only if $\{\mathbf{x}^0_u, \mathbf{x}^0_v\} \neq \{\mathbf{x}^0_{u'}, \mathbf{x}^0_{v'}\}$. However, many datasets, including those used in our experimental setting reported in Section 5, do not have initial node features. Therefore, it is necessary to assign initial features to the nodes. The choice of how to assign these initial features is crucial as it is strictly related to the model's expressiveness. For example, it has been shown that using random node features can increase expressiveness (Abboud et al., 2020). However, the use of random features can cause similar nodes to have different representations. Consequently, this undermines the GNN's inductive learning ability to map nodes and links with identical neighborhoods to the same representation, leading to a significant loss of generalization capability. As a result, the model loses its permutation invariance/equivariance, violating the fundamental design principle of GNNs.

In contrast, LT methods (see Section 3.3) use node features based on their distance from the nodes of the links for which they aim to make predictions. This technique has been shown to allow for structural link representation while maintaining inductive learning ability. However, as explained in the Section 3.3, the dependency of these features on the nodes for which predictions are made implies that these features must change each time a different link is to be predicted.

In this work, we have chosen to use topological features as initial features, specifically clustering coefficient, closeness centrality, and betweenness centrality (Wasserman & Faust, 1994). These measures are informative, enable better learning, are independent of the link whose existence is to be predicted, and are identical for similar nodes. This is a direct consequence of the fact that automorphisms, by definition, preserve adjacencies and they also preserve distances between nodes. Indeed, the distance between nodes is defined as the length of the shortest path connecting them. Given two nodes $u, v \in V_G$ with

distance $d(u, v) = k$ and an automorphism $\sigma \in \Sigma_n$, let $P = (u_0 = u, u_1, ..., u_k = v)$ be the shortest path connecting them. Since $\sigma\ P' = (\sigma(u_0), \sigma(u_1), ..., \sigma(u_k))$ will be a path of the same length k between $\sigma(u)$ and $\sigma(v)$. This length is also the minimum: if there were a path of length $k - 1$, applying σ^{-1} to the nodes of this path would yield a path of length $k - 1$ between u and v, leading to a contradiction.

The fact that automorphisms preserve distances ensures that similar nodes have the same centrality measures. Analogous arguments based on the preservation of adjacencies lead to the conclusion that similar nodes also have the same clustering coefficient. This guarantees that using these features as initial features does not break the intrinsic symmetries of the graph's topological structure.