Applied Intelligence (2025) 55:489
https://doi.org/10.1007/s10489-025-06394-5

Multi-scale contrastive learning via aggregated subgraph for link prediction

Yabing Yao[1] · **Pingxia Guo**[1] · **Zhiheng Mao**[1] · **Ziyu Ti**[1] · **Yangyang He**[2] · **Fuzhong Nian**[1] · **Ruisheng Zhang**[3] · **Ning Ma**[4]

Accepted: 17 February 2025

Abstract

Link prediction seeks to uncover potential or future connections within a network using structural or attribute information. Recently, Graph Neural Network (GNN)-based methods have attracted considerable attention for their effectiveness in link prediction. However, most GNN-based approaches focus solely on single-scale input graphs, which limits their ability to comprehensively capture network structure information. In this paper, multi-scale subgraphs are introduced as input graphs to obtain complementary network structures from different perspectives. Simultaneously, to obtain embedding vectors with better representational capacity, contrastive loss from self-supervised learning is incorporated for link prediction. Specifically, **M**ulti-scale **C**ontrastive learning framework based on **A**ggregated **S**ubgraph (MCAS) is proposed for predicting missing links. Firstly, we construct enclosing subgraph by extracting neighbors of target nodes. By applying aggregation operation to these subgraphs, different granularities of multi-scale subgraphs are obtained. Secondly, encoders are used to learn information from multiple scales of subgraphs separately. Next, contrastive learning is employed to achieve information balance among the multi-scale subgraphs. Finally, the minimization of the loss allows us to improve the model's robustness. Empirical evidence indicates that our approach excels state-of-the-art methods on nine datasets, including biological and citation networks. All source code is publicly available at: https://github.com/yabingyao/MCAS4LinkPrediction.

Keywords Complex networks · Link prediction · Multi-scale subgraph · Contrastive learning

1 Introduction

Link prediction, an important data mining technique [1], aims to infer unobserved connections within networks or predict potential future interactions based on known structures or related information [2]. This technique has wide-ranging applications across various fields. For example, it is used for friends recommendations in social networks [3], predicting protein-protein interactions in biological networks [4], and recommending routes in transportation networks [5]. Moreover, in the realm of security, link prediction helps identify hidden connections between terrorists and organizations [6]. Accurate predictions in these diverse domains provide valuable insights for decision-makers.

In previous research on link prediction, Graph Neural Networks (GNNs) have been widely utilized due to their strong expressive power [7, 8]. GNNs learn node representations through iterative parameter optimization, enabling the prediction of link relationships between nodes [9, 10]. However, traditional GNN-based methods typically process the entire graph, resulting in significantly increased computational costs as the network scale grows. To address this issue, GNN methods based on enclosing subgraphs have been proposed, such as SEAL [11] and Scaled [12]. These approaches extract local subgraphs around target nodes and compute distance labels as features to learn node representations. While effective in capturing local structural information, these methods rely heavily on similarly labeled nodes and single-scale subgraphs, which limits their expressive capability.

From a multi-scale perspective, multi-scale structures have become a vital approach to addressing the link prediction problem and have garnered significant attention in recent years [13–16]. Among them, Cai et al. [16] propose a multi-scale node aggregation method, which integrates redundant nodes into multi-scale representations while preserving the

Extended author information available on the last page of the article structural information between target nodes. However, this method may result in the loss of certain node information during the aggregation process.

Therefore, achieving a balance between information retention and the construction of multi-scale structures remains a critical challenge, which is the central objective of the methodology introduced herein. To address this challenge, we introduce a novel approach, Multi-scale Contrastive learning based on Aggregated Subgraphs (MCAS), designed to preserve and effectively leverage information across multiple granularities within networks. MCAS employs a distance-labeling algorithm to aggregate enclosing subgraphs centered around target links into multi-scale graphs. By comparing the original enclosing subgraph with its aggregated counterparts at different scales, MCAS effectively maximizes information retention and minimizes information loss. This contrastive learning framework facilitates the model's ability to learn high-quality representations, thereby enhancing the performance of link prediction task. The key contributions of this work are summarized as follows:

- To capture richer hierarchical information within networks, we introduce MCAS, a novel approach that employs a multi-scale contrastive learning framework based on aggregated subgraphs.
- To enhance the quality of node representations for link prediction and minimize information loss in multi-scale graphs, we propose contrastive learning between the original enclosing subgraph and its aggregated counterparts.
- We conduct a comprehensive performance comparison against baseline methods across nine datasets from diverse domains. Experimental results demonstrate the superior robustness and efficacy of our proposed model.

The paper is organized as follows: Sect. 2 provides an overview of related work and the technologies employed. Sect. 3 defines the link prediction task. Section 4 presents a detailed overview of the proposed MCAS method. Section 5 outlines the experimental setup, including datasets and baseline methods, and discusses the experimental results. Finally, this paper concludes in Sect. 6 with a summary of the proposed approach.

2 Related work

This section explores into the relevant literature and the techniques utilized in our proposed method. It is structure into three parts. The first part provides a summary of link prediction methods, including their classifications, advantages, and limitations. The second part explores the application of multi-scale methods for tackling graph-based tasks. Finally, this third part offers a brief overview of how the multi-scale contrastive learning approach is used in various downstream tasks, including link prediction.

2.1 Link Prediction Methods

Heuristic-based methods Heuristic-based methods predict link existence based on the similarity scores between two target nodes and rely on various underlying assumptions. Local similarity methods, such as Common Neighbors (CN) [17], Adamic-Adar (AA) [18], and Resource Allocation (RA) [19], focus on the immediate neighborhood of target nodes. These methods estimate the likelihood of a link by extracting and counting first-order and second-order neighbors based on the similarity of their local attributes. To capture more comprehensive information, high-order heuristic methods like KATZ [20] and SimRank [21], utilize information from the entire graph to compute the similarity scores between two nodes. Although these structure heuristic-based methods are relatively simple and interpretable, their limited generalization capability makes them unsuitable for diverse datasets.

Representation learning-based methods Deep learning has paved the way for evaluating target nodes similarity through vector embeddings learned via shallow representation methods. These methods integrate information from the neighbors of target nodes into their embeddings, thereby improving performance in downstream tasks. Perozzi et al. [22] introduce DeepWalk, which learns vector representations of nodes by utilizing their co-occurrence information within the graph. Building on DeepWalk, Cai et al. [23] incorporate clustering method into the link prediction process, demonstrating that nodes with higher structural and attribute similarity are more likely to connect. Tang et al. [24] introduce LINE, a network embedding model well-suited for large-scale information networks, achieving robust performance in link prediction task. Grover et al. [25] develop Node2vec, which enhances traditional random walk and word embedding approaches by combining depth-first and breadth-first search strategies and considering second-order node similarity. Beyond shallow representation learning, matrix factorization-based methods also show promise. Nasiri et al. [26] introduce a robust method RGNMF-AN, for nonnegative matrix factorization on attributed networks. This method can accurately predict links even in noisy or incomplete networks by modeling both network topology and node attributes. To address the challenge of network sparsity in link prediction, Yao et al. [27] propose EG-DNMF and EG-FDNMF, which are deep nonnegative matrix factorization models integrated with an edge generator to enhance link prediction accuracy.

Although representation learning-based methods outperform heuristic-based methods in link prediction accuracy, they struggle with evaluating the quality of learned embeddings due to the absence of supervisory signals during training phase. Furthermore, these methods often struggle to capture the rich network topology and detailed node attribute information, which can limit their effectiveness when applied to large-scale and complex graphs [28].

GNN-based methods Neural networks have driven a revolution in deep learning, achieving remarkable advancements in recent years. This progress has significantly propelled the development of graph learning. GNNs [7] typically leverage encoder-decoder architectures for end-to-end modeling. The encoder maps graph nodes into low-dimensional vector representations, while the decoder predicts link probabilities between nodes based on these embeddings. Graph Convolutional Networks (GCNs) [29] have been widely utilized for semi-supervised learning on graph data, learning hidden layer representations of node features and applying them to tasks like link prediction. GraphSAGE [30] is an inductive learning framework that leverages neighborhood aggregation to generate node embeddings, enabling effective representation of unseen nodes. This approach proves valuable in tasks such as link prediction. Building upon GraphSAGE, Jiawei et al. [31] introduce the GraphSAGE++ framework, which enhances the aggregation process by combining double aggregation with weighted concatenation, significantly improving the model's ability to recognize and preserve structural information. Focusing on capturing structural information for link prediction, Shi et al. [32] propose a framework that utilizes GNNs to model neighborhood graph structures. Furthermore, they also propose a binary structure transformer to capture the structural relationships among target nodes.

Recent approaches have shown promising performance by extracting local enclosing subgraph structures (e.g. SEAL [11], Scaled [12], and PS2 [33]). However, these methods always adopt single-scale input graphs and neglect the multi-scale structural features in networks, which limits the representation quality of GNNs to some extent.

2.2 Multi-scale representation learning

Multi-scale representation learning is a training paradigm designed to enhance model robustness by incorporating data of varying scales. This approach has gained significant traction in computer vision, where it is widely adopted to extract richer and more diverse information from multiple perspectives [34, 35]. For instance, in image classification, Shi et al. [36] leverage multi-scale convolutional neural networks to capture scale-specific object information, improving classification accuracy. Similarly, in object detection, Pang et al. [37] propose self-interaction modules that enable networks to adaptively retrieve multi-scale information, effectively addressing scale variations and enhancing detection performance.

Inspired by these approaches, researchers have proposed transforming image pixels into network nodes and constructing scale graphs with varying granularities from network data, which are then processed through multi-scale learning frameworks. This framework aims to obtain informative vector representations of the graph. Benedek et al. [38] introduce a multi-scale attribute node embedding method that encodes features by leveraging information from different node proximities, making it adaptable to various downstream applications. Zhang et al. [15] propose a method to model the intrinsic correlation between the evolutionary dynamics at different structural scales for dynamic link prediction. Cai et al. [16] utilize a node aggregation operation to generate multi-scale input graphs, framing link prediction as a binary classification problem addressed through shared graph neural networks.

2.3 Multi-scale contrastive learning

Graph Contrastive Learning (GCL) is a leading self-supervised learning approach in the GNN domain, developed to address the scarcity of labeled graph data. GCL enhances the robustness of node representations by contrasting augmented views of the same graph. Specifically, it generates positive and negative sample pairs through graph augmentation techniques, encodes these pairs into low-dimensional representations using GNN, and employs a contrastive loss function to maximize the similarity of positive pairs while minimizing that of negative pairs. Multi-scale contrastive learning has seen success in object detection [39, 40], drug target prediction, and protein prediction [41]. Zhang et al. [42] propose line graph contrastive learning for link prediction using subgraph contrast. In the task of node classification, Zhang et al. [43] introduce a multi-scale contrastive training scheme that enriches node representations by integrating information across three granular levels: subgraph, graph, and node-level context.

Motivated by the success of these approaches, we propose a multi-scale contrastive method for node aggregation. This approach capitalizes on the richer complementary information inherent in multi-scale graphs, while mitigating the potential loss of node-level details during graph scaling. Unlike previous methods that focus on graph-level contrastive learning, our approach operates at a finer granularity by emphasizing contrasts at the node level.

3 Problem formulation

Consider an unweighted and undirected network G, which can be represented as $G = (V, E)$. Here, V is a set of nodes containing N elements, and $E \subseteq V \times V$ represents the set of observed links. The set of all possible links of network G is denoted by U, and the set of unobserved links is $(U - E)$. The network can also be described by its adjacency matrix $A \in \mathbb{R}^{N \times N}$, which is a real-valued matrix of size $N \times N$. Additionally, the feature matrix is $X \in \mathbb{R}^{N \times F}$, where F denotes the dimensionality of the node features. For the task of link prediction, the adjacency matrix of the network is specifically defined as follows:

$$A(i, j) = \begin{cases} 1 & \text{if } e_{ij} \in \text{E}, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

the adjacency matrix A represents the network's connectivity. Specifically, $A(i, j)$ equals 1 if there is a link between nodes v_i and v_j, and $A(i, j) = 0$ otherwise. Link prediction aims to utilize this information, along with the overall structure of the graph, to forecast potential connections between nodes.

Since the future potential links of the network G are unknown, we randomly select $p\%$ of the existing links in E as the observed links E^T for training and the remaining $1 - p\%$ as the test links E^P for evaluating the model's performance. These sets are mutually exclusive ($E^T \cap E^P = \varnothing$) and together encompass all observed links ($E^T \cup E^P = E$). In the training process, the observed links E^T are used to construct the input graph for feature extraction, ensuring that E^P remains unseen throughout the feature computation process.

To enhance computational efficiency during training, we do not use all links in E^T as positive samples. Instead, we randomly select a subset of links from E^T as positive samples and an equal number of non-existing links from $(U - E^T)$ as negative samples. These positive and negative samples form the input data for training the binary classifier. During training, the model learns node embeddings and graph representations from the training graph. For the testing phase, these learned embeddings are used to predict the presence or absence of links in the test set E^P based on the extracted features. After obtaining the predictions, the model's outputs are compared with the true links in the unobserved link set to compute the loss, which is then used to optimize the model's parameters and improve its performance.

4 The proposed method

GNNs excel at capturing information within graph structures and encoding it into expressive vector representations. To further enhance these representations, we introduce a multi-scale contrastive learning framework. By integrating multi-scale information, the model effectively captures structural features at different hierarchical levels within the graph. Figure 1 outlines the four key components of the MCAS framework: (a) Extracting enclosing subgraph; (b) Aggregating multi-scale subgraphs; (c) Inputting scale graphs; (d)

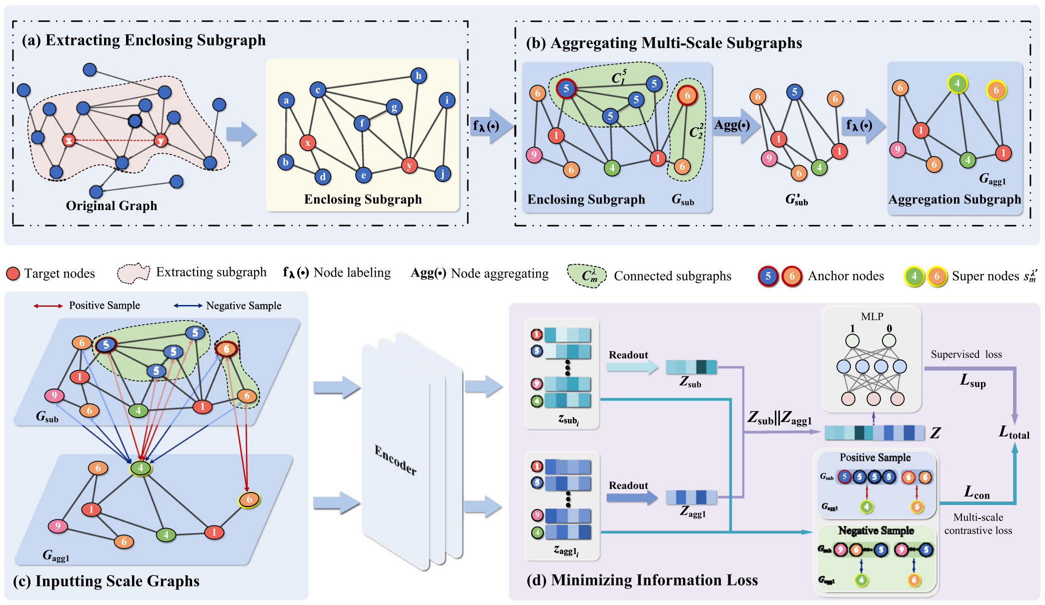


Fig. 1 Overview of the MCAS framework. Parts (a) and (b) describe the formation of scale subgraph G_{agg1} derived from the enclosing subgraph G_{sub}. Parts (c) and (d) outline the prediction model based on multi-scale contrastive learning. In this process, positive sample pairs are created by pairing the node set C_m^λ (before aggregation in the enclosing subgraph) with the supernode $s_m^{\lambda'}$ (after aggregation in the multi-scale graphs). Negative sample pairs are generated by pairing the remaining nodes in the enclosing subgraph with the supernode $s_m^{\lambda'}$ (after aggregation in the multi-scale graphs). Finally, the model parameters are updated to minimize the supervised and contrastive loss

Minimizing information loss. Specific details are described below.

4.1 Constructing enclosing subgraph around the target nodes

Similar to classical supervised methods, we leverage subgraph features for link prediction (Fig. 1(a)). To capture the relative positional relationships between target nodes, we extract h-hop subgraphs centered around the target nodes. Each node within the enclosing subgraph is labeled through a distance-based labeling function, which encodes the node's structural importance within the subgraph.

Enclosing subgraph extraction In link prediction, capturing the local graph structure around target nodes is essential for predicting potential links. To achieve this, we extract h-hop enclosing subgraph for the target node pair (v_x, v_y), as defined by (2) and (3):

$$V_{\mathrm{sub}} = \left\{v \mid \min(d(v, v_x), d(v, v_y)) \leq h\right\} \ \forall v \in V, \tag{2}$$

$$E_{\mathrm{sub}} = \left\{e_{v_i, v_j}\right\} \cap E, \forall v_i, v_j \in V_{\mathrm{sub}}, \tag{3}$$

in this context, the function $d(\cdot)$ denotes the shortest distance from target nodes $(v_x,\ v_y)$ to any other node when extracting enclosing subgraph, while h specifies the length of hops of the walk neighbors. The set of nodes in the subgraph is represented as $V_{\mathrm{sub}} = \{v_1, v_2, ..., v_n\}$, and the set of links is denoted as E_{sub}. Consequently, the enclosing subgraph for any pair of target nodes $(v_x,\ v_y)$ is defined as $G_{\mathrm{sub}} = (V_{\mathrm{sub}}, E_{\mathrm{sub}})$, which corresponds to the newly generated h-hop enclosing subgraph.

Node labeling To evaluate the structural importance of each node within an enclosing subgraph, we introduce a label generated by a label function $f_\lambda(\cdot)$ [16]. This label captures the positional relationship of a node relative to the target nodes, reflecting the node's significance in predicting the target node pair. According to (4), we can assign corresponding structural importance labels to all nodes v_i in the enclosing subgraph centered at the target nodes (v_x, v_y):

$$f_\lambda(v_i) = \begin{cases} 1, & \text{if } v_i = v_x \text{ or } v_y, \\ 1 + \min(d_{v_x}, d_{v_y}) + d_{v_x} + d_{v_y}, & \text{if } v_i \neq v_x \text{ and } v_y. \end{cases} \tag{4}$$

The function $f_\lambda(\cdot)$ represents the label assigned to each node in the subgraph. Specifically, the target nodes are assigned a label of 1. For all other nodes, their labels are determined as shown in the second line of (4). Here, d_{v_x} and d_{v_y} denote the shortest path distances from the target nodes pairs (v_x, v_y) to node v_i, respectively. If $d_{v_x} = \infty$ or $d_{v_y} = \infty$, then $f_\lambda(v_i) = 0$.

4.2 Building multi-scale subgraphs by aggregating nodes

Network structures at different scales contain complementary information that is valuable for link prediction task and can further enrich graph-level vector representations. Therefore, the enclosing subgraph can be modeled as a series of multi-scale subgraphs of varying granularity. To obtain multi-scale subgraphs, we use aggregation functions to convert enclosing subgraph into K multi-scale subgraphs $(G_{\mathrm{agg1}}, G_{\mathrm{agg2}}, ..., G_{\mathrm{aggK}})$. Then, these K multi-scale subgraphs are fed into a graph neural network model as input separately. By independently processing information at different scales, the model can capture representations at different scales. Finally, these representations are used to predict the presence of a link in the network. In order to strike a balance between efficiency and cost, we perform two iterations to generate three scale graphs: G_{sub}, G_{agg1} and G_{agg2}. MCAS only utilizes the information from G_{sub} and G_{agg1}.

Subgraphs are coarsened by aggregating nodes with similar structural information in the network. Node labels are initially assigned using the labeling function $f_\lambda(\cdot)$ (Eq. (4)). If nodes share the same label and are connected, they are considered redundant and merged into a supernode. Since these nodes maintain an equal distance to the target node, they possess similar structural properties. This aggregation process simplifies the network while preserving its essential characteristics. The aggregation principle is shown in (5):

$$\left\{(v_i, v_j) \mid f_{\lambda(v_i)} = f_{\lambda(v_j)} \ and \ A(v_i, v_j) = 1\right\}, \tag{5}$$

in (5), any two connected nodes v_i and v_j within the enclosing subgraph that share the same label are aggregated. By aggregating redundant nodes, the enclosing subgraph can be effectively transformed into multi-scale subgraphs. Building upon the work presented in [16], we refines the aggregation process. As depicted in Fig. 1, our specific aggregation method encompasses the following steps.

Node aggregating To simplify the graph structure, we identify M connected subgraphs $C_{\mathrm{sub}} = \left\{C_1^\gamma, ..., C_m^\lambda\right\}$ within each enclosing subgraph G_{sub}, where these subgraphs satisfy the condition that all nodes within each connected subgraph share the same label. The superscript denotes the different label values. Each $C_m^\lambda \in C_{\mathrm{sub}}$ consists of nodes that share the same label λ. Subsequently, through the node aggregation operation $\mathrm{Agg}(\cdot)$, the connected subgraphs C_{sub} is aggregated into a set of supernodes $S_{\mathrm{agg1}} = \left\{s_1^\gamma, ..., s_m^\lambda\right\}$. In the aggregation process, an anchor node v_k^λ is randomly selected from C_m^λ, and the remaining nodes in C_m^λ are aggregated to the anchor node v_k^λ. The links of these nodes are reconnected to v_k^λ, and then the remaining nodes are deleted, resulting in the supernode s_m^λ (i.e., $s_m^\lambda = \text{Agg}(C_m^\lambda)$). After traversing all connected subgraphs, the supernode set S_{agg1} is formed, and these supernodes constitute the components of the coarsened subgraph G'_{sub} (as shown in Fig. 1(b)).

Node relabeling After applying the function $f_\lambda(\cdot)$ again to the coarse-grained subgraph G'_{sub}, new labels are assigned to each node within G'_{sub}(including supernodes). The resulting aggregated subgraph G_{agg1}, following relabeling, consists of the supernode set $S_{\text{agg1}} = \left\{s_1^{\gamma'}, ..., s_m^{\lambda'}\right\}$, where the superscript denotes the updated label values within the aggregated graph. For instance, in Fig. 1(b), G_{sub} contains the eligible connected subgraph for aggregation: $C_1^5 = \left\{v_c^5, v_f^5, v_g^5, v_h^5\right\}$. We select $v_c^5 \in C_1^5$ as the anchor node for performing the aggregation operation, leading to the inclusion of supernode s_1^4 in G_{agg1} after the relabeling process.

Finally, by repeating the node relabeling and aggregation operations described earlier, subgraphs at multiple scales (e.g., G_{agg2}) can be derived. As this process is similar to the construction of G_{agg1}, a detailed explanation of the deeper-scale subgraph construction is omitted. The specific effects and performance implications of G_{agg2} are analyzed in the experimental section.

4.3 Minimizing information loss

Through the above aggregation operations, multi-scale enclosing subgraphs G_{sub} and G_{agg1} are obtained. These scale subgraphs serve as inputs to the encoder, sequentially producing multi-scale representation vectors Z_{sub} and Z_{agg1}. To obtain high-quality node representations and facilitate the model's learning during training phase, the MCAS framework updates model parameters by optimizing both the supervised link prediction loss and the multi-scale contrastive loss. These two modules will be explained in detail below.

Supervised link prediction loss By using the encoder, we aggregate neighborhood information of nodes through convolutional operations, enabling the learning of more expressive node representations, as shown in (6):

$$Z_{\text{sub}} = f_s(X_s, A_s), \quad Z_{\text{agg1}} = f_{a1}(X_{a1}, A_{a1}), \tag{6}$$

where $f(\cdot)$ denotes the encoder. $Z_{\text{sub}} \in \mathbb{R}^{n\times d}$ and $Z_{\text{agg1}} \in \mathbb{R}^{n'\times d}$ represent the embedding matrices of the multi-scale subgraphs, where n, n' denote the number of nodes in G_{sub} and G_{agg1}, respectively. The parameter d specifies the dimensionality of the embedding. X_s refers to the feature matrix of all nodes in the enclosing subgraph. The features are constructed by one-hot encoding the node labels and appending the embedding or attribute vector of each node to the corresponding row in the feature matrix [11]. A_s indicates the adjacency matrix of the G_{sub}. Similarly, X_{a1} denotes the feature matrix of the G_{agg1}, where the features of the supernodes correspond to the features of the anchor nodes. A_{a1} represents the adjacency matrix of the G_{agg1}.

Subsequently, the node embeddings $z_{\text{sub}_i} \in \mathbb{R}^{1\times d}$ for the node v_i within the enclosing subgraph G_{sub} are obtained by the GNN module in (7):

$$z^l_{\text{sub}_i} = \sigma\Big(\sum_{v_j \in \Gamma(\text{sub}_i)} \hat{D}_s^{-\frac{1}{2}} \hat{A}_s \hat{D}_s^{-\frac{1}{2}} z^{l-1}_{\text{sub}_j} W_s^{l-1}\Big), \tag{7}$$

where $\Gamma(\text{sub}_i)$ represents the neighbors of node v_i in G_{sub}, $\hat{D}_s$ is diagonal degree matrix of the G_{sub} with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. $\hat{A}_s = A_s + I$ denotes the adjacency matrix of the G_{sub} with a self-loop and I is the identity matrix. $z^{l-1}_{\text{sub}_j}$ refers to the embeddings of the neighbor node v_j of node v_i at layer $l-1$. W_s^{l-1} is the trainable graph convolution parameter matrix in the l-1 layer. $\sigma(\cdot)$ is the nonlinear activation function. To obtain the graph-level representation of the G_{sub}, a readout operation is performed on the obtained node representations, as shown in (8):

$$Z_{\text{sub}} = Readout\left(z^l_{\text{sub}_i} : v_i \in V_{\text{sub}}\right), \tag{8}$$

specifically, average pooling is used as the readout function, aggregating the node representations to produce the final graph-level representation Z_{sub}.

Similarly, the embeddings of the aggregated subgraph G_{agg1} is denoted as Z_{agg1}. The convolution process follows the same procedure as that for the enclosing subgraph G_{sub}, the embedding representation z_{agg1_i} of node v_i in the G_{agg1} is given in (9):

$$z^l_{\text{agg1}_i} = \sigma\Big(\sum_{v_j \in \Gamma(\text{agg1}_i)} \hat{D}_{a1}^{-\frac{1}{2}} \hat{A}_{a1} \hat{D}_{a1}^{-\frac{1}{2}} z^{l-1}_{\text{agg1}_j} W_{a1}^{l-1}\Big). \tag{9}$$

Likewise, the subgraph embedding obtained after a single aggregation is denoted as Z_{agg1}:

$$Z_{\text{agg1}} = Readout\left(z^l_{\text{agg1}_i} : v_i \in V_{\text{agg1}}\right). \tag{10}$$

Through the above process, we obtain the corresponding representation vectors Z_{sub} and Z_{agg1} from the multi-scale input graphs. These vectors are concatenated to generate the representation of the entire scaled subgraph for supervised link prediction, as defined in (11):

$$Z = Concat(Z_{\text{sub}}, Z_{\text{agg1}}). \tag{11}$$

After obtaining the final representation vector Z, it is fed into a fully connected MLP network to obtain the predicted probability $\hat{y}_t$ of the target node pair t being connected, as shown in (12):

$$\hat{y}_t = MLP\,(Z)\,. \tag{12}$$

Finally, the model is trained using the cross-entropy loss function through the supervision loss, as formulated in (13):

$$L_{\text{sup}} = -\frac{1}{|T|}\sum_{t=1}^{|T|} y_t \log \hat{y}_t + (1 - y_t)\log(1 - \hat{y}_t), \tag{13}$$

where $|T| = |T|^+ + |T|^-$, $|T|^+$ represents the positive sample number of target node pairs with existing links and $|T|^-$ denotes the negative sample number of target nodes where a link does not exist. $\hat{y}_t$ represents the predicted probability for the target link t, while y_t denotes the corresponding ground truth label 1 or 0.

Multi-scale contrastive loss Since multi-scale subgraphs capture different granular views that reflect the inherent structure of networks, and the aggregation process may lose some node information, we use contrastive learning to compensate for this missing information. We elaborate on the contrastive learning process between the original enclosing subgraph G_{sub} and the aggregated subgraph G_{agg1}. In G_{agg1}, the supernode $s_m^{\lambda'} \in S_{\text{agg1}}$ is obtained by aggregating all nodes in $C_m^{\lambda} \in C_{\text{sub}}$. Naturally, the nodes in C_m^{λ} and the corresponding supernode $s_m^{\lambda'}$ share similar structural information. During the contrastive learning process, all nodes in C_m^{λ} are treated as positive pairs with its corresponding supernode $s_m^{\lambda'}$, while the remaining nodes V'_{sub} (excluding the target nodes) in G_{sub} are treated as negative pairs with $s_m^{\lambda'}$. For example, in Fig. 1(c), within the orginal subgraph G_{sub}, the nodes $\left\{v_c^5, v_f^5, v_g^5, v_h^5\right\} \in C_1^5$ and the supernode s_1^4 in the aggregration subgraph G_{agg1} form the positive pair. The remaining nodes $\left\{v_a^6, v_b^9, v_d^6, v_e^4, v_i^6, v_j^6\right\}$ and the supernode s_1^4 form the negative pair. By maximizing the similarity measure between positive sample pairs and minimizing it between negative sample pairs, contrastive learning effectively preserves the structural and semantic information of the original graph.

Leveraging the node embeddings obtained from (7) and (9), we optimize our objective by maximizing the mutual information between the enclosing and aggregated subgraphs. This is achieved by computing the Normalized Temperature-scaled cross entropy loss (NT-Xent) [44] as defined in (14):

$$L_{\text{con}} = \frac{1}{|T|}\sum_{t=1}^{|T|} -(\frac{1}{|M|}\sum_{m=1}^{|M|} \log \frac{exp(sim(z_{C_m^{\lambda}}, z_{s_m^{\lambda'}})/\tau)}{\sum_{\text{sub}_i \in V'_{\text{sub}}} exp(sim(z_{\text{sub}_i}, z_{s_m^{\lambda'}})/\tau)}), \tag{14}$$

where $|T|$ denotes the number of target nodes, $|M|$ represents the number of supernodes, and the $z_{C_m^{\lambda}}$ is defined as $\frac{1}{|C_m^{\lambda}|}\sum_{k=1}^{|C_m^{\lambda}|} z_{\text{sub}_k}$,where the embeddings z_{sub_k} of all nodes within the connected subgraphs C_m^{λ} in the subgraph are averaged. τ is a temperature parameter.

The contrastive learning process between the remaining multi-scale graphs is similar to the process described above. Specifically, after obtaining the corresponding representation vectors, the model is optimized using the supervised loss and contrastive loss based on positive and negative sample pairs between the scale graphs.

Total Loss The overall loss L_{total} combines supervised loss and self-supervised task loss, including binary cross-entropy loss and normalized temperature-scaled cross-entropy loss, as shown in (15):

$$L_{\text{total}} = \alpha L_{\text{sup}} + (1 - \alpha)L_{\text{con}}, \tag{15}$$

where α is adjustable hyperparameters.

4.4 Algorithm

Algorithm 1 MCAS's algorithmic

```
Input: Original graph $G = (V, E)$, the training target nodes set $T$
Output: Prediction probability $\hat{y}_t$
 1: for each target link $t \in T$ do
 2:     Extract $h$-hop subgraph $G_{sub}$ via (2), (3)
 3:     Apply node labeling to $G_{sub}$ via (4)
 4:     Generate aggregated subgraph using (5)
 5:     Apply encoder to obtain representation $Z_{sub}$ and $Z_{agg1}$ via (8), (10)
 6:     Calculate score $\hat{y}_t$ using a binary classification network via (12)
 7:     Calculate supervised link prediction loss and contrastive loss via (13), (14)
 8:     Calculate total loss $L_{total}$ via (15)
 9:     Update parameters
10: end for
```

Algorithm 1 outlines the framework of our proposed method. Consider a enclosing subgraph with n nodes, where the target link is t, the time complexity of obtaining an enclosing subgraph is $O(\langle d\rangle^h)$, where $\langle d\rangle$ is the average degree of the network, and h is the number of hops. The operations for multi-scale subgraphs depend on the process of node aggregation, which has a time complexity of $O(n^2)$. Therefore, the time complexity for two scale subgraphs is $2(O(\langle d\rangle^h * n^2))$. Subsequently, model training process is divided into supervised loss and contrastive loss. The supervised loss is primarily determined by (7), (9). Equations (7) and (9) involve the generation of node representations using GNN, resulting in a time complexity of $O(n^2)$ [45]. The contrastive loss, governed by (14), considers negative samples selected from all nodes in the subgraph, excluding the target node and positive samples. This process incurs a time complexity of $O(n^2)$. In summary, the time complexity of a enclosing subgraph in the MCAS model is approximately $2(O(\langle d\rangle^h * n^2)) + 2O(n^2) \approx O(\langle d\rangle^h * n^2)$. This indicates that the time complexity of the MCAS model is primarily influenced by the size of the multi-scale subgraphs, the number of scales, the GNN computation, and the selection of negative samples in contrastive learning.

5 Experiments

5.1 Datasets and baseline models

5.1.1 Datasets

To evaluate the effectiveness of MCAS across different domains, we conduct experimental research on nine real-world datasets. Refer to Table 1 for detailed information about each dataset, all available from https://noesis.ikor.org/datasets/link-prediction.

- Biological networks (YST and HPD [46]): In these networks, the nodes denote proteins, and the edges indicate interactions between proteins.
- Co-authorship networks (KHN, ZWL, and LDG [47]): In these networks, the nodes represent authors, and the edges indicate co-authorship relationships between authors in different fields.
- Social networks (ADV [48]): The nodes stand for individuals, and the edges represent social relationships between individuals.
- Citation networks (CORA, CITESEER, and PUBMED [29]): In the three networks, nodes correspond to published papers, the edges depict citation relationships between papers, and the features are represented by the bag-of-words vectors of the respective paper titles.

5.1.2 Baseline models

This section presents a concise overview of the baseline methods used for comparison with MCAS. These include heuristic-based approaches, representation learning methods, state-of-the-art (SOTA) graph neural network techniques, and contrastive learning-based methods.

Heuristic-based methods

- Common Neighbor (CN) [17]: Nodes with a higher count of shared neighbors are more likely to have a connecting link and the connecting likehood can be quantified by simply counting the number of common neighbors.
- Resource Allocation (RA) [19]: This method improves upon the CN algorithm by normalizing the common neighbors' score. This adjustment effectively reduces the impact of skewed node degree distributions, thereby enhancing the accuracy of link prediction results.
- KATZ [20]: The method traverses all possible path sets and introduces an exponential decay factor based on path length to compute the similarity between node pairs. This approach effectively captures multi-hop relationships between nodes.

Representation learning-based methods

- Node2Vec [25]: This method extends DeepWalk by proposing a new node representation learning model. It adopts a biased random walk strategy combining breadth-

Table 1 Topology information of nine networks: the number of nodes (N), links (M), average node degree ($\langle d\rangle$), network density (S), node features (F), the "-" represents a network with no attribute features, where the features are the one-hot encoding vectors generated from the node labels λ, and the graph type (Type)

Datasets	N	M	$\langle d\rangle$	S	F	Type
YST	2284	6646	5.82	0.255%	-	Biology network
CORA	2708	5278	3.90	0.144%	1433	Citation network
CITESEER	3327	4676	2.74	0.085%	3703	Citation network
KHN	3772	12718	6.74	0.179%	-	Co-authorship
ADV	5155	39285	15.24	0.296%	-	Social network
ZWL	6651	54182	16.29	0.245%	-	Co-authorship
LDG	8324	41532	9.98	0.120%	-	Co-authorship
HPD	8756	32331	7.38	0.084%	-	Biology network
PUBMED	19717	44338	2.28	0.023%	500	Citation network

first and depth-first searches, effectively capturing both local and global structural information of nodes in the network. As a result, it facilitates the learning of more discriminative node representations.

GNN-based methods

- SEAL [11]: This can be considered as the pioneering work of GNN to the link prediction task. It extracts enclosing subgraph around target links and computes the probability of link existence based on the binary labels assigned to nodes within these subgraphs.
- MLINK [16]: This method converts enclosing subgraphs into multi-scale input graphs with varying granularities. These subgraphs are then processed by a shared-parameter encoder, and the probability of link existence is predicted using a binary classification model.
- LGLP [47]: Convert enclosing subgraph into line graphs of the same scale and transform the link prediction task into a binary classification task.
- BSAL [49]: This method performs link prediction using a dual-component framework that integrates both graph topology and feature information, enabling a more comprehensive analysis of link existence.
- GCCL [50]: This method applies a GNN to generate node embeddings, reframing link prediction as a graph classification problem. It further combines the strengths of GNNs and Capsule Networks (CapsNets) [51] to learn rich feature representations of the edge feature graph, effectively capturing node pair attributes from multiple perspectives.

Contrastive learning-based methods

- SUBG-CON [52]: By leveraging the close relationship between central nodes and their sampled subgraphs, this method captures regional structural information. It employs subgraph sampling as a data augmentation technique to enhance self-supervised graph representation learning.
- LGCL [42]: Subgraph views are obtained by sampling h-hop subgraphs for target node pairs. These sampled subgraphs are then transformed into line graphs, and a novel cross-scale contrastive learning approach is applied to both the line graphs and subgraphs. The goal of this method is to maximize the mutual information between these representations, effectively combining both structural and feature information.

5.2 Experiment setup

5.2.1 Training optimization

To evaluate the effectiveness of MCAS method, we utilize the Adam optimization algorithm with a learning rate of 0.001 to optimize model parameters for all datasets. For our experiments, we randomly divided the links E into training and testing sets with p set to 30%, 40%, 50%, 60%, 70%, 80%, and 90%, respectively. The damping factor for the KATZ method is set to 0.001. Node2Vec embeddings are generated with 128 dimensions, and a batch size of 50 is used for training. Our method utilizes a three-layer graph convolutional network to extract node features, with the number of channels for the three convolutional layers set to 32, 32, 32, respectively. The binary classification network consists of two fully connected neural network layers, with dimensions of 128 and 1. The network model is trained for 50 epochs. To embody a more effective aggregation process, the scale used to extract subgraph includes 2-hops neighbors. The experiments are conducted on an NVIDIA GeForce RTX 2080 Ti GPU.

5.2.2 Evaluation metrics

In this experiment, two evaluation metrics are used to measure the performance of MCAS algorithm: Area Under the ROC Curve (AUC) [53] and Area Under the Precision-Recall curve (AUPR) [54]. AUC measures the quality of the method from an overall perspective [1]. For datasets with extremely imbalanced class distributions, AUPR offers a more granular performance evaluation, allowing for a more accurate assessment of the algorithm's classification effectiveness under varying decision thresholds.

AUC is defined as the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample. Specifically, in each iteration, an link from the test set is randomly selected as a positive sample, followed by the random selection of a non-existent link as a negative sample. After performing n comparisons, if the score of a missing link from the test set is higher than the score of a non-existent link, $n^{'}$ is incremented. If the scores are equal, $n^{''}$ is incremented. AUC can be calculated as (16):

$$AUC = \frac{n^{'} + 0.5 \times n^{''}}{n}, \tag{16}$$

the value of AUC typically ranges between [0,1], where higher values signify a greater ability of the algorithm to accurately distinguish between positive and negative instances.

The distribution of samples in real-world networks is often highly imbalanced, resulting in a substantial disparity between the number of positive and negative instances. The AUPR metric is more useful and informative for imbalanced datasets [55]. It is an evaluation metric used to assess the performance of binary classifiers. This value is calculated based on the precision-recall curve, which is a plot of precision values on the Y-axis and recall values on the X-axis. The

Table 2 Results of AUC comparison with baseline methods (80% training links)

	Method	YST	CORA	CITESEER	KHN	ADV	ZWL	LDG	HPD	PUBMED
Heuristic	CN	68.23±0.42	69.04±0.22	69.83±0.14	76.89±0.13	87.37±0.85	88.93±0.31	86.78±0.23	73.73±0.25	79.69±0.28
	RA	68.83±0.22	70.34±0.16	70.64±0.33	78.23±0.27	88.27±0.36	90.34±0.67	87.93±0.14	74.15±0.76	79.31±0.35
	KATZ	81.93±0.37	76.21±0.16	73.62±0.15	83.73±0.34	90.35±0.14	92.64±0.34	91.23±0.47	84.77±0.21	79.31±0.35
Embedding	Node2Vec	83.97±0.43	83.80±0.52	84.22±0.37	83.27±0.64	91.56±0.21	93.09±0.35	92.29±0.41	87.89±0.16	87.67±0.10
GNN	SEAL	90.33±0.21	92.78±0.36	89.03±0.13	92.32±0.19	94.43±0.42	96.22±0.31	95.54±0.32	92.76±0.35	93.32±0.13
	MLINK	91.76±0.37	92.54±0.34	91.67±0.57	93.11±0.28	94.96±0.70	96.54±0.11	96.23±0.67	92.39±0.36	94.52±0.13
	LGLP	92.37±0.15	90.47±0.49	92.33±0.34	93.51±0.32	95.67±0.23	97.01±0.32	96.37±0.76	93.01±0.25	92.43±0.42
	BSAL	91.23±0.45	91.14±0.53	**93.39±0.53**	92.54±0.24	93.89±0.31	95.92±0.58	94.67±0.32	92.23±0.29	92.83±0.42
	GCCL	88.13±0.36	94.42±0.45	89.32±0.77	91.74±0.21	93.67±0.23	95.32±0.03	94.89±0.56	91.59±0.31	93.10±0.32
GCL	SUBG-CON	89.31±0.52	85.89±0.56	85.45±0.08	90.33±0.89	89.29±0.28	92.79±0.22	91.94±0.43	90.51±0.74	89.83±0.32
	LGCL	92.55±0.19	91.24±0.07	91.23±0.32	92.63±0.54	95.24±0.56	95.89±0.23	96.41±0.45	93.22±0.98	93.22±0.04
Ours	MCAS	**93.73±0.42**	**94.66±0.23**	92.99±0.37	**94.43±1.02**	**96.63±0.43**	**97.45±0.96**	**97.25±0.41**	**93.95±0.12**	**95.63±0.45**

Top results are in bold and second-best results are underlined

higher the AUPR value of a model, the better its predictive performance. Precision and recall values can be calculated using (17) and (18), respectively:

$$Precision = \frac{TP}{TP + FP}, \tag{17}$$

$$Recall = \frac{TP}{TP + FN}, \tag{18}$$

where TP (True Positive) are the positive data item predicted as positive, FP (False Positive) represents negative data items incorrectly predicted as positive and FN (False Negative) are the positive data items incorrectly predicted as negative.

5.3 Experiment results and analysis

In this section, we conduct perform a validation study of the proposed method by addressing the following questions:

- **Q1**: How does our method perform compare to baseline methods in terms of AUC and AUPR metrics?
- **Q2**: How robust is our method compared to that of baseline methods across different proportions of the training set?
- **Q3**: Do the individual components of our method contribute to enhancing the accuracy of the final link prediction task?
- **Q4**: What are the distributions of the hyperparameters for each component of the model?

Table 3 Results of AUPR comparison with baseline methods (80% training links)

	Method	YST	CORA	CITESEER	KHN	ADV	ZWL	LDG	HPD	PUBMED
Heuristic	CN	70.03±0.17	70.67±0.34	70.23±0.41	78.37±0.32	88.17±0.62	89.93±0.47	87.24±0.33	74.35±0.07	80.14±0.22
	RA	70.56±0.34	71.66±0.09	71.53±0.34	79.29±0.53	89.56±0.23	90.93±0.16	88.76±0.23	75.47±0.21	81.96±0.11
	KATZ	84.13±0.23	77.83±0.35	74.97±0.25	83.96±0.17	91.27±0.49	93.97±0.35	91.43±0.37	87.56±0.32	83.82±0.41
Embedding	Node2Vec	85.33±0.45	84.31±0.27	84.86±0.98	84.78±0.11	92.05±0.34	93.56±0.19	92.87±0.35	87.97±0.64	88.13±1.71
GNN	SEAL	90.54±0.34	91.98±0.21	91.52±0.22	91.95±0.32	94.57±0.27	95.73±0.33	95.89±0.51	92.92±0.26	90.54±0.12
	MLINK	91.93±0.71	90.34±0.27	92.32±0.21	93.89±0.21	95.45±0.32	96.93±0.56	96.18±0.39	93.08±0.42	92.77±0.31
	LGLP	92.85±0.26	89.96±0.57	92.43±0.22	94.23±0.21	95.89±0.67	97.27±0.18	96.92±0.54	93.80±0.32	90.97±0.32
	BSAL	90.96±0.35	92.16±0.32	**94.58±0.71**	90.89±0.21	92.85±0.78	94.53±0.71	93.31±0.62	91.89±0.38	92.74±0.33
	GCCL	89.89±0.45	91.73±0.31	90.45±0.67	91.82±0.17	94.34±0.16	93.89±0.91	95.35±0.74	91.55±0.32	94.21±0.44
GCL	SUBG-CON	89.67±0.24	86.43±0.12	89.51±0.57	91.61±0.34	91.03±0.78	95.43±0.83	93.72±0.21	92.67±0.53	92.04±0.78
	LGCL	91.68±0.56	90.79±0.32	90.12±0.34	92.72±0.63	96.67±0.21	95.53±0.53	93.78±0.34	92.36±0.67	92.67±0.22
Ours	MCAS	**92.93±0.76**	**93.04±0.25**	93.76±0.24	**94.72±0.31**	**96.78±0.32**	**97.59±0.21**	**97.05±0.32**	**94.21±0.15**	**95.36±0.76**

Top results are in bold and second-best results are underlined

5.3.1 Performance comparison with baseline methods (Q1)

To validate the performance of the MCAS method, we compare it with eleven baseline methods across nine datasets. For each network dataset, we randomly select $p = 80\%$ of the existing links as the training set, while the remaining links are used as the test set. To reduce evaluation randomness and improve reliability, we adopt random sampling validation, dividing the dataset randomly 50 times and calculating the average AUC and AUPR. The AUC and AUPR results of the MCAS method on the 80% training datasets are shown in Tables 2 and 3, respectively.

As can be seen from the results in Tables 2 and 3, MCAS significantly outperforms other methods in terms of AUC and AUPR, demonstrating the effectiveness of multi-scale contrastive learning in link prediction task. Additionally, MCAS enhances prediction performance compared to the MLINK method, providing initial evidence for the effectiveness of applying contrastive loss training at a multi-scale level. GNN-based methods have significantly improved prediction performance compared to heuristic and embedding-based methods. For example, SEAL and MLINK outperform Node2vec on all networks. This demonstrates the powerful feature extraction and domain information capture capabilities of GNN-based methods, leading to significant advantages in performance. Moreover, experimental results demonstrate that models employing subgraph extraction methods generally achieve slightly better prediction performance on most datasets. This indicates that the local neighborhood structure of the target nodes has a significant

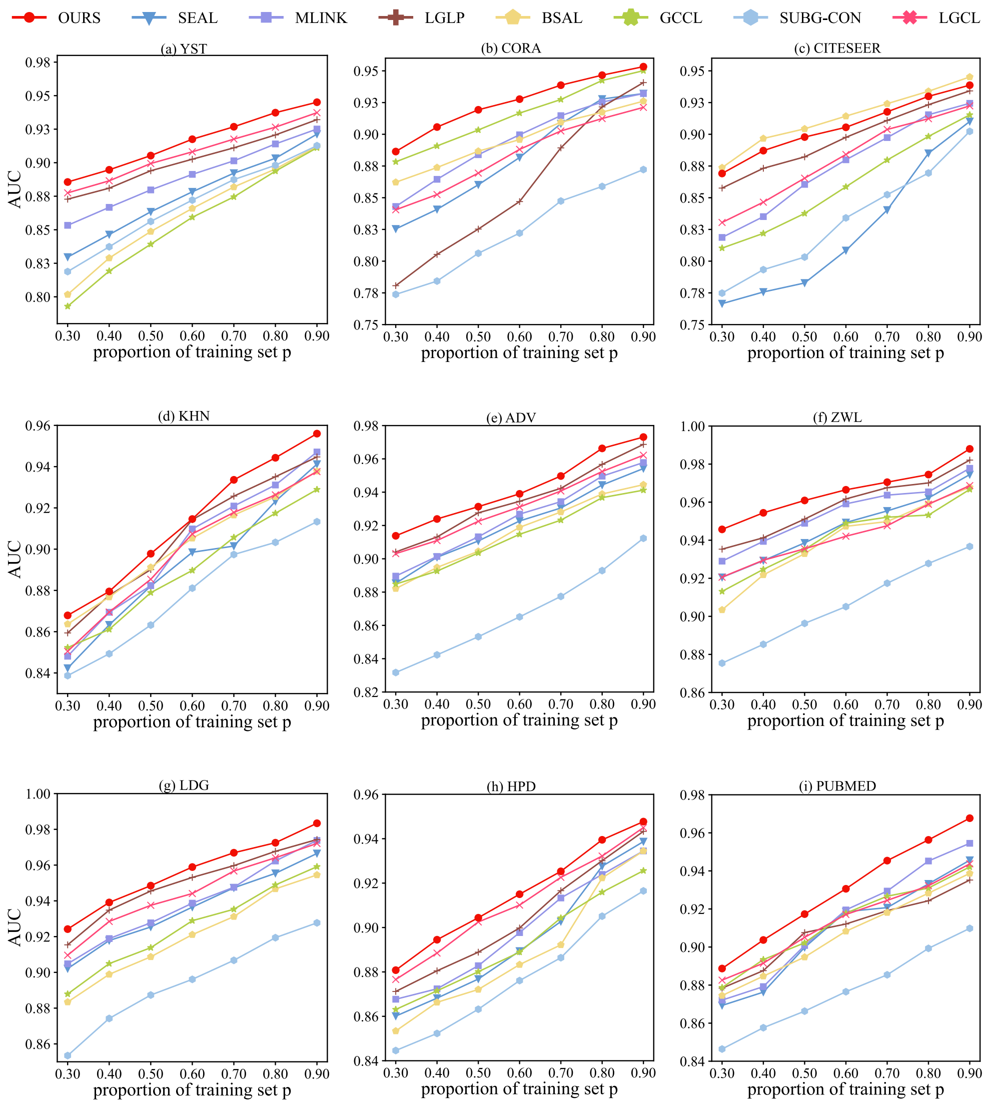


Fig. 2 Predictive performance of eight methods on nine datasets measured by AUC. The x-axis ranges from 0.3 to 0.9, indicating the training proportion, while the y-axis represents the AUC value

impact on prediction results. For instance, SEAL, MLINK, and LGLP outperform BASL and GCCL on networks such as YST and KHN.

In the view of the datasets, the performance results verified on networks with denser structures are significantly better than those on networks with relatively sparse structures. For example, the performance of datasets such as ADV and ZWL in heuristic methods significantly outperforms that of other datasets, closely linked to their relatively dense edge distributions. These denser networks contain richer topological information, which often leads to improved link prediction performance. In contrast, networks such as CITESEER and HPD exhibit sparser edge distributions, which results in comparatively lower performance.

To demonstrate the model's effectiveness in self-supervised learning for link prediction, we evaluate the aggregation comparison strategy against two other contrastive self-supervised learning GCL approaches: SUBG-CON and LGCL. The experiments assess the AUC and AUPR metrics across nine datasets, focusing on a comparison between two augmentation strategies: subgraph comparison and line graph comparison. The corresponding results are detailed in the penultimate and final rows of Tables 2 and 3. The experimental results indicate that MCAS surpasses two contrastive strategies. While the SUBG-CON method effectively learns node representations through contrastive learning on diverse subgraphs, it falls short in directly addressing link prediction task. LGCL reduces dependency on labels but struggles to capture fine-grained node information due to its focus on global patterns. In contrast, MCAS method excels at capturing local node-level characteristics by comparing aggregated subgraph representations before and after node manipula-

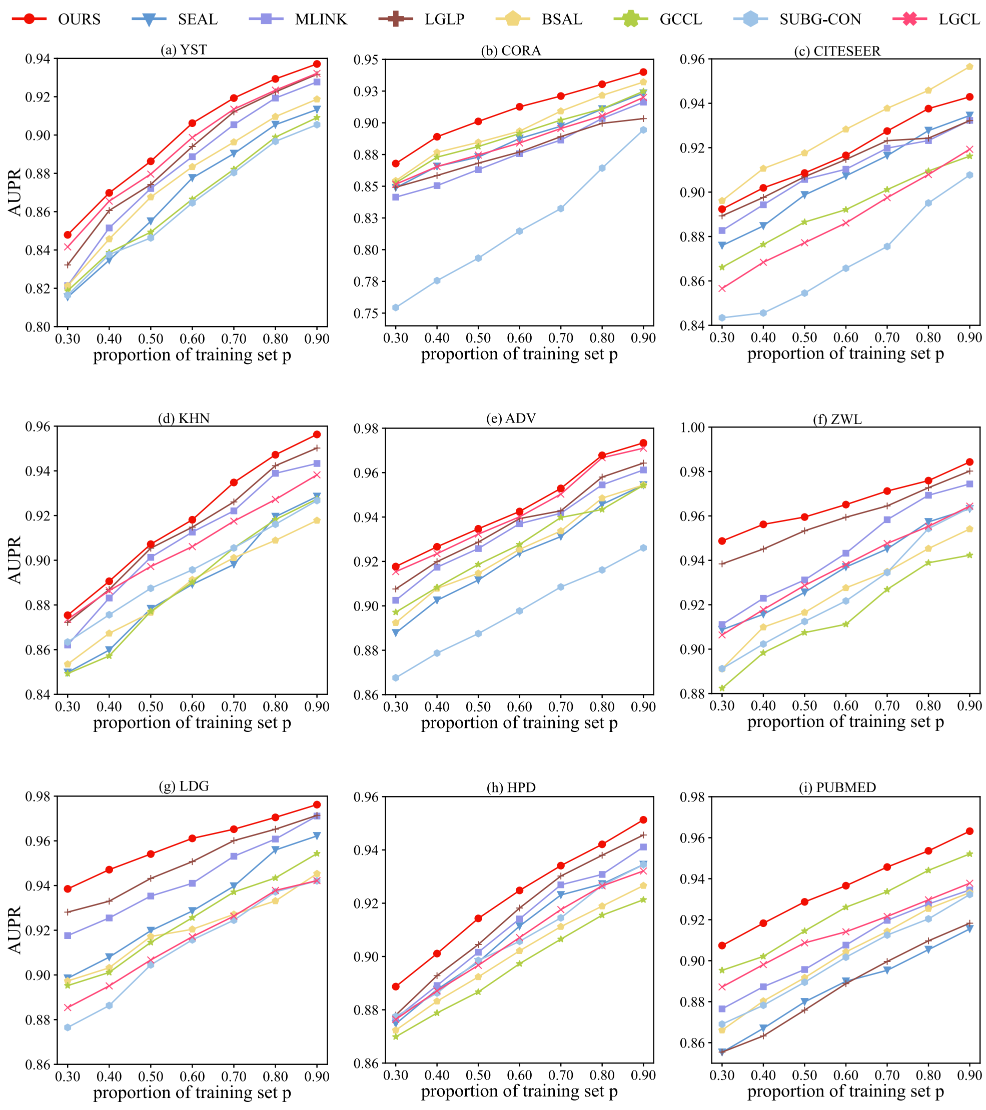


Fig. 3 Predictive performance of eight methods on nine datasets measured by AUPR. The x-axis ranges from 0.3 to 0.9, indicating the training proportion, while the y-axis represents the AUPR value

tion, thereby achieving superior performance compared to the other methods.

5.3.2 The performance robustness of the MCAS method (Q2)

To validate the robustness of the model, we randomly select link proportions of $p = 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%$ from the networks as training sets. The experimental results presented in Tables 2 and 3 indicate that heuristic and embedding methods do not show significant performance advantages. Therefore, we compare our proposed method MCAS with five SOTA methods SEAL, MLINK, LGLP, BASL, GCCL and two contrastive learning methods SUBGCON, LGCL. The results of these comparisons are shown in Figs. 2 and 3.

Overall, the experiments demonstrate that MCAS generally outperforms other methods in terms of model robustness across different training sets. The only exception is the CITESEER network, where the BASL method outperforms all others, with MCAS ranking second. BASL's effective combination of semantic and structural features within CITESEER directly contributes to its superior performance. Conversely, the MCAS method demonstrates consistent performance across different training sets, highlighting its robustness. In addition, as the training set proportion increases, the structural information available in the network expands, leading to improved performance of state-of-the-art (SOTA) methods and contrastive learning methods on both AUC and AUPR metrics. The MLINK method achieves performance improvement by converting enclosing subgraphs in the network into subgraphs of different scales as input. However, this aggregation process can lead to the loss of network structural information. MCAS addresses this issue by mitigating the adverse effects of such information loss within the models architecture, resulting in substantial improvements.

5.3.3 Ablation study (Q3)

To evaluate the effectiveness of the multi-scale subgraph module and contrastive learning component in the MCAS framework, we conduct experiments for each component one by one. The detailed explanations of each component are provided in Table 4. For each dataset, p=30%, 50%, and 80% of the link from networks are randomly sampled as the training set, and the results are shown in Figs. 4 and 5.

As illustrated in Figs. 4 and 5, incorporating multi-scale subgraphs can effectively enrich vector representations and enhance model robustness. The single-scale input subgraph G_{sub} contain relatively limited information. The aggregation operation results in the subgraphs within the aggregated subgraphs G_{ggg1} and G_{ggg2} gradually losing information, which causes a decline in the model's prediction accuracy. Experimental results demonstrate that the model using G_{sub}+G_{agg1} as input to the encoder achieves superior performance compared to using G_{sub}+G_{agg2}. Furthermore, when feeding the information from G_{sub}+G_{agg1}+G_{agg2} into the encoder, we observe that incorporating three-scale information further enhances model performance to a certain extent.

However, due to the inherent limitations of the aggregation process, certain constraints arise from information loss. To address this issue, we explore various combinations includ-

Table 4 The specific explanations of each component

Model	Explanations
G_{sub}	Use the single-scale enclosing subgraph G_{sub} as the input graph
G_{agg1}	Use the single-scale aggregating subgraph G_{agg1} as the input graph.
G_{agg2}	Use the single-scale aggregating subgraph G_{agg2} as the input graph.
G_{sub}+G_{agg1}	Use two-scale subgraphs G_{sub} and G_{agg1} as input graphs
G_{sub}+G_{agg2}	Use two-scale subgraphs G_{sub} and G_{agg2} as input graphs.
G_{sub}+G_{agg1}+G_{agg2}	Use three-scale subgraphs G_{sub}, G_{agg1} and G_{agg2} as input graphs
G_{sub}+G_{agg1}+con (MCAS)	Use two-scale subgraphs G_{sub}, G_{agg1} as input graphs with contrastive learning (con)
G_{sub}+G_{agg2}+con	Use two-scale subgraphs G_{sub}, G_{agg2} as input graphs with contrastive learning (con)
G_{agg1}+G_{agg2}+con	Use two-scale subgraphs G_{agg1}, G_{agg2} as input graphs with contrastive learning (con).
G_{sub}+G_{agg1}+G_{agg2}+con	Combine the supervised loss of the three-scale subgraphs G_{sub}, G_{agg1} and G_{agg2} with the contrastive loss between G_{sub} and G_{agg1}, as well as the contrastive loss between G_{agg1} and G_{agg2}.

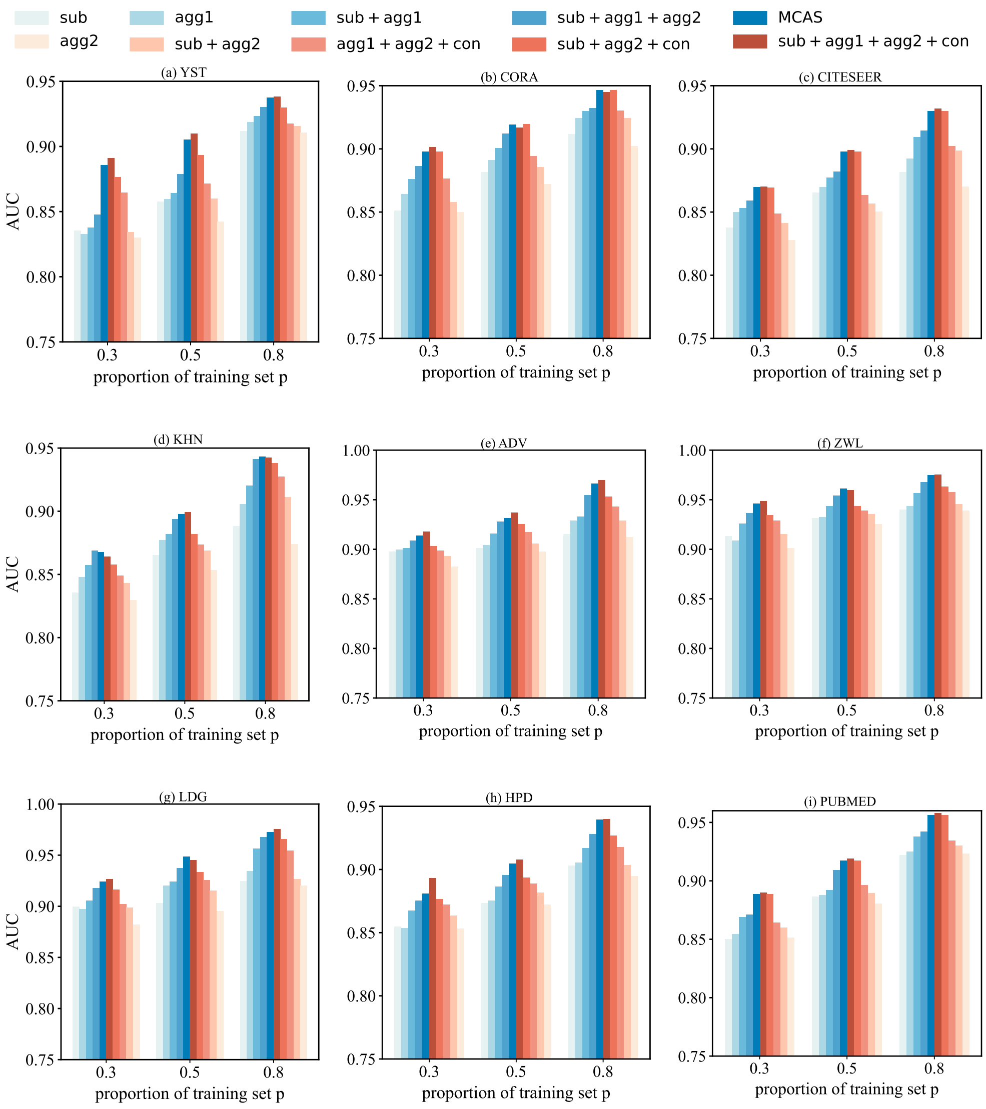


Fig. 4 Results of the ablation experiments on AUC for nine datasets. The x-axis ranges from 0.3 to 0.5 to 0.8, indicating the training proportion, while the y-axis represents AUC value

ing G_{sub}+G_{agg1}+con (MCAS), G_{sub}+G_{agg2}+con, G_{agg1}+G_{agg2}+con, and G_{sub}+G_{agg1}+G_{agg2}+con by applying contrastive learning. Experimental results demonstrate that the performance of all four combination models is enhanced with the incorporation of more scale information. Notably, the MCAS and G_{sub}+G_{agg1}+G_{agg2}+con combinations achieved the best results across all datasets. Although the G_{sub}+G_{agg1}+G_{agg2}+con combination offers richer contrastive information, it comes at the cost of reduced model efficiency. This is particularly evident in large networks such as PUBMED where training becomes significantly slower. Consequently, we choose MCAS for its superior performance and efficiency.

From the perspective of datasets, the MCAS model performs exceptionally well in networks such as YST, ADV, and ZWL because in denser networks (as shown by the network density (S) in Table 1), a single aggregation is sufficient to fully utilize the topological information of enclosing subgraphs. This effectively captures the tight local structural features between nodes, maximizing the capability of local topological features and enabling the model to accurately predict potential links[56]. In contrast, in sparser networks such as CORA, CITESEER, and PUBMED, the G_{sub}+G_{agg2}+con model, which incorporates twice-aggregated subgraphs, shows performance similar to

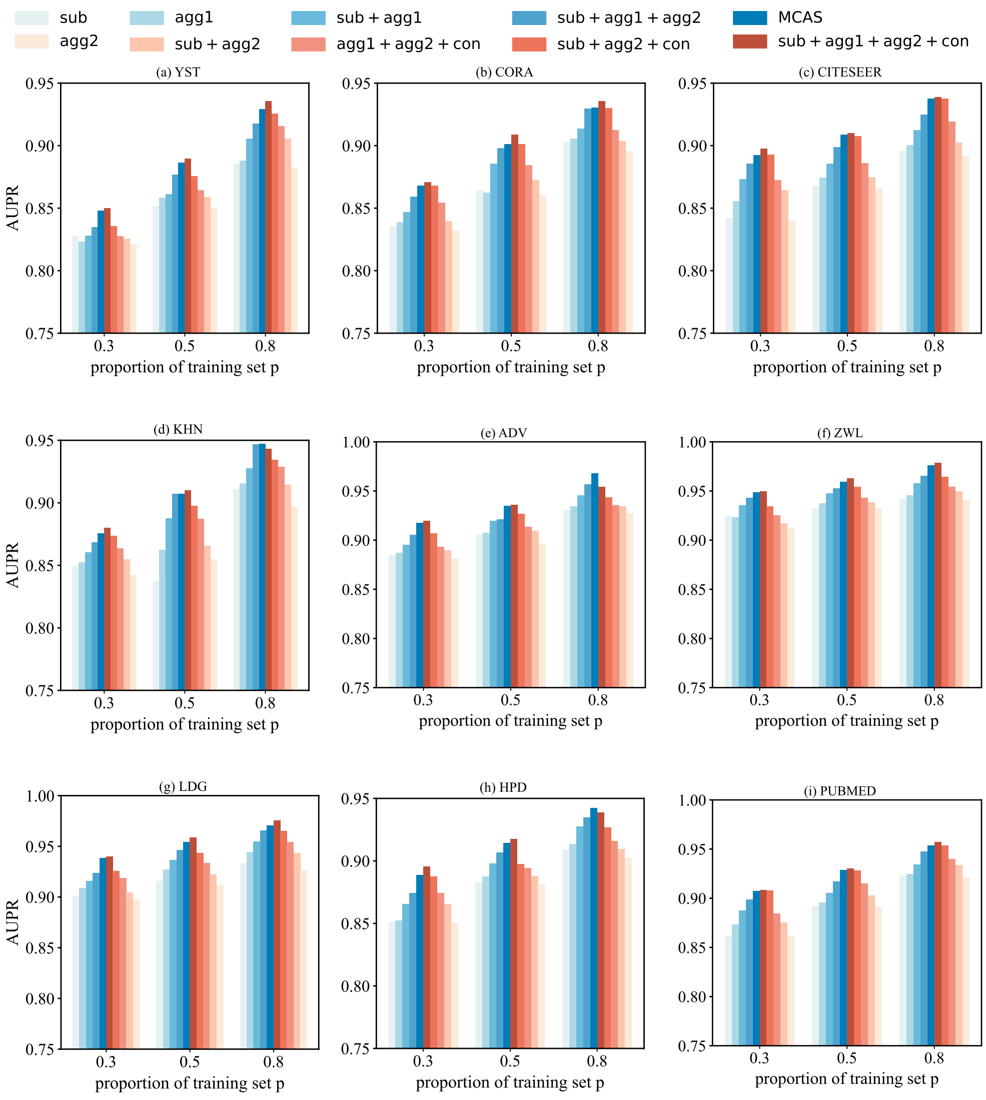


Fig. 5 Results of the ablation experiments on AUPR for nine datasets. The x-axis ranges from 0.3 to 0.5 to 0.8, indicating the training proportion, while the y-axis represents AUPR value

the MCAS model. The reason is that in these sparse networks, after subgraph extraction, the information contained in twice-aggregated subgraphs is nearly identical to that in once-aggregated subgraphs, resulting in no significant improvement in performance.

5.3.4 Hyperparameters Analysis (Q4)

In this section, we delve into the impact of two crucial hyperparameters, the temperature parameter τ in the contrastive loss and the weight coefficient α in the overall loss, on the

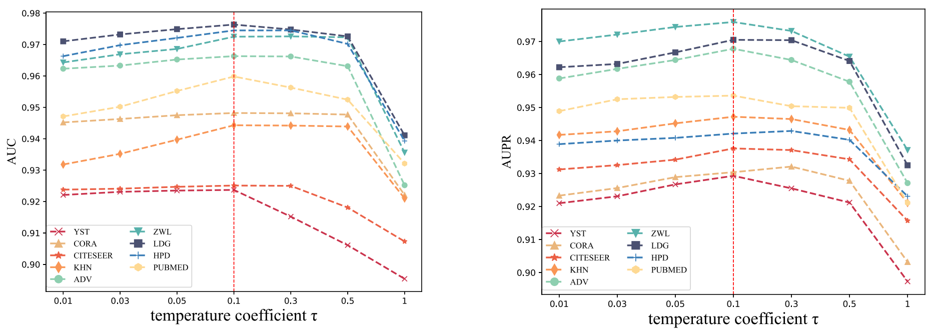


Fig. 6 Sensitivity analysis of temperature parameters τ and comparison of AUC and AUPR performance of MCAS

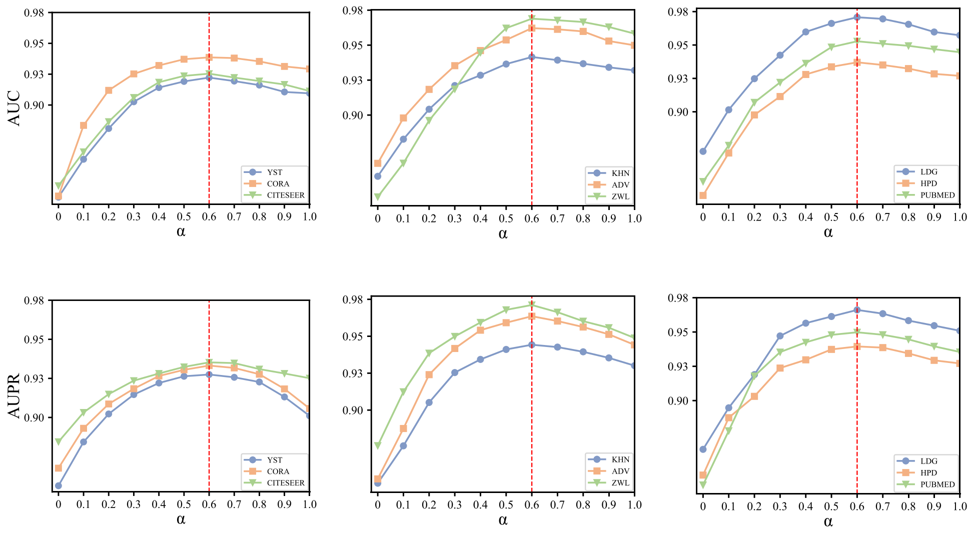


Fig. 7 Sensitivity analysis of parameters α and comparison of AUC and AUPR performance for MCAS

performance of our model in link prediction tasks. Through a series of experiments, we have conducted an in-depth analysis of the model's sensitivity to these parameters.

For the temperature parameter τ in (14), we study the impact of the τ by adjusting the Normalized Temperature-scaled Cross-entropy loss (NT-Xent) [44], where we examine the impact of the temperature coefficient τ. Figure 6 presents the AUC and AUPR results of the model on nine datasets with different values of the temperature coefficient τ. Specifically, τ takes values of 0.01, 0.03, 0.05, 0.1, 0.3, 0.5, and 1. As shown in Fig. 6, the model's performance gradually improves as τ increases from 0.01 to 0.1. This is likely because, with a larger temperature coefficient, the model pays less attention to difficult negative samples, avoiding excessive focus on those hard-to-distinguish negative samples, thus making the learning process more balanced. However, as τ increases further beyond 0.1, the model's performance begins to decrease. The results suggest that an optimal value of τ around 0.1 provides the best balance between learning efficiency and model performance across datasets.

In the MCAS method, for the weight coefficient α in (15), we employ a stepwise experimental approach to fine-tune the hyperparameter α in order to balance the weights between supervised loss and self-supervised loss. Figure 7 illustrates the model's AUC and AUPR performance on nine datasets under different values of α. As shown in Fig. 7, the models performance gradually improves as increases, reaching an optimal value of 0.6 on nearly all datasets. The AUC and AUPR metrics indicates a decreasing trend in performance across all datasets when α exceeds 0.6. The results indicate that the self-supervised multi-scale subgraph module makes a significant contribution and plays a crucial role.

6 Conclusion

This work proposes a novel link prediction framework MCAS, based on multi-scale contrastive learning with aggregated subgraph. From a multi-scale modeling perspective, MCAS aims to exploit complementary information within networks by exploring subgraph at different scales. It leverages graph contrastive learning to enhance the representation capability of these subgraph, minimizing information loss during the aggregation process. Experimental results demonstrate that MCAS achieves excellent accuracy and robustness across nine datasets.

However, the current research only considers structural redundancy during the aggregation process, overlooking the similarity information encoded in node attributes. Future work could delve deeper into integrating node attributes with structural information to improve the aggregation ability of models in attributed graphs. Additionally, further research should focus on the scalability of the model and reducing its complexity.

Acknowledgements This work is partly supported by the National Natural Science Foundation of China (No. 62366030, 62162040), the Gansu Provincial Natural Science Foundation (No.23JRRA8222), the Higher Education Innovation Fund project of Gansu (No. 2022A-022, 24CXGA046), the Open Project of Key Laboratory of Linguistic and Cultural Computing Ministry of Education (No.KFKT202304), the Gansu Provincial Science Fund for Distinguished Young Schol (No.23JRRA766).

Author Contributions Yabing Yao: Methodology and Project administration. Pingxia Guo: Conducting experiment and Writing. Zhiheng Mao: Software and Data analysis. Ziyu Ti: Data curation and Graphing. Yangyang He: Data collection. Fuzhong Nian: Project administration. Ruisheng Zhang: Device support. Ning Ma: Project administration.

Funding This work is supported by the following funding sources: the National Natural Science Foundation of China (No. 62366030, 62162040), the Gansu Provincial Natural Science Foundation (No. 23JRRA8222), the Higher Education Innovation Fund project of Gansu (No. 2022A-022, 24CXGA046). the Open Project of Key Laboratory of Linguistic and Cultural Computing Ministry of Education (No.KFKT202304), the Gansu Provincial Science Fund for Distinguished Young Schol (No.23JRRA766).

Data Availability The dataset analyzed during the experiments conducted in this paper can be obtained at the following URL: https://noesis.ikor.org/datasets/link-prediction. The source code of our method is publicly available at: https://github.com/yabingyao/MCAS4LinkPrediction.

Declarations

Conflict of interest We declare that there are no Conflict of interest in this work. We are committed to conducting our research with the utmost integrity and adhering to the highest ethical standards.

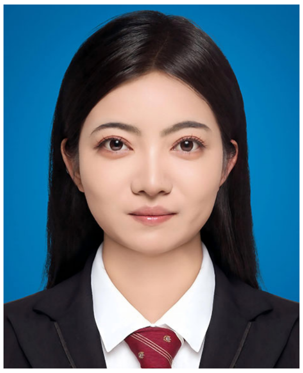
Ethical and informed consent for data used In our research, we place a high emphasis on ethical practices and informed consent in data acquisition and utilization. The datasets used are sourced from publicly accessible platforms https://noesis.ikor.org/datasets/link-prediction. To safeguard the privacy of research subjects, we strictly adhere to anonymization principles during data processing, thoroughly removing any information that could potentially reveal personal identities. Given that the data used are from public datasets, we rigorously comply with the ethical guidelines and legal requirements stipulated by the data sources. Additionally, we follow universally recognized research ethics principles to ensure compliance throughout the data collection, storage, and usage processes.

References

1. Wu H, Song C, Ge Y, Ge T (2022) Link prediction on complex networks: an experimental survey. Data Scie Eng 7(3):253–278
2. Zhou T (2021) Progresses and challenges in link prediction. Iscience 24(11)
3. Gao Z, Rezaeipanah A (2023) A novel link prediction model in multilayer online social networks using the development of katz similarity metric. Neural Process Lett 55(4):4989–5011
4. Nasiri E, Berahmand K, Rostami M, Dabiri M (2021) A novel link prediction algorithm for protein-protein interaction networks by attributed graph embedding. Comput Biol Med 137:104772
5. Cui Z, Henrickson K, Ke R, Wang Y (2019) Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. IEEE Trans Intell Transp Syst 21(11):4883–4894
6. Kumar A, Singh SS, Singh K, Biswas B (2020) Link prediction techniques, applications, and performance: a survey. Physica A Stat Mech Appl 553:124289
7. Arrar D, Kamel N, Lakhfif A (2024) A comprehensive survey of link prediction methods. J Supercomput 80(3):3902–3942
8. Yao Y, Cheng T, Li X, He Y, Yang F, Li T, Liu Z, Xu Z (2023) Link prediction based on the mutual information with high-order clustering structure of nodes in complex networks. Physica A Stat Mech Appl 610:128428
9. Wang Z, Li W, Su H (2021) Hierarchical attention link prediction neural network. Knowl-Based Syst 232:107431
10. Kumar, S., Mallik, A., Panda, B (2022) Link prediction in complex networks using node centrality and light gradient boosting machine. In: Proceedings of the 31th international conference on World Wide Web(WWW). pp 2487–2513
11. Zhang M, Chen Y (2018) Link prediction based on graph neural networks. Adv Neural Inform Process Syst 31
12. Louis, P., Jacob, S.A., Salehi-Abari, A (2022) Sampling enclosing subgraphs for link prediction. In: Proceedings of the 31st ACM international Conference on Information & Knowledge Management(CIKM). pp 4269–4273
13. Chien E, Chang W-C, Hsieh C-J, Yu H-F, Zhang J, Milenkovic O, Dhillon IS (2021) Node feature extraction by self-supervised multi-scale neighborhood prediction. arXiv:2111.00064
14. Guo Z, Wang F, Yao K, Liang J, Wang Z (2022) Multi-scale variational graph autoencoder for link prediction. In: Proceedings of the fifteenth ACM international conference on web search and data mining. pp 334–342
15. Zhang G, Ye T, Jin D, Li Y (2023) An attentional multi-scale co-evolving model for dynamic link prediction. Proc ACM Web Conf 2023:429–437
16. Cai L, Ji S (2020) A multi-scale approach for graph link prediction. Proc AAAI Conf Artif Intell 34:3308–3315
17. Newman ME (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64(2):025102
18. Adamic LA, Adar E (2003) Friends and neighbors on the web. Social Netw 25(3):211–230
19. Zhou T, Lü L, Zhang Y-C (2009) Predicting missing links via local information. Eur Phys J B 71:623–630
20. Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43
21. Jeh G, Widom J (2002) 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
22. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data Mining. pp 701–710
23. Cai L, Wang J, He T, Meng T, Li Q (2018) A novel link prediction algorithm based on deepwalk and clustering method. In: Journal of physics: conference series. IOP Publishing, p 012131
24. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on World Wide Web(WWW). pp 1067–1077
25. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp 855–864
26. Nasiri E, Berahmand K, Li Y (2023) Robust graph regularization nonnegative matrix factorization for link prediction in attributed networks. Multimed Tools Appl 82(3):3745–3768
27. Yao Y, He Y, Huang Z, Xu Z, Yang F, Tang J, Gao K (2024) Deep non-negative matrix factorization with edge generator for link prediction in complex networks. Appl Intell 54(1):592–613
28. Zhang P, Chen J, Che C, Zhang L, Jin B, Zhu Y (2023) Iea-gnn: Anchor-aware graph neural network fused with information entropy for node classification and link prediction. Inf Sci 634:665–676
29. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
30. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inform Process Syst 30
31. Jiawei E, Zhang Y, Yang S, Wang H, Xia X, Xu X (2024) Graphsage++: Weighted multi-scale gnn for graph representation learning. Neural Process Lett 56(1):24

32. Shi L, Hu B, Zhao D, He J, Zhang Z, Zhou J (2024) Structural information enhanced graph representation for link prediction. Proc AAAI Conf Artif Intell 38:14964–14972
33. Tan Q, Zhang X, Liu N, Zha D, Li L, Chen R, Choi S-H, Hu X (2023) Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In: Proceedings of the sixteenth ACM international conference on Web Search and Data Mining(WSDM). pp 625–633
34. Liu Z-Y, Liu J-W, Zuo X, Hu M-F (2021) Multi-scale iterative refinement network for rgb-d salient object detection. Eng Appl Artif Intell 106:104473
35. Zhang R, Chen C, Peng J (2024) Multi-scale graph feature extraction network for panoramic image saliency detection. Vis Comput 40(2):953–970
36. Shi C, Pun C-M (2019) Adaptive multi-scale deep neural networks with perceptual loss for panchromatic and multispectral images classification. Inf Sci 490:1–17
37. Pang Y, Zhao X, Zhang L, Lu H (2020) Multi-scale interactive network for salient object detection. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern recognition(CVPR). pp 9413–9422
38. Rozemberczki B, Allen C, Sarkar R (2021) Multi-scale attributed node embedding. J Complex Netw 9(2):014
39. Duan J, Wang S, Zhang P, Zhu E, Hu J, Jin H, Liu Y, Dong Z (2023) Graph anomaly detection via multi-scale contrastive learning networks with augmented view. Proc AAAI Conf Artif Intell 37:7459–7467
40. Chen J, Qin D, Hou D, Zhang J, Deng M, Sun G (2022) Multiscale object contrastive learning-derived few-shot object detection in vhr imagery. IEEE Trans Geosci Remote Sens 60:1–15
41. Wu L, Huang Y, Tan C, Gao Z, Hu B, Lin H, Liu Z, Li SZ (2024) Psc-cpi: Multi-scale protein sequence-structure contrasting for efficient and generalizable compound-protein interaction prediction. arXiv:2402.08198
42. Zhang Z, Sun S, Ma G, Zhong C (2023) Line graph contrastive learning for link prediction. Pattern Recogn 140:109537
43. Zhang H, Ren Y, Fu L, Wang X, Chen G, Zhou C (2023) Multi-scale self-supervised graph contrastive learning with injective node augmentation. IEEE Trans Knowl Data Eng 36(1):261–274
44. Oord A, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv:1807.03748
45. Liu L, Xie Q, Wen W, Zhu J, Peng M (2024) Edge contrastive learning for link prediction. Inform Process Manag 61(6):103847
46. Peri S, Navarro JD, Amanchy R, Kristiansen TZ, Jonnalagadda CK, Surendranath V, Niranjan V, Muthusamy B, Gandhi T, Gronborg M et al (2003) Development of human protein reference database as an initial platform for approaching systems biology in humans. Genome Res 13(10):2363–2371
47. Cai L, Li J, Wang J, Ji S (2021) Line graph neural networks for link prediction. IEEE Trans Pattern Anal Mach Intell 44(9):5103–5113
48. Massa P, Salvetti M, Tomasoni D (2009) Bowling alone and trust decline in social network sites. In: 2009 eighth IEEE international conference on dependable, autonomic and secure computing. IEEE, pp 658–663
49. Li B, Zhou M, Zhang S, Yang M, Lian D, Huang Z (2022) Bsal: a framework of bi-component structure and attribute learning for link prediction. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval. pp 2053–2058
50. Liu X, Li X, Fiumara G, De Meo P (2023) Link prediction approach combined graph neural network with capsule network. Expert Syst Appl 212:118737
51. Hinton GE, Krizhevsky A, Wang SD (2011) Transforming auto-encoders. In: Artificial neural networks and machine learning–ICANN 2011: 21st International conference on artificial neural networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21. Springer, pp 44–51
52. Jiao Y, Xiong Y, Zhang J, Zhang Y, Zhang T, Zhu Y (2020) Subgraph contrast for scalable self-supervised graph representation learning. In: 2020 IEEE International Conference on Data Mining (ICDM). IEEE, pp 222–231
53. Hanley JA, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143(1):29–36
54. Davis J, Goadrich MH (2006) The relationship between precision-recall and roc curves. Proceedings of the 23rd international conference on Machine learning
55. Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE 10(3):0118432
56. Ran Y, Xu X-K, Jia T (2024) The maximum capability of a topological feature in link prediction. PNAS Nexus 3(3):113

Yabing Yao is currently an associate professor at the School of Computer and Communication, Lanzhou University of Technology. He received the Ph.D. degree in the School of Information Science and Engineering at Lanzhou University in 2017. His work has focused on link prediction in complex networks. His research interests include machine learning on graphs and network science.



Pingxia Guo enrolled in Lanzhou Institute of Technology in 2018 to study Software Engineering and obtained a Bachelor's degree in Engineering in 2022. In the same year, she entered the School of Computer Science and Communication at Lanzhou University of Technology to pursue a Master's degree, with her main research focus on link prediction and multi-scale representation learning.

Zhiheng Mao received the bachelor's degree from the Lanzhou University of Technology, Lanzhou, China, in 2021. He is currently pursuing the M.S. degree with the School of Computer and Communication, Lanzhou University of Technology, Lanzhou, China. His research interests include link prediction and graph neural architecture search.

Ziyu Ti received her B.S. computer science and technology from University of Xianyang Normal in 2021. She is currently a M.S. student in the School of Computer and Communication at Lanzhou University of Technology. Her research interests include link prediction and graph self supervised learning.

Yangyang He received the master's degree from the Lanzhou University of Technology, Lanzhou, China, in 2024. He is currently pursuing a PhD in the School of Computer Science and Technology, Beijing Jiaotong University, Beijing, China. His research interests include complex network analysis, link prediction and high-performance computer architectures.

Fuzhong Nian received his B.S. degree in Physics from Northwest Normal University, Lanzhou, China, in 1998; and M.S. degree in Control Theoretics & Engineering from Lanzhou University of Technology, Lanzhou, China, in 2004, respectively. He received the Ph.D degree at Dalian University of Technology. He is currently a professor at the School of Computer and Communication, Lanzhou University of Technology. His research interests include nonlinear dynamics and control, complex networks and systems, neural networks.

Ruisheng Zhang received the B.S. degree from the Department of Mathematics, Lanzhou University, in 1983, and the M.A. and Ph.D. degrees from the Department of Chemistry, Lanzhou University, in 1990 and 1996, respectively. He is currently a Professor and Ph.D. Supervisor with the School of Information Science and Engineering, Lanzhou University, where he is also the Director of the Institute of Computer Architecture. His main research fields are network science, service computing, chemo-informatics, and bioinformatics.

Ning Ma is the doctoral supervisor of Language Intelligence and Cultural Computing in Northwest University for Nationalities, and the postgraduate supervisor of Computer Science and Technology and Artificial Intelligence in Computer technology. His research interest is machine learning and natural language processing.

Authors and Affiliations

Yabing Yao[1] · **Pingxia Guo**[1] · **Zhiheng Mao**[1] · **Ziyu Ti**[1] · **Yangyang He**[2] · **Fuzhong Nian**[1] · **Ruisheng Zhang**[3] · **Ning Ma**[4]

✉ Yabing Yao
yaoyabing@lut.edu.cn

Pingxia Guo
guopingxia2024@163.com

Zhiheng Mao
mzh123452021@163.com

Ziyu Ti
tzy97666@163.com

Yangyang He
yangyanghe2023@163.com

Fuzhong Nian
gdnfz@lut.edu.cn

Ruisheng Zhang
zhangrs@lzu.edu.cn

Ning Ma
maning@xbmu.edu.cn

[1] School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China

[2] School of Computer Science and Technology, Beijing Jiaotong University, Beijing 100044, China

[3] School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China

[4] Key Laboratory of Linguistic and Cultural Computing Ministry of Education, Northwest Minzu University, Lanzhou 730030, China