MLGLP: Multi-Scale Line-Graph Link Prediction Based on Graph Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

Scale invariance is a critical property for deep models, enabling them to consistently recognize patterns across varying resolutions or granularities. In this paper, we leverage this principle by proposing a multi-scale link prediction approach based on Graph Neural Networks (GNNs). The proposed method, Multi-Scale Line-Graph Link Prediction (MLGLP), learns graph structure and extracts informative edge representations to address information loss, capture multi-scale structural information, and improve robustness when informative node features are limited or unavailable. This approach utilizes embedding vectors generated by GNNs from enclosing subgraphs. While expanding GNN layers can capture more intricate relations, it often leads to over-smoothing, where node representations become overly similar after many layers. To mitigate this issue, we propose constructing coarse-grained graphs at three distinct scales to uncover complex relations. To apply multi-scale subgraphs in GNNs without using pooling layers that lead to information loss, we convert each subgraph into a line-graph and reformulate the task as a binary node classification problem. The hierarchical structure facilitates exploration across three levels of abstraction, fostering deeper comprehension of the relationships and dependencies inherent within the graph. We perform extensive experiments on several well-known benchmarks and compare the results with state-of-the-art link prediction methods. The experimental results demonstrate the superiority of our proposed model in terms of average precision and area under the curve.

1 Introduction

In our increasingly interconnected world, many problems can be seen as graph-structured data. Graphs are mathematical structures composed of nodes (vertices) and edges (links or connections between nodes). They are widely used to represent complex systems in various domains such as social networks, biological networks, recommendation systems, chemistry, citation networks, and power networks (Cai et al., 2021). Link prediction is a fundamental task in graph analytics with diverse applications across multiple domains such as friend recommendations in social networks or predicting interactions between genes (Zhu et al., 2023). Traditional methods for link prediction use graph structural properties (e.g., common neighbors), often overlook important information such as features associated with each node. In contrast, graph representation-learning techniques such as Graph Neural Networks (GNNs), integrate graph structures with node/edge features, allowing the capture of complex relationships through iterative message passing across graph edges.

GNN-based link prediction methods are categorized into node-based and subgraph-based approaches (Zhang et al., 2020). Node-based techniques like Graph Convolutional Networks (GCN) (Yao et al., 2019), Graph Attention Network (GAT) (Veličković et al., 2017), GraphSAGE (SAGE) Hamilton et al. (2017), and Graph AutoEncoder (GAE) (Kipf & Welling, 2016) incorporate multi-hop graph structures through message passing. They first extract the node embeddings and then predict the possible link between two nodes using a similarity method or a classifier, such as a multi-layer perceptron, to process both nodes' representations and determine the likelihood of a link between the two nodes. In contrast, subgraph-based methods, such as SEAL (Zhang & Chen, 2018), BUDDY (Chamberlain et al., 2022), mLink (Cai & Ji, 2020), LGLP (Cai et al., 2021), and LGCL (Zhang et al., 2023) extract an h-hop enclosing subgraph around the target link, learning a representation specifically tailored to that subgraph. They then use a binary classifier to determine whether

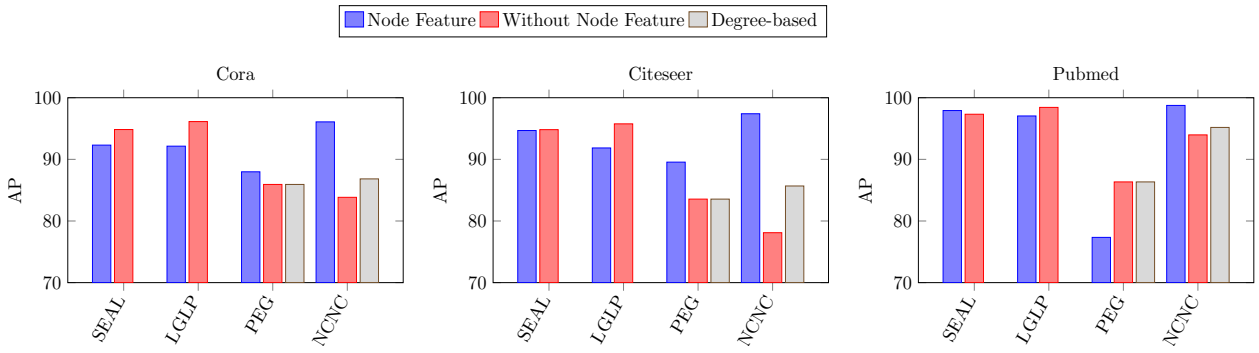


Figure 1: Effect of the presence/absence of node features on different methods (SEAL, LGLP, PEG, NCNC) for AP scores on Cora, Citeseer, and Pubmed

the subgraph indicates the presence or absence of a link (Li et al., 2024). However, methods like SEAL may struggle when node representations closely resemble those of their local neighborhoods, as they may fail to effectively capture information from distant neighbors. This limitation results in subgraphs containing redundant or irrelevant details, which can diminish model performance. If nodes share similar features with nearby nodes, they may miss crucial information from farther nodes, leading to less accurate predictions.

A possible solution is to expand the GNN layers to extract more intricate relations. However, it faces challenges like over-smoothing, where nuanced messaging becomes difficult, resulting in node homogenization, particularly in limited subgraph scenarios. Multi-scale methods such as mLink (Cai & Ji, 2020) provide hierarchical information retrieval capabilities. However, they rely on pooling layers to handle varying node numbers in subgraphs, which may result in information loss.

Recently, approaches like NCNC (Wang et al., 2023), PEG (Wang et al., 2022), and BUDDY have achieved high performance in link prediction tasks. However, they face significant challenges when node features are unavailable, as they rely on message-passing mechanisms based on node features. Consequently, these methods are highly sensitive to node features, and their performance significantly deteriorates in the absence of node features (Wang et al., 2022). In contrast, methods such as SEAL and LGLP, while capable of utilizing node features, often perform better when relying solely on structural information. In fact, incorporating both node and structural features can sometimes lead to a decline in their overall effectiveness (Cai et al., 2021). This suggests that although these methods have the capability to leverage node features, their core strength lies in capturing structural patterns, which play a more crucial role in achieving high performance.

Figure 1 shows the results of an experiment we conducted to examine the impact of the presence or absence of node features on different methods (SEAL, LGLP, PEG, and NCNC). We compare their AP scores on Cora, Citeseer, and Pubmed under three conditions: using real node features, without node features, and using degree-based features instead of real node features. The results demonstrate that while NCNC and PEG suffer a significant performance drop when node features are removed, SEAL and LGLP remain robust, reinforcing their ability to rely primarily on structural information.

Present work. Motivated by this observation, we propose Multi-Scale Line-Graph Link Prediction (MLGLP), a structure-centric framework that enhances link prediction by learning edge representations across multiple structural scales. MLGLP extracts enclosing subgraphs around target node pairs and constructs coarse-grained variants of each subgraph to capture both local neighborhood patterns and broader structural dependencies. Each scale captures information at a different level of granularity. This allows the model to understand relationships between nodes at multiple levels of detail. Each scaled subgraph is then transformed into a line-graph, where edges in the original subgraph become nodes in the derived graph. This transformation reformulates link prediction as a binary node classification problem on line-graphs, enabling the model to learn edge-centric representations while preserving structural information. A GCN is applied to each scaled line-graph, and the resulting target-edge embeddings are concatenated and passed to an MLP classifier for final prediction. Hierarchical structures resulting from different graph scales, enable the anal-

ysis of graphs at different levels of granularity, allowing us to group nodes based on their relationships and capture broader patterns. Unlike SEAL and LGLP, which primarily rely on single-scale enclosing-subgraph representations, MLGLP captures collective interactions across multi-scale subgraphs and clusters, enabling richer structural information extraction and more expressive edge-centric representations for link prediction. The results obtained from experiments on real-world datasets demonstrate the superiority of the proposed method compared to state-of-the-art approaches. The main contributions of this work are summarized as follows:

1. Our method incorporates both local and global graph structures by leveraging subgraphs at different scales, allowing for a more comprehensive understanding of graph relationships, capturing both micro and macro-level graph information.

2. Our approach reformulates the link prediction problem as a node classification task using line-graphs. This approach helps in reducing information loss and simplifying the learning process for the model.

3. The proposed approach captures collective interactions within multi-scale subgraphs, enabling richer edge-centric feature extraction and more expressive structural representations for link prediction.

4. We demonstrate through extensive experiments on benchmark datasets that MLGLP consistently improves link prediction performance over heuristic, embedding-based, node-based GNN, and subgraph-based GNN baselines.

2 Related work

Several approaches have been proposed to address the link prediction task, which can be classified into proximity-based (non-learning) and learning-based approaches. Proximity-based methods rely on statistical properties of nodes/edges within the graph without explicitly learning embedding of nodes or edges. They typically use heuristic or feature-based techniques. Heuristic methods generally employ predefined rules or measures and evaluate link existence by assigning scores derived from the graph structure, utilizing either common neighbors or path information. Examples include Common Neighbour (CN) (Newman, 2001), Adamic Adar (AA) (Adamic & Adar, 2003), Resource Allocation (RA) (Zhou et al., 2009), Significant Influence (SI) (Yang et al., 2018), Shortest Path, and Katz (Katz, 1953). They often assess the existence of a link by assigning a score derived from the graph structure. CN, AA, and RA methods primarily depend on common neighbors, whereas SI, Shortest Path, and Katz methods utilize the graph paths. These methods are widely used due to their simplicity and interoperability. However, each heuristic relies heavily on an underlying assumption regarding the likelihood of two nodes forming a connection, which constrains their efficacy when these assumptions are not met in certain network contexts. Moreover, they rely solely on graph structure and overlook node or edge features, often effective in link prediction tasks. Feature-based methods, on the other hand, use machine learning models trained on a set of features, such as node, edge, or graph attributes. While they incorporate explicit features, they may not fully exploit the underlying graph structure, potentially missing important relational information and dependencies. This can lead to less accurate or insightful models than those that utilize the graph structure directly.

Representation learning methods transform the graph structure into a low-dimensional vector space. They are divided into embedding-based and GNN-based methods. Popular embedding-based methods include Matrix Factorization (MF) (Menon & Elkan, 2011), MLP, Large-scale Information Network Embedding (LINE) (Tang et al., 2015), DeepWalk (Perozzi et al., 2014), and node2vec (Qiu et al., 2018). For instance, DeepWalk (Perozzi et al., 2014) uses the random walk strategy to generate node sequences and applies the Skip-gram model to learn node embeddings. node2vec (Qiu et al., 2018) method extends the DeepWalk by using a biased random walk to better explore neighborhoods. The LINE (Tang et al., 2015) approach captures both first-order and second-order proximities in the graph for better embedding quality. A key limitation of these methods is their inability to leverage node features, relying solely on graph structure. Furthermore, they learn node embeddings with free parameters from the observed network in a transductive manner, meaning they cannot generalize to new nodes or networks not seen during training.

GNN-based methods leverage both network structure and node features. For the link prediction task, GNN-based approaches can be broadly divided into two categories: node-based and subgraph-based methods. In the Node-based category, models like GCN (Yao et al., 2019), Graph Attention Networks (GAT) (Veličković et al., 2017), GraphSAGE (SAGE) (Hamilton et al., 2017), and Graph Autoencoders (GAE) (Kipf & Welling, 2016) represent nodes by leveraging the multi-hop structure of the graph through a message-passing mechanism. GCN utilizes convolutions operations on graphs to aggregate information from neighbors and learn node embedding. GAT assigns varying importance to neighbors using attention mechanisms when aggregating information. GAE uses an Encoder-Decoder framework to learn node embeddings, where the encoder maps nodes to embedding, and the decoder reconstructs the graph structure. GraphSAGE samples and aggregates features from a node's local neighborhood using neural networks.

Recent research studies have focused on subgraph-based methods, which integrate GNNs with enclosing subgraphs extracted from target node pairs, demonstrating remarkable effectiveness. The Weisfeiler-Lehman Neural Machine (WLNM) (Zhang & Chen, 2017) was among the first to apply subgraph-based GNN approaches for link prediction(Zhang & Chen, 2018). Subgraph-based methods integrate additional information, such as subgraph features and common neighbor information to gain a deeper understanding of the relationships between nodes in predicted links. Well-known methods in this category include SEAL (Zhang & Chen, 2018), BUDDY (Chamberlain et al., 2022), mLink (Cai & Ji, 2020), LGLP (Cai et al., 2021), LGCL (Zhang et al., 2023), DE-GNN (Li et al., 2020), and NBFNet (Zhu et al., 2021). The subgraph-based methods, such as SEAL (Zhang & Chen, 2018), extract an h-hop enclosing subgraph around the target link, learning a representation tailored to that subgraph.

3 Preliminaries

In this section, we state the problem of link prediction and provide the formal definitions for the concepts of graphs, h-hop enclosing subgraph, line-graph, Multi-scale graph.

Graph. Let $G = (V, E, \mathcal{A})$ be an undirected graph, where $V = \{1, 2, \ldots, n\}$ is the set of n vertices, and $E \subseteq V \times V$ is the observed edge set, which represents observed relationships and forms a subset of the complete link set E^*. The tensor $\mathcal{A} \in \mathbb{R}^{n \times n \times k}$ encapsulates the features of both nodes and edges. In this representation, diagonal entries $\mathcal{A}_{i,i,:}$ capture the node attributes, while off-diagonal entries $\mathcal{A}_{i,j,:}$ store the edge features. Additionally, we define $\mathbf{A} \in \{0, 1\}^{n \times n}$ as the adjacency matrix, where $\mathbf{A}_{i,j} = 1$ if and only if there exists an edge $(i, j) \in E$, and the matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$, as the matrix of node features where $\mathbf{X}_i = \mathcal{A}_{i,i,:}$ for each node $i \in V$. Without node or edge attributes, we simply set $\mathcal{A} = \mathbf{A}$, treating the adjacency matrix as the feature tensor. Otherwise, the adjacency matrix $\mathbf{A}$ is derived from the first slice of $\mathcal{A}$, such that $\mathbf{A} = \mathcal{A}_{:,:,1}$.

Link Prediction Problem. The link prediction (LP) task is framed as designing a link predictor that operates on an observed subgraph $G \subset G^*$, defined as $\mathrm{LP}(G) = \Pi : V \times V \to \{\mathrm{True}, \mathrm{False}\}$, which classifies the existence of links in the set of candidate edges E_c. The goal of LP is to estimate the likelihood of a potential connection between two nodes, u and v, leveraging both the structural characteristics of the graph and the feature information provided by $\mathcal{A}$. Mathematically, this can be expressed as $p(u, v) = p(u, v|G, \mathbf{X})$, where $\mathbf{X}$ is the node feature matrix derived from the diagonal entries of $\mathcal{A}$. While traditional methods relied on heuristic approaches to estimate $p(u, v)$, contemporary techniques employ a learnable function f, parameterized by Θ, enabling a more flexible and data-driven estimation: $p(u, v) = f(u, v|G, \mathbf{X}, \Theta)$. These advanced methods, often implemented through GNNs, capture complex patterns in the graph structure and feature representations, improving their ability to identify potential links, specifically true missing links. The main objective is to create a vector for each edge in the graph, which captures relevant features or characteristics of the nodes and their relationships. This vector is then fed into a binary classifier that predicts the likelihood of the presence of a given edge, or in other words, predicts whether a target node pair is likely to be connected by a true missing link in the future while avoiding misclassification of false missing links. To train the classifier, two sets of edges are used: positive and negative samples. Positive samples are those edges that currently exist in graph G, while negative samples are a set of pairs randomly sampled from the graph where no edge currently exists.

H-hop enclosing subgraph. The h-hop enclosing subgraph for a node pair (u, v) is the subgraph induced by the set of nodes within h-hops of either u or v, i.e., nodes that are at most h-hops away from either u or v. Specifically, the h-hop enclosing subgraph $G^h_{(u,v)}$ is represented as $G^h_{(u,v)} = (V^h_{(u,v)}, E^h_{(u,v)})$, where $V^h_{(u,v)}$ is the set of nodes within h-hops of u or v, and $E^h_{(u,v)}$ is the set of edges between these nodes in the original graph. The node set $V^h_{(u,v)}$ consists of all nodes $x \in V$ that satisfy $V^h_{(u,v)} = \{x \in V : d(x, u) \leq h \text{ or } d(x, v) \leq h\}$, where $d(x, y)$ represents the shortest-path distance between nodes x and y in the graph $G = (V, E)$. This set includes all nodes that are reachable from either u or v within h-hops. Additionally, the edge set $E^h_{(u,v)}$ includes all edges $(x, y) \in E$ such that both x and y belong to $V^h_{(u,v)}$, Formally expressed as $E^h_{(u,v)} = \{(x, y) \in E : x, y \in V^h_{(u,v)}\}$. These edges represent connections between nodes within the h-hop neighborhood of u and v.

Multi-scaled graph. A multi-scaled graph $SG = (V_s, E_s)$ can be defined through several key steps, starting with the original graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. The first step involves a coarse-graining process, in which a similarity measure $S : V \times V \to \mathbb{R}$ is defined to quantify the similarity between pairs of nodes. Various similarity measures can be utilized; specifically for the link prediction task, the similarity of a group of nodes is determined by their proximity to the target nodes. Next, a partitioning function $P : V \to C$ is established to assign each node $v \in V$ to a cluster $c \in C$ based on the similarity measure, where C represents a set of clusters or hyper-nodes, denoted as $C = \{C_1, C_2, \ldots, C_k\}$. Nodes are then grouped into hyper-nodes according to predefined criteria, such that $\forall u, v \in V$, if $S(u, v) \geq \theta$, then $P(u) = P(v)$, where θ is a threshold for similarity. The vertex set of the scaled graph SG contain the hyper-nodes, defined as $V_s = \{c_i : c_i \in C\}$. The edge set E_s is constructed by connecting hyper-nodes that represent original nodes sharing edges in G, expressed as $E_s = \{(c_i, c_j) : \exists u \in c_i, \exists v \in c_j, (u, v) \in E\}$. Finally, the process can be recursively repeated to create multiple scales SG_1, SG_2, SG_3 for different levels l in the hierarchy, where the l-th scaled graph is defined as $SG_l = (V^{(l)}_s, E^{(l)}_s)$. Each level l represents a different granularity of the original graph, capturing varying patterns of interactions. Transferring a graph to a new scale reduces the complexity of the graph while preserving structural information. Fig 2 shows the process of generating graphs in different scales. The hierarchical structure of a graph enables analyzing

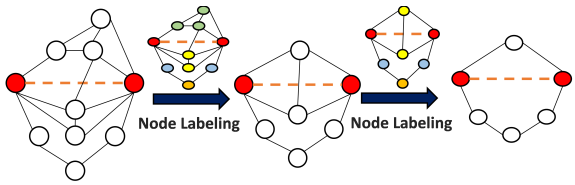


Figure 2: Example of generating graph in different scales. Similar nodes group together.

the graph at varying granularity levels. Rather than treating every individual node independently, we can group nodes based on their relations, allowing us to capture broader patterns and features that might be more insightful for classification tasks. This approach enables extracting richer information by considering not just individual nodes, but also their collective interactions within subgraphs or clusters, leading to a more nuanced understanding of the graph's structure and content for classification purposes. When a node's representation closely mirrors that of its local neighborhood, it struggles to gather information effectively from distant neighbors. Consequently, the surrounding subgraph may contain repetitive or unnecessary details, which can negatively impact the performance of models designed for link prediction. In simpler terms, if a node's features are too similar to those of its nearby nodes, it may miss out on important information from farther away, potentially leading to less accurate predictions in link prediction tasks.

Line-graph. A line-graph $L(G)$ of a graph $G = (V, E)$ is a graph where each node in $L(G)$ corresponds to an edge in G, and two nodes in $L(G)$ are adjacent if and only if their corresponding edges in G share a common vertex. Formally, given a graph $G = (V, E)$, the line-graph $L(G) = (V_L, E_L)$ can be described as follows: the vertex set V_L consists of the edges of the original graph G, meaning $V_L = E$. The edge set E_L is defined as $E_L = \{(e_1, e_2) \in V_L \times V_L : e_1 = (u, w), e_2 = (w, v) \text{ for some } u, v, w \in V\}$. This implies that two nodes $e_1 = (u, w)$ and $e_2 = (w, v)$ in $L(G)$ are connected if the edges e_1 and e_2 in G share a common vertex w. In other words, two nodes in $L(G)$ are connected if their corresponding edges in G share at least one

common vertex. If the G has edges (u, w) and (w, v), then in the L(G), there will be nodes corresponding to these edges. Two nodes in the L(G) are connected if their corresponding edges in the G share a common node. For example, if edges (u, w) and (w, v) are present in the G, then in the L(G), the nodes corresponding to these edges will be connected because they share the common node w. Fig. 3 illustrates the line-graph derived from the original graph.

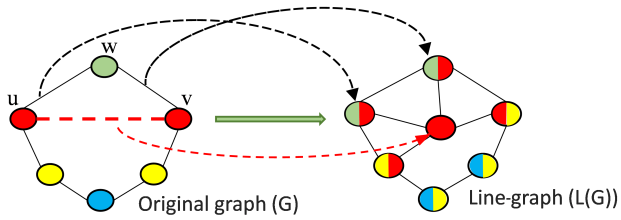


Figure 3: Demonstration of the line-graph conversion process. In the line-graph, each node corresponds to a specific edge in the original graph and is labeled with the identifiers of its two endpoints.

This approach involves representing a link as a new node in a graph and then calculating the representation of this new node to serve as a proxy for the original link. This concept, known as line-graph transformation, treats edges (links) in the original graph as nodes in a derived line-graph, thereby capturing relationships between edges through the derived graph's node representations. In this study, inspired by a method proposed in (Cai et al., 2021), we utilized the line-graph to convert the link prediction task to the node classification by using the line-graph.

4 Proposed methodology

In this section, we introduce the Multi-Scale line-graph Link Prediction (MLGLP) framework, depicted in Fig. 4. The first step is to group nodes with similar characteristics and connections. This consolidation allows us to transform the graph into a new scale, enabling a more efficient representation. Next, we convert the graph into a line-graph, creating three distinct line-graphs at different scales, which provide rich hierarchical structural information. Using GCN, we implement a message-passing mechanism to capture local collaborative patterns among nodes. We then focus on the embedding of target nodes within each line-graph, corresponding to target edges in each multi-scale graph derived from the original graph. By concatenating these embedding vectors, we reframe the problem from binary graph classification into a binary node classification task. To accomplish this transformation, we employ two fully connected layers as a binary classifier. We introduce a novel GNN designed to learn comprehensive relationships and features from subgraphs at different scales. This approach captures a deeper understanding of the underlying structure and dynamics of the data. The following section describes the MLGLP method in detail.

4.1 Subgraph extraction

Detecting the presence of a link between two nodes relies on examining the topology of the graph centered around them. While leveraging global graph structural information often improves the performance, subgraph-based methods typically limit the 2-hop neighbours to balance performance and computational cost. We extract a subgraph containing the target nodes, along with all nodes connected to them within a distance of 1 or 2. For subgraph extraction, we include the target edge even for negative samples, as this step is necessary for later conversion of the subgraph into a line-graph.

4.2 Node aggregation - multi-scale graph transformation

Following existing GNN-based link prediction models, e.g. Zhang & Chen (2018), after extracting the h-hop enclosing subgraph $G^h_{(v_i,v_j)}$ of target pair node (v_i, v_j), we map $G^h_{(v_i,v_j)}$ to three different scales and form coarse-grained graphs SG_1, SG_2, SG_3. We aim to construct these coarse-grained graphs by grouping nodes with similar connections. To do this, we measure the similarity between nodes and assign labels based on

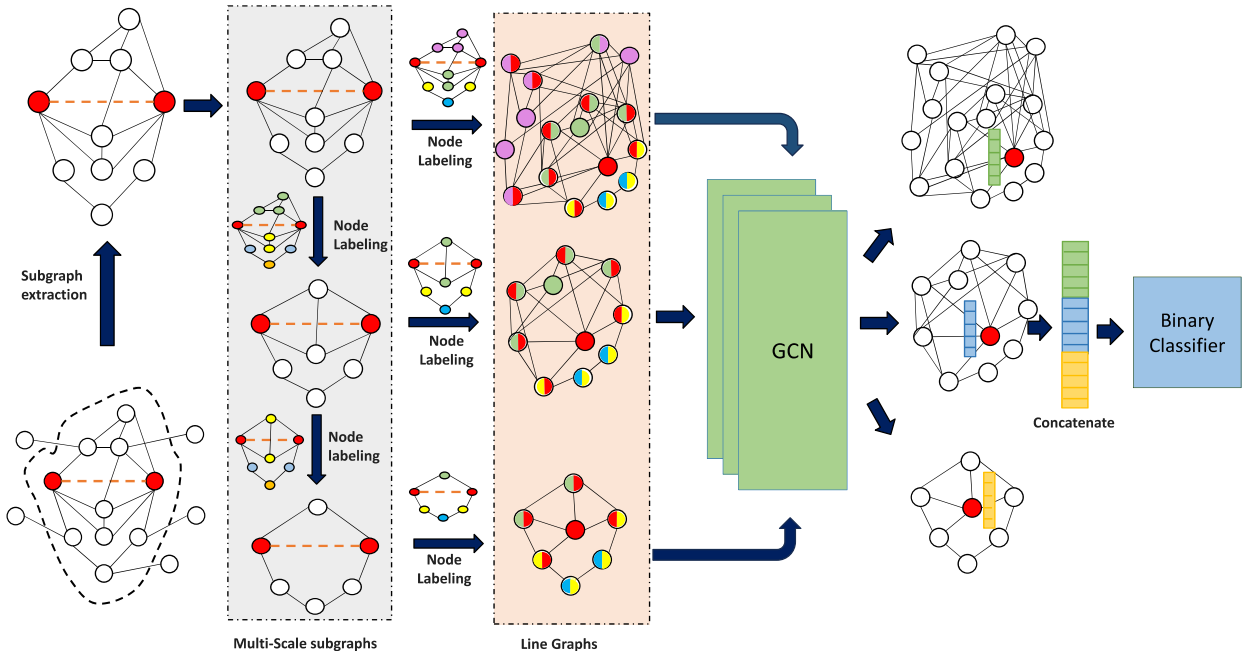


Figure 4: The overall structure of the MLGLP Framework. The process begins by extracting the enclosing subgraph from target pair nodes. We then group nodes with similar characteristics and connections, effectively merging them into single nodes. These graphs are then transformed into a line-graphs, generating three distinct line-graphs at varying scales. Using GCNs, a graph-based message-passing mechanism captures local collaborative patterns among nodes. We implement embeddings of target nodes corresponding to target edges within each multi-scale graph derived from the original graph. These embedding vectors are concatenated, reframing the problem from binary graph classification to binary node classification. A binary classifier, implemented with two fully connected layers, is employed to learn comprehensive relationships and features of subgraphs at different scales.

their proximity to the target nodes. This *labelling process* is crucial for predicting the presence of a link between the target nodes. Target nodes receive label of 1, while others are labeled based on their distance from the target nodes. Neighbour nodes with the same labels ($f(i) = f(j)$) are grouped together to create a hyper node in a subgraph. We assign node labels using equation 1.

$$f_i(u) = 1 + \min(d(u, v_i), d(u, v_j)) + d(u, v_i) + d(u, v_j) \tag{1}$$

where the variables $d(u, v_i)$ and $d(u, v_j)$ represent the shortest distances between a node u and target pair nodes, v_i and v_j, respectively. By aggregating nodes, we form a subgraph, which is then relabeled and further aggregated iteratively to produce three subgraphs representing varying detail levels.

4.3 Line-graph transformation

In this stage, we transform subgraphs SG_1, SG_2, SG_3 into three line-graphs $L(SG_1)$, $L(SG_2)$, $L(SG_3)$, enriching our understanding of the hierarchical structural information within the data. Each node's label in the subgraph is derived from the computation defined by equation 2, and these labels are then encoded as one-hot vectors.

$$h_l(u) = 1 + \min\left(d(u, v_i), d(u, v_j)\right) + \left\lfloor \frac{d(u, v_i)}{2} \right\rfloor \left[\left\lfloor \frac{d(u, v_i)}{2} \right\rfloor + d(u, v_j)\%2 - 1 \right] \tag{2}$$

The *labelling function* must be able to distinguish and identify two specific nodes within the subgraph. It should also be able to assign a label that reflects how important or relevant each node is in relation to the

two target nodes. This involves considering the position and role of each node in the overall structure of the subgraph. Next, each subgraph is transformed into a *line-graph*, where subgraph edges become nodes in the line-graph. Then we assign a label to each node of the line-graph and use them as the initial features for the target link (target node in the line-graph). This process is applied for every edge in the original graph using graph transformation function $T(.)$ in equation 3.

$$T(v_i, v_j) = \text{Concat}(\min(h_l(v_i), h_l(v_j)), \max(h_l(v_i), h_l(v_j))) \tag{3}$$

where $h(v_i)$ and $h(v_j)$, computed using (2), are used as the node representations of v_i and v_j, respectively, after being encoded as one-hot vectors. v_i and v_j denote the two endpoints of an edge. The Concat$(\cdot)$ operation represents the concatenation of the two inputs, v_i and v_j, combining their information into a single feature vector. We merge the two one-hot vectors of nodes into a single, order-invariant vector to create their feature representation. This approach allows the edge attributes to be used as node attributes in the line-graph, thereby preserving the structural information.

4.4 Loss function

We apply GCN to each line-graph to generate the node representations. We focus on the embeddings of target nodes within each line-graph, particularly those corresponding to target edges for pairs of nodes within each multi-scale graph. Therefore, by concatenating these embeddings, the link prediction task is transformed into a binary node classification task. Node embeddings in the line-graph allows us to predict the likelihood of a potential link in the network, framing the task as a binary node classification problem. To achieve this, we employ two fully connected layers as a binary classifier, each with a dimensionality of 32.

Here we use binary cross-entropy as an objective function to treat the link prediction task as a binary classification problem. The training process minimizes the cross-entropy loss across all training links. The loss function is defined as:

$$L = -\sum_{t=1}^{N} (y_t \log(\hat{y}_t) + (1 - y_t) \log(1 - \hat{y}_t)) \tag{4}$$

where N represents the total number of target links used for training, y_t and $\hat{y}_t$ denote the true label value and predicted probability value of the tth sample, respectively, indicating whether the link exists or not. The function $\log(\cdot)$ corresponds to the natural logarithm. The pseudocode of the MLGLP algorithm and its computational complexity analysis are provided in Appendices B and G, respectively.

5 Performance of MLGLP on benchmark datasets

In this section, we evaluated our method (MLGLP) and compared it with 14 methods spanning various categories: heuristics, embeddings, node-based GNNs, and sub-graph-based GNNs. We report Average Precision (AP), Area Under the Curve (AUC), and training loss. Details of the evaluation metrics are provided in Appendix E. Our results show that MLGLP significantly outperforms other methods, demonstrating its effectiveness in link prediction tasks. In our study, we evaluated MLGLP against heuristic approaches including CN (Newman, 2001), AA (Adamic & Adar, 2003), RA (Zhou et al., 2009), PPR , Shortest Path (Liben-Nowell & Kleinberg, 2003), Katz (Katz, 1953), embedding techniques like MLP and MF (Menon & Elkan, 2011), node-based GNN methods including GCN (Yao et al., 2019), GAE (Kipf & Welling, 2016), NCNC (Wang et al., 2023), PEG (Wang et al., 2022), and subgraph-based GNN methods such as SEAL (Zhang & Chen, 2018) and LGLP (Cai et al., 2021). Detailed descriptions of the baselines are provided in Appendix D.

Datasets. In this study, we evaluate the MLGLP method on a set of 8 datasets including Celegans, USAir, Power, NSC, Cora, Citeseer, Pubmed, and Router. Our experiments cover graphs of different magnitudes, encompassing variations in both node count and edge connections. Our goal is to demonstrate the broad applicability of our method across diverse datasets of varying scales, affirming its versatility and efficacy in addressing real-world challenges. The characteristics and statistics of the datasets are presented in Table 6 of Appendix C, with further information provided in the same appendix.

5.1 Settings

In our experiments, we set test-ratio as 0.2, which means that the datasets were randomly divided to have 80% for training and 20% for testing. All learning-based methods are trained for 50 epochs. Also, the batch number is set as 50. Additionally, experiments were conducted ten times, and the results were averaged. The damping factor is set to 0.05 for Katz, and 0.85 for PPR. The learning rate for MF and MLP is set as 0.01. For a fair comparison especially with SEAL and LGLP, we set all parameters as mentioned in the original papers. The output feature dimension for all three graph convolution layers is configured to be 32. For MLGLP the output feature dimension is set to 3×32 due to different scaled subgraphs. All experiments were conducted on AWS EC2 ml.p3.2xlarge instances equipped with 1 NVIDIA V100 GPU, 8 vCPUs, 61 GB of RAM.

5.2 Results

The resulting average AP are presented in Table 1. It is evident that Heuristic-based methods fail to deliver satisfactory performance across all datasets due to their manually designed functions, which are unable to handle diverse cases effectively. Moreover, results show that embedding-based methods exhibit varying performance across different datasets. Additionally, since the methods are applied to plain graphs without node features, the performance of node-based GNNs decreases. Based on the results from Table 1 subgraph-based GNNs achieve the best performance. It indicates that they are capable of automatically learning the link representations from the datasets.

Table 1: AP comparison of MLGLP and all baselines across eight datasets using a test ratio of 0.2.

Methods	Celegans	Power	Router	USAir	NSC	Cora	Citeseer	Pubmed
Heuristics								
CN	78.04%	56.88%	55.02%	93.93%	96.00%	68.66%	64.26%	63.15%
AA	85.45%	57.31%	55.16%	94.90%	96.81%	69.74%	64.33%	63.21%
RA	87.22%	57.41%	55.46%	94.67%	96.31%	70.27%	64.33%	63.19%
PPR	80.28%	76.37%	64.67%	90.38%	97.99%	87.96%	80.02%	85.23%
Shortest Path	67.35%	74.93%	61.42%	75.82%	96.05%	84.15%	78.08%	83.17%
Katz	87.23%	75.36%	63.85%	93.86%	98.22%	85.40%	73.35%	78.81%
Embedding								
MF	84.41%	63.00%	82.55%	94.81%	99.35%	72.95%	81.83%	91.97%
MLP	65.38%	51.96%	61.69%	83.72%	93.21%	57.22%	55.11%	55.37%
GNN based methods								
GCN	79.00%	59.66%	69.35%	92.71%	99.07%	69.95%	71.86%	65.51%
GAE	70.68%	58.16%	55.55%	82.05%	72.57%	86.18%	69.01%	86.04%
SEAL	86.63%	86.71%	97.31%	96.44%	99.65%	94.80%	94.82%	97.32%
LGLP	89.38%	93.71%	99.09%	97.23%	99.79%	96.20%	95.86%	98.28%
PEG	84.06%	72.97%	81.15%	95.28%	97.51%	85.58%	83.32%	85.58%
NCNC	71.39%	69.68%	84.07%	82.54%	98.63%	84.43%	76.84%	94.62%
MLGLP	**93.13%**	**94.96%**	**99.20%**	**98.28%**	**99.89%**	**96.23%**	**96.41%**	**98.29%**

In terms of the AP, our approach achieves the best performance across all datasets, with the LGLP method securing the second-best performance. Specifically, for Celegans, USAir, and Router datasets, the AP values for our method are 93.13, 98.29, and 99.20, respectively, while for the LGLP method, these values are 89.38, 97.23, and 99.09. This demonstrates that our proposed method can effectively learn superior features to represent the target link for prediction in the line-graph space.

The resulting average AUC are presented in Table 2. The data in the table indicates that subgraph-based GNNs excel compared to alternative approaches, showcasing their effectiveness in automatically capturing link representations from the datasets. For AUC, our method gains the best performance except for NSC which gains 99.68.

To further evaluate the robustness of MLGLP under limited supervision, we conduct an additional experiment using only 50% of the training links, with the remaining links used for testing. As reported in Appendix F, MLGLP maintains competitive performance compared with the 80% training setting and consistently outperforms the baselines in AP across all datasets, while achieving the best or near-best AUC on most datasets.

Table 2: AUC comparison of MLGLP and all baselines across eight datasets using a test ratio of 0.2.

Methods	Celegans	Power	Router	USAir	NSC	Cora	Citeseer	Pubmed
Heuristics								
CN	79.93%	56.91%	55.05%	94.42%	96.11%	68.75%	65.00%	63.54%
AA	83.92%	57.29%	55.09%	94.98%	96.78%	69.64%	65.01%	63.54%
RA	86.76%	57.40%	55.33%	93.99%	96.28%	70.33%	65.01%	63.54%
PPR	81.41%	62.57%	45.83%	89.70%	97.99%	82.03%	74.95%	75.84%
Shortest Path	74.69%	62.52%	40.74%	82.48%	96.72%	79.25%	73.38%	75.01%
Katz	87.74%	71.17%	43.89%	92.10%	98.18%	79.45%	73.63%	79.06%
Embedding								
MF	85.34%	62.77%	79.29%	94.64%	99.27%	70.88%	79.45%	92.36%
MLP	64.80%	52.20%	61.66%	83.42%	93.09%	56.27%	55.64%	56.11%
GNN based methods								
GCN	79.45%	58.89%	68.74%	92.02%	98.93%	67.97%	68.06%	62.25%
GAE	70.53%	55.53%	53.05%	82.51%	82.42%	88.17%	66.99%	88.62%
SEAL	89.16%	83.67%	97.36%	96.58%	99.66%	93.40%	86.87%	96.95%
LGLP	90.75%	92.11%	99.05%	98.14%	**99.82%**	95.25%	94.48%	98.22%
PEG	84.77%	68.53%	79.87%	94.79%	96.42%	83.90%	80.15%	84.05%
NCNC	68.76%	63.28%	80.62%	89.17%	98.41%	78.35%	70.27%	94.35%
MLGLP	**90.76%**	**93.84%**	**99.11%**	**98.31%**	99.68%	**95.79%**	**95.72%**	**98.27%**

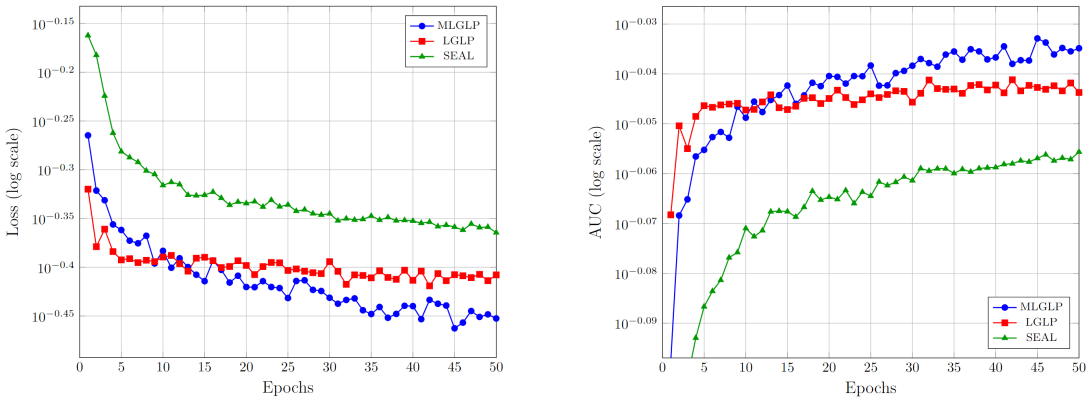
In sub-graph-based approaches, the primary focus is on extracting the enclosing subgraph around target nodes to effectively represent them based on the structure of each subgraph. In subgraph-based GNNs, each subgraph is treated independently as a training sample. Therefore, the presence of test edges in one subgraph does not influence other subgraphs. However, if test edges are masked, it hinders the accurate calculation of the structure of positive and negative samples, which can impact the learning process. To evaluate the impact of masking test edges, we conducted experiments analyzing the performance of sub-graph-based methods, specifically SEAL, LGLP, and MLGLP. The results in Table 3 indicate that masking the test data reduces performance and may introduce inaccuracies in learning patterns for both positive and negative samples. However, our proposed method, MLGLP, consistently demonstrates superior performance across all datasets compared to SEAL and LGLP.

Table 3: AP comparison between masked and unmasked test data using a test ratio of 0.2.

Type	Methods	Celegans	Power	Router	USAir	NSC	Cora	Citeseer	Pubmed
Masked	SEAL	83.12%	77.73%	91.10%	95.33%	99.61%	89.03%	88.79%	94.87%
	LGLP	88.25%	84.66%	93.43%	**96.21%**	99.65%	93.12%	90.62%	**96.77%**
	MLGLP	**90.15%**	**87.01%**	**94.25%**	96.20%	**99.78%**	**93.60%**	**90.96%**	95.93%
Unmasked	SEAL	86.63%	86.71%	97.31%	96.44%	99.65%	94.80%	94.82%	97.32%
	LGLP	89.38%	93.71%	99.09%	97.23%	99.79%	96.20%	95.86%	98.28%
	MLGLP	**93.13%**	**94.96%**	**99.20%**	**98.28%**	**99.89%**	**96.23%**	**96.41%**	**98.29%**

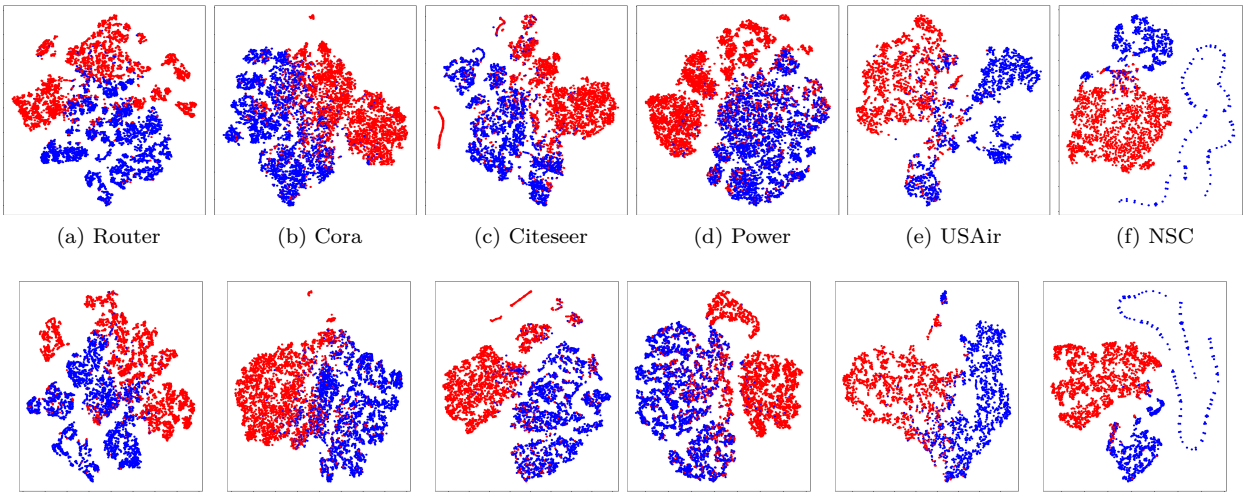
Fig. 5a illustrates the training loss over 50 epochs. It is evident that MLGLP outperforms both LGLP and SEAL and achieves lower loss compared to other methods. This suggests that our proposed approach learns more effective features for representing target links in the line-graph space. Specifically, MLGLP gathers more information from different scales during the training process, enabling it to extract complex features crucial for accurate predictions. In contrast, LGLP performs better than SEAL but does not reduce the loss as effectively as MLGLP over the training period. LGLP converges quickly but struggles to extract complex features effectively during the learning phase, as depicted in the figure. This highlights the superior capability of MLGLP in leveraging training data to enhance feature representation for link prediction tasks. Fig. 5b shows the AUC comparison between LGLP, SEAL, and MLGLP methods for Celegans dataset. The results clearly demonstrate that our proposed model significantly outperforms SEAL and LGLP in terms of achieving a higher AUC.

To highlight the performance of our proposed method, we extracted edge features from the penultimate fully connected layer and applied t-distributed stochastic neighbor embedding (t-SNE) for visualization. Fig. 6 illustrates the results on the Router, Cora, Citeseer, Power, USAir, and NSC datasets under a 0.2 test ratio, with each column corresponding to one dataset. The top row shows the visualizations from our method, while the bottom row shows those from the LGLP baseline. Positive links are depicted in Red and negative links in Blue. The visualization clearly demonstrates that the features learned by our model form well-



(a) Training loss (b) AUC

Figure 5: Training loss and AUC comparison on the Celegans dataset.



(a) Router (b) Cora (c) Citeseer (d) Power (e) USAir (f) NSC

(g) Router (h) Cora (i) Citeseer (j) Power (k) USAir (l) NSC

Figure 6: T-SNE visualization of edge features learned by our method (top row) and the LGLP method (bottom row) across six datasets: Router, Cora, Citeseer, Power, USAir, and NSC. Red indicates positive links; blue indicates negative links.

separated clusters compared to LGLP, making the classification of positive and negative links remarkably straightforward. This showcases the strong discriminative power of our approach.

5.3 Ablation study

Table 4 presents the AP and AUC scores for our proposed MLGLP framework across eight datasets, evaluated using different scales and a combination of all scales. The test ratio for this evaluation was set to 0.2, and we aimed to examine the contribution of each scale within the multi-scale approach. Evaluating individual scales indicates that each scale contributes valuable information, highlighting the significance of capturing different levels of structural patterns within the graph. Depending on the dataset, individual scales can sometimes yield competitive results, particularly on the NSC and Router datasets. When all scales are used together (the "All" method), the model achieves the highest performance on most datasets. For instance, on the USAir dataset, we observe an AP of 98.28% and an AUC of 98.31%. Also the Celegans dataset achieves

Table 4: Impact of different structural scales on MLGLP in terms of average precision (AP) and AUC across eight datasets using a test ratio of 0.2.

Methods	Celegans	Power	Router	USAir	NSC	Cora	Citeseer	Pubmed
				AP				
All(MLGLP)	**93.13%**	**94.96%**	**99.20%**	**98.28%**	**99.89%**	**96.23%**	**96.41%**	**98.29%**
Scale-1 (LGLP)	89.38%	93.71%	99.09%	97.23%	99.79%	96.20%	95.86%	98.28%
Scale-2	88.65%	89.14%	97.07%	95.87%	99.32%	93.43%	93.43%	94.36%
Scale-3	75.67%	89.53%	97.06%	93.32%	99.33%	93.50%	93.50%	98.28%
				AUC				
All(MLGLP)	**90.76%**	**93.84%**	**99.11%**	**98.31%**	99.68%	**95.79%**	**95.72%**	**98.27%**
Scale-1 (LGLP)	90.75%	92.11%	99.05%	98.14%	**99.82**%	95.25%	94.48%	98.22%
Scale-2	88.11%	87.93%	97.29%	95.75%	99.61%	92.35%	92.34%	95.24%
Scale-3	75.33%	87.34%	97.27%	92.81%	99.37%	92.42%	92.42%	98.24%

AP and AUC values of 93.13% and 90.76%, respectively. These results demonstrate that leveraging all scales together consistently provides the best results, confirming the robustness of the multi-scale approach.

6 Conclusion and future research

In this study, we propose a novel approach using GNNs called Multi-Scale Line-Graph Link Prediction (MLGLP). This method aims to effectively learn the graph structure and extract representative features from edges, addressing challenges such as information loss and handling multi-scale information. To facilitate hierarchical learning, our approach involves constructing coarse-grained graphs at three distinct scales, thereby revealing complex relationships within the data. Furthermore, to accommodate GNN learning across multi-scale graphs with varying node and edge sizes, we transform the graphs into line-graph representations. This transformation allows us to learn node embeddings within each subgraph and translates the link prediction task into a node classification problem. Experimental results across eight benchmark datasets show that MLGLP consistently achieves the best AP and the best or near-best AUC compared with heuristic, embedding-based, node-based GNN, and subgraph-based GNN baselines. Future work will extend MLGLP to heterogeneous graphs, where node and edge types can provide additional structural and semantic information.

References

Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2003.

Lei Cai and Shuiwang Ji. A multi-scale approach for graph link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3308–3315, 2020.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:5103–5113, 2021.

Benjamin Paul Chamberlain, Sergei Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Y. Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Jiacheng Li, Harry Shomer, Haitao Mao, Shichang Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

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

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pp. 556–559, 2003.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011*, pp. 437–452, 2011.

Mark E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64: 025102, 2001.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 459–467, 2018.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, 2015.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Hongwei Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022.

Xiyuan Wang, Haoteng Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023.

Yang Yang, Jianpei Zhang, Xiaohong Zhu, and Liang Tian. Link prediction via significant influence. *Physica A: Statistical Mechanics and its Applications*, 492:1523–1530, 2018.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7370–7377, 2019.

Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 575–583, 2017.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Revisiting graph neural networks for link prediction. *arXiv preprint arXiv:2010.16103*, 2020.

Zhen Zhang, Shuyi Sun, Guixiang Ma, and Chuan Zhong. Line graph contrastive learning for link prediction. *Pattern Recognition*, 140:109537, 2023.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009.

Jiong Zhu, Yao Zhou, Vassilis N. Ioannidis, Sijie Qian, Wei Ai, Xiang Song, and Danai Koutra. Spottarget: Rethinking the effect of target edges for link prediction in graph neural networks. *arXiv preprint arXiv:2306.00899*, 2023.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29476–29490, 2021.

A Notations

This section provides an overview of the symbols and notations utilized in this paper, with a detailed summary provided in Table 5.

Table 5: Summary of notations used in the paper

Notations	Definitions or Descriptions
$G = (V, E, X)$	Graph with node set V, edge set E, and node features X
n	Number of nodes, $n = \|V\|$
m	Number of edges, $m = \|E\|$
E^*	Complete set of possible edges between nodes in V
$\mathcal{A}$	Feature tensor, $\mathcal{A} \in \mathbb{R}^{n\times n\times k}$, capturing both node and edge features
$\mathcal{A}_{i,i,:}$	Node features for node i
$\mathcal{A}_{i,j,:}$	Edge features for edge (i, j)
$\mathbf{A}$	Adjacency matrix, $\mathbf{A} \in \{0, 1\}^{n\times n}$
$\mathbf{A}_{i,j}$	Adjacency matrix entry, 1 if edge (i, j) exists, otherwise 0
$\mathbf{X}$	Node feature matrix, $\mathbf{X} \in \mathbb{R}^{n\times k}$
$\mathbf{X}_i$	Feature vector for node i, $\mathbf{X}_i = \mathcal{A}_{i,i,:}$
G^*	Complete graph with all possible edges E^* between nodes in V
E_c	Set of candidate edges for link prediction
$p(u, v)$	Probability of a link between nodes u and v
$f(u, v\|G, \mathbf{X}, \Theta)$	Learnable function to estimate $p(u, v)$, parameterized by Θ
Θ	Parameters of the learnable function f
h	Maximum number of hops in the h-hop enclosing subgraph
$G^h_{(u,v)}$	h-hop enclosing subgraph for node pair (u, v)
$V^h_{(u,v)}$	Node set within h-hops of either u or v
$E^h_{(u,v)}$	Edge set within h-hops of either u or v
$d(x, y)$	Shortest-path distance between nodes x and y
$SG = (V_s, E_s)$	Multi-scaled graph with vertex set V_s and edge set E_s
$S(u, v)$	Similarity measure between nodes u and v
$P(v)$	Partitioning function assigning node v to a cluster
C	Set of clusters or hyper-nodes
θ	Threshold for node similarity
$SG_l = (V_s^{(l)}, E_s^{(l)})$	Scaled graph at level l in a hierarchical structure
$L(G) = (V_L, E_L)$	Line-graph of graph G
V_L	Node set of the line-graph
E_L	Edge set of the line-graph
L	Loss function
N	Total number of target links used for training
y_t	True label value for the tth sample
$\hat{y}_t$	Predicted probability value for the tth sample
$f_i(u)$	Label of node u
$h(u)$	Node representation of node u
$T(v_i, v_j)$	Function for concatenating features of nodes v_i and v_j

True Missing Links are edges that should exist in the graph but are not currently observed. These edges are part of the complete graph's true edge set E^* but are not included in the observed edge set E. Formally, a true missing link (u, v) satisfies:

$$\{(u, v) \in E^* \text{ and } (u, v) \notin E\}$$

where E is the set of obsserved edges and E^* is the set of all true edges in the complete graph. For example if E^* includes an edge (u, v) but (u, v) is not in E, then (u, v) is a true missing link.

False Missing Links are pairs of nodes that are incorrectly considered as potential links but are not part of the true edge set E^*. These are often pairs where a link is not present in both the complete graph and the observed graph. Formally, a false missing link (u, v) satisfies:

$$\{(u, v) \mid u, v \in V \text{ and } (u, v) \notin E^* \text{ and } (u, v) \notin E\}$$

for example if E^* does not include an edge (u, v) and (u, v) is also not in E, this edge can be mistakenly considered a potential link (false positive) by some models.

Positive Samples are defined as edges that exist in both the observed edge set E and the complete graph E^*. In other words, these are edges that are part of the true edge set and are observed. Formally, these are edges in $E \cap E^*$.

Negative Samples are defined as edges that do not exist in either the set of true edges E^* or the set of observed edges E. They can be considered as false positives, i.e., pairs of nodes that should not be connected. These correspond to pairs of nodes that are not connected by an edge in the graph. Since identifying true negative samples, edges that should never exist, is often challenging due to the lack of complete knowledge about the graph or underlying data distribution, a common strategy is randomly selecting negative samples from the set of nonexistent edges. Thus, the negative samples are selected randomly from edges that are not in E, acknowledging that while these may not be guaranteed to be true negatives, they serve as a practical and sufficient set of negative examples for training the model.

Candidate Set (E_c) is constructed to include both types of samples to train and evaluate the link prediction model effectively. The goal is to differentiate between these two types of samples and correctly classify which candidate edges should be present in the graph. These candidates help train and test the model to accurately predict the presence of true missing links while avoiding false positives.

B Algorithm

In this section, we present the pseudo-code of the MLGLP framework to enhance clarity and understanding.

Algorithm 1 MLGLP Algorithm.

Data: Target link $(\mathbf{v_i}, \mathbf{v_j})$, graph $\mathbf{G}$
Result: Predicting the existence or nonexistence of the target link
Input: $\mathbf{h = 2}$

1. Extract h-hop enclosing subgraph of target node pair $G^h_{(v_i, v_j)}$
2. Compute node labeling by equation (3)
3. Transfer $G^h_{(v_i, v_j)}$ to three multi-scale subgraphs SG_1, SG_2, SG_3
4. Convert multi-scale subgraphs SG_1, SG_2, SG_3 to line graphs LSG_1, LSG_2, LSG_3
5. Initialize node embedding
6. Compute the node embedding using GCN
7. Concatenate embedding vectors of target nodes in LSG_1, LSG_2, LSG_3
8. Predict link existence (v_i, v_j) with a binary classifier

C Datasets

A brief overview of the benchmark datasets utilized in this study is as follows. These datasets are used for evaluating models in various types of graphs, including social networks, citation networks, and biological networks.

- Cora: This citation graph includes 2708 scientific publications and 5278 links, with a dictionary of 1433 unique words derived from the papers.
- Citeseer: This dataset features 3312 scientific publications and 4552 links, accompanied by a dictionary of 3703 unique words from the publication texts.

- Pubmed: This citation graph includes 19,717 scientific publications and 44,324 citation links, with a dictionary of 500 unique words derived from the papers.
- Router: This dataset represents a router-level Internet graph with 5022 nodes and 6258 edges, modeling connections between routers in the network.
- USAir: This dataset represents a graph of US airlines, containing 332 nodes and 2126 edges.
- NSC: This dataset illustrates the collaboration relationships of network science researchers, containing 1589 nodes and 2742 edges.
- Celegans: This dataset contains the biological neural network of *C. elegans*, consisting of 297 nodes and 2148 edges.
- Power: This dataset illustrates the topology of the Western States Power Grid of the United States, containing 4941 nodes and 6594 edges.

Table 6: Summary statistics of the datasets used in the study

Statistic	**Router**	**Cora**	**Citeseer**	**Pubmed**	**USAir**	**NSC**	**Celegans**	**Power**
#Nodes	5022	2708	3312	19,717	332	1461	297	4941
#Edges	6258	5429	4552	44,324	2126	2742	2148	6594
#Features	NA	1432	3703	500	NA	NA	NA	NA

D Baseline methods

This section provides detailed descriptions of the baseline methods utilized in this paper. We compare our proposed MLGLP with 14 methods spanning various categories: heuristics, embeddings, node-based GNNs, and sub-graph-based GNNs. In our study, we evaluated our proposed method against heuristic approaches such as CN, AA, PPR, Shortest Path, and, Katz, embedding techniques like MLP and MF, node-based GNN methods including GAE, GCN, NCNC, PEG, and finally subgraph-based GNN methods such as SEAL and LGLP. Table 7 shows Details of heuristic-based methods utilized in this paper.

Table 7: Details of heuristic-based methods for link prediction utilized in this paper

Name	**Formula**	**Order**
Common Neighbors (CN)	$\lvert\Gamma(x) \cap \Gamma(y)\rvert$	first
Adamic-Adar(AA)	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log \lvert\Gamma(z)\rvert}$	second
Resource Allocation (RA)	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\lvert\Gamma(z)\rvert}$	second
PageRank(PPR)	$[\pi_x]_y + [\pi_y]_x$	high
Shortest Path	$\frac{1}{length(shortestpath(x,y)}$	high
Katz	$\sum_{l=1}^{\infty} \beta^l \lvert \text{walks}^{\langle l \rangle}(x,y)\rvert$	high

E Evaluation metrics

This section provides detailed descriptions of evaluation metrics including AUC, AP.

a) AUC: The AUC is computed as the number of successful predictions divided by the total number of comparisons. Successful predictions can be determined based on scores for each node pair using predefined heuristics (e.g., common neighbors) or probabilities for each node pair using the GNN model. Compute the AUC using the formula:

$$AUC = \frac{n' + 0.5 \times n''}{n} \tag{5}$$

where n is the total number of predictions, n' is the number of successful predictions (i.e., the number of times for each positive-negative pair the positive sample has a higher score or probability than the negative sample), and n'' is the number of times that scores are the same.

b) AP: Average Precision measures the precision of a model at various threshold levels, capturing the ability to rank positive links higher than negative ones. It is formulated as:

$$AP = \sum_{k=1}^{n} P(k) \cdot \Delta r(k) \tag{6}$$

Where n is the total number of positive and negative links, $P(k)$ is the precision at rank k, and $\Delta r(k)$ is the change in recall at rank k.

F Robustness evaluation with reduced training data

To demonstrate the robustness of our proposed approach with reduced training data, we conducted experiments on all datasets using only 50 percent of the training links. The remaining links were used as test data. The outcomes, including AUC and AP values, are presented in Tables 8 and 9 The results consistently show that our method outperforms all baseline methods in terms of AP across all datasets and AUC in the majority of datasets. Remarkably, our method maintains strong performance even with only 50 percent of the training links, achieving AUC and AP values comparable to those obtained with 80 percent of the training links. For example, AP for Celegans, Power, Router, USAir, NSC, and Cora just reduced by 3.45%, 1.69%, 0.58%, 2.67%, 0.06%, and 0.90%, respectively.

Table 8: AUC comparison of MLGLP and all baseline methods across eight datasets using a test ratio of 0.5.

Methods	Celegans	Power	Router	USAir	NSC	Cora	Citeseer	Pubmed
Heuristics								
CN	70.87%	53.14%	53.39%	87.82%	90.38%	59.69%	57.71%	57.41%
AA	72.52%	53.05%	52.57%	88.11%	92.41%	59.79%	57.63%	57.39%
RA	72.78%	53.28%	52.86%	87.77%	92.60%	59.12%	58.41%	57.28%
PPR	79.51%	57.73%	54.32%	85.42%	95.56%	68.39%	67.29	73.06%
Shortest Path	72.11%	57.45%	54.44%	82.66%	94.86%	67.10%	66.53%	72.80%
Katz	80.09%	58.28%	54.41%	89.71%	96.17%	68.95%	68.01%	50.00%
Embedding								
MF	71.99%	54.70%	77.47%	90.87%	98.64%	63.96%	62.28%	87.18%
MLP	62.81%	51.19%	60.32%	80.32%	89.31%	55.88%	53.19%	54.57%
GNN based methods								
GCN	73.80%	52.44%	59.49%	90.08%	98.09%	60.14%	60.40%	65.72%
GAE	62.20%	53.88%	49.65%	70.70%	76.57%	83.44%	62.85%	83.75%
SEAL	88.19%	82.21%	97.12%	96.32%	99.64%	93.42%	92.73%	96.86%
LGLP	90.94%	91.78%	98.98%	97.34%	99.77%	95.22%	94.35%	98.38%
PEG	73.42%	57.67%	61.51%	88.16%	84.39%	67.48%	65.25%	70.41%
NCNC	81.16%	61.20%	80.18%	81.42%	87.70%	67.67%	60.73%	87.71%
MLGLP	**91.52%**	**93.28%**	**98.62%**	**97.22%**	**99.83%**	**95.33%**	**94.99%**	**98.40%**

G Time complexity analysis

For extracting each graph $G^h_{(v_i,v_j)}$ with n nodes and m edges where (v_i, v_j) is the target pair of nodes, total time complexity for calculating distances for both target nodes is $\mathcal{O}(n + m)$ per scale. Constructing a *line-graph* has a time complexity of $\mathcal{O}(m^2)$, where m is the number of edges in the original graph (since each edge in the original graph becomes a node in the *line-graph*). Time Complexity for GCN operation

Table 9: AP comparison of MLGLP and all baseline methods across eight datasets using a test ratio of 0.5.

Methods	**Celegans**	**Power**	**Router**	**USAir**	**NSC**	**Cora**	**Citeseer**	**Pubmed**
				Heuristics				
CN	68.23%	53.12%	53.38%	87.47%	90.34%	59.57%	57.68%	57.38%
AA	70.72%	53.05%	52.55%	88.46%	92.44%	59.91%	57.64%	57.39%
RA	72.24%	53.27%	52.80%	88.51%	92.61%	59.18%	58.41%	57.28%
PPR	78.25%	57.68%	60.89%	84.90%	95.58%	75.41%	69.07%	80.17%
Shortest Path	66.00%	57.65%	60.64%	77.60%	94.47%	73.46%	68.70%	79.10%
Katz	79.90%	58.42%	61.07%	92.03%	96.25%	75.71%	69.49%	50.00%
				Embedding				
MF	82.25%	54.48%	81.38%	91.19%	98.86%	66.42%	65.66%	87.73%
MLP	64.28%	51.19%	60.06%	81.35%	89.67%	56.84%	54.26%	54.10%
				GNN based methods				
GCN	73.06%	53.08%	62.04%	90.56%	98.33%	62.96%	64.25%	68.18%
GAE	61.54%	53.96%	49.80%	68.44%	66.95%	82.48%	65.96%	82.64%
SEAL	86.58%	85.57%	97.06%	95.96%	99.51%	94.73%	94.43%	97.24%
LGLP	89.63%	93.87%	98.86%	98.28%	99.77%	95.98%	95.69%	98.40%
PEG	74.49%	57.41%	58.13%	90.15%	89.31%	67.98%	66.52%	72.28%
NCNC	79.93%	68.35%	83.47%	74.51%	90.13%	74.16%	66.14%	88.29%
MLGLP	**93.43%**	**95.04%**	**99.06%**	**98.54%**	**99.92%**	**96.42%**	**95.79%**	**98.41%**

depends on the number of nodes and edges in the line-graph. For a graph with $|V|$ nodes and $|E|$ edges, the complexity is $\mathcal{O}(|V| + |E|)$. Since we are dealing with line-graphs, this translates to $\mathcal{O}(n^2 + n^2) = \mathcal{O}(n^2)$ for each scale. For three scales, it's $\mathcal{O}(3n^2) = \mathcal{O}(n^2)$.