Physica A 646 (2024) 129874

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

# Relationship updating network with contrastive learning

Pengyu Zhang [a], Yong Zhang [a,*], Xinglin Piao [a], Yongliang Sun [b], Baocai Yin [a]

[a] Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Faculty of Information Technology, Beijing Institute of Artificial Intelligence, Beijing University of Technology, Beijing, 100124, China
[b] China Electronics Technology Group, Beijing, China

ARTICLE INFO

*Keywords:*
Graph analysis
Graph convolutional networks
Semi-supervised
Classification graph neural networks

ABSTRACT

In Graph Neural Networks (GNNs), a common feature across many datasets is the Power-law Distribution of node degrees, where most nodes exhibit few connections, contrasting with a small fraction that possesses a high number of links. This difference often introduces training instability and compromises performance on tasks like node classification, particularly for low-degree nodes. To tackle these challenges, we introduce **RUNCL**: **R**elationship **U**pdating **N**etwork with **C**ontrastive **L**earning, a novel model designed to ensure that the model learns more accurate node features, especially the features of low-degree nodes. Specifically, RUNCL comprises a graph generation module that generates different neighborhood information graphs based on the node feature graphs. The optimal graph selection module selects the neighborhood information graph that best reflects the relationship between nodes and a contrastive learning module to learn more accurate node embeddings by contrasting positive and negative samples. We evaluate the performance of RUNCL on six datasets, and the experimental results demonstrate its effectiveness. The model exhibited an improvement of 2.5% in the testset. Moreover, the model's performance boosted to 6% when the testset only included low-degree nodes. The implementation and data are made available at https://github.com/pengyu-zhang/RUNCL-Relationship-Updating-Network-with-Contrastive-Learning.

## 1. Introduction

Graphs are a kind of data structure that models a set of objects (nodes) and their relationships (edges) [1]. Graphs are essential for effectively representing network structures, such as bibliographic [2] and social networks [3], as well as biomedical networks [4]. With a more accurate node relationship graph, Graph Neural Networks (GNNs) can capture complex relationships and rich semantics that exist in real-world situations [5]. However, relationships in real-world graph data are often incomplete and contain errors, making message passing in graph networks difficult. This will lead to a degradation in the performance of GNN models, especially the low-degree nodes in graph data. Low-degree nodes have fewer connections for message passing, making it challenging for the model to capture and exploit the relational intricacies effectively. For instance, in graph data, if node A has hundreds of edges while node B has none at all, then the ability of the model to accurately learn and generalize relationships involving node B is substantially hindered. This imbalance in the number of edges influences the model's learning process, potentially causing biases and affecting the overall predictive accuracy and robustness of the GNN model.

Hence, graph structure learning, which uses raw graph data to estimate graph quantities more accurately, has emerged as a promising solution for graph representation learning [6]. This approach can address the issue of false links in graph data and improve the accuracy of downstream tasks by providing more accurate node relationships. However, many current methods parameterize

* Corresponding author.
*E-mail address:* zhangyong2010@bjut.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.physa.2024.129874
Received 9 December 2023; Received in revised form 16 April 2024
Available online 4 June 2024

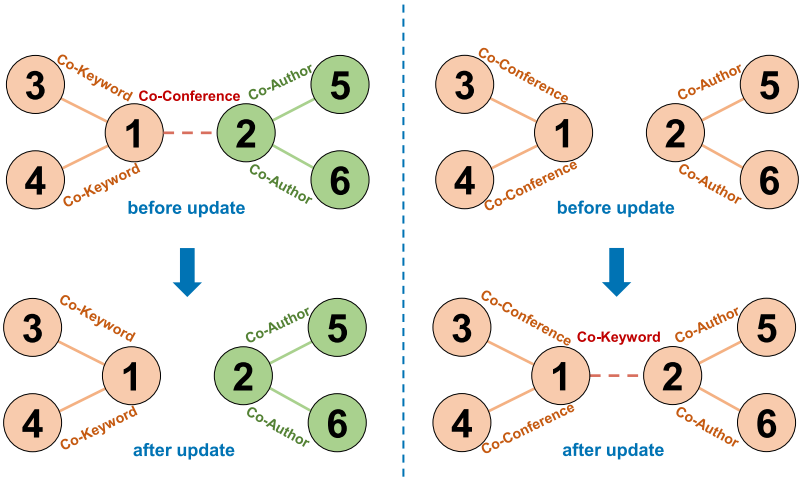



**Fig. 1.** An example of relationship update.

each edge locally, resulting in less accurate node relationships. For instance, missing data in authors' publication records in the citation network can lead to erroneous connections between authors in the original relationships, making it challenging to predict research interests, as illustrated in Fig. 1. This introduces more noise into the graph networks, particularly for low-degree nodes. Additional noise disproportionately affects their limited connections, making it even more challenging to capture and represent the underlying relationships accurately.

To tackle this challenge, we hypothesize that not only do existing relationships influence node embeddings, but newly constructed relationships can also impact the embeddings. Typically, when there are more links between nodes, these nodes profoundly influence each other, and the model can more easily learn local information. However, if two nodes are distantly placed with no links between them, it becomes challenging for them to influence each other, even if their features are similar—this phenomenon is particularly noticeable in low-degree nodes. In graph structure learning, creating previously non-existent links can allow the model to learn local and global information. For example, in a citation network, if author A has hundreds of edges, and author B is an isolated node with no edges, but A and B have similar research interests, constructing a link between A and B could enable the model to learn richer information for the isolated node B.

To better uncover the local and global information of low-degree nodes and isolated nodes, we have also introduced a graph contrastive learning module which has been widely applied in computer vision and natural language processing [7–11]. Identifying more negative samples of low-degree and isolated nodes allows for more refined discrimination between the essential connective features and the incidental or noisy associations. This can enhance the robustness and generalizability of the model, enabling it to better capture and represent the nodes' intrinsic structural and attributive properties despite their sparse connectivity or isolated nature in the graph, improving the overall effectiveness and accuracy of the graph neural network in various downstream tasks [12,13].

We introduce **RUNCL**: **R**elationship **U**pdating **N**etwork with **C**ontrastive **L**earning. We use the graph generation module to effectively use the node features in generating the node feature graph. The optimal graph construction module considers the neighborhood information of the nodes in the graph and the available multiview information. By treating the multiview information as observations of the optimal graph, we avoid bias that may arise in the process of selecting the best graph. Finally, in the collaborative contrastive optimization module, we leverage the node feature graph and the node relationship graph as two views to monitor each other collaboratively. The main contributions can be summarized as follows:

- We propose a novel model, RUNCL, which addresses the bias issue by injecting multi-order neighborhood information. The model ensures that contrastive learning between different views can supervise each other and generate negative samples with high quality.
- Since RUNCL can combine graph structure learning and cross-view contrastive learning. As a result, our model can capture high-level factors in node embedding via contrastive learning, enabling its application to real-world datasets with very few labels.
- We evaluate the performance of RUNCL on six publicly available datasets and compare it to the current state-of-the-art baselines to demonstrate its effectiveness. The improvement verifies the rationality of the different modules across the datasets.

## 2. Related work

In this part, we examine previous studies on graph structure learning and contrastive learning, and highlight their relevance to the current research.

### 2.1. Graph Neural Network

The field of Graph Neural Networks (GNNs) aims to develop methods that enable deep neural networks to process and analyze graph-structured data [14]. In recent years, researchers have been exploring ways to generalize traditional convolutional neural networks to the domain of graphs [15], and there are two main approaches to achieve this: spectral and non-spectral methods. These approaches seek to leverage the unique structure of graphs to enable more effective analysis and prediction of complex systems that can be represented as graphs. Spectral approaches process graphs using a spectral representation. For instance, based on [16], Convolutional neural networks (CNNs) have been extended to signals defined on domains that do not have a translation group. This generalization allows for the analysis of data with more complex structures, such as graphs. By adapting convolutional operations to these structures, the resulting models can extract meaningful features and achieve high performance on various tasks, such as image and graph classification, object recognition, and natural language processing. While [17] has been centered on expanding the capabilities of convolutional neural networks beyond their traditional use on regular grids of low dimensions. On the other hand, [18] employed an alternative technique to acquire node embeddings, which relies on a localized first-order estimation of spectral graph convolutions. This allows for the processing of large-scale graph data, by approximating the convolutional operations and reducing the computational complexity. In addition, [19] introduced a general approach for extracting locally connected regions from graphs. By doing so, the model can learn graph structure and node features for each node. Moreover, [20] proposed a method for generating embeddings for nodes in a graph. This method allows for the inductive learning of node representations, which means that the model can generalize to unseen nodes that were not present during training. While [21] introduced a new type of graph neural network, which leverages gated recurrent units to capture the information flow between nodes in a graph. This method involves applying a set of recurrent units to update each node based on the features of its neighboring nodes. [22] proposed an alternative Long Short-Term Memory (LSTM) structure for processing sequential data such as sentences. This approach utilizes recurrent steps to enable local and global information exchange between words in parallel. Furthermore, [23] studied the attention mechanism in graph neural networks (GNNs) and introduced a new method for incorporating attention into the propagation step. This approach involves computing attention coefficients for each neighbor of a target node, which reflect their importance in the computation of the target node's representation. Building on this work, [24] proposed a novel approach for modeling heterogeneous graphs using a graph attention network. Their method incorporates both node-level and semantic-level attention, allowing for more effective modeling of the graph's structure and content. [25] suggested a refined propagation method that utilizes personalized PageRank and exploits the connection between graph convolutional networks and PageRank. This approach involves computing a personalized PageRank score for each node in the graph, which reflects its importance in the graph structure.

In general, GNN models heavily rely on input graphs that are treated as ground truth. However, this approach has a limitation when dealing with uncertain and incomplete graph structures. This can be a significant challenge in real-world scenarios where graph data is often noisy and incomplete due to missing or unreliable information. Which highlights the need for more robust and adaptive GNN models that can deal with uncertain and incomplete graphs.

### 2.2. Graph structure learning

Graph structure learning is a relatively new research topic that has gained attention in recent years, with several works devoted to it [26]. Many of these works have combined graph structure learning and graph neural networks to improve the performance of downstream tasks. For instance, the Bayesian GCNN [27] uses the adjacency matrix of the graph to combine features for joint optimization. This approach enables the model to capture uncertainty in the graph structure and generate robust predictions. Meanwhile, LDS [12] proposes a framework that can optimize and learn the structure of the GNN in an end-to-end manner. This method aims to address the challenge of incomplete graphs by jointly inferring the graph structure and the model parameters. Pro-GNN [28] proposes a low-rank sparsity regularization for adversarial reconstruction. In this approach, the original graph is reconstructed using a combination of low-rank sparsity and feature smoothing regularization techniques. The goal is to mitigate the negative effects of adversarial structure in the graph by producing a more robust and accurate representation. GEN [29] generates community structure graphs and observation models to fit the mechanisms of graph neural networks. For unsupervised graph structure learning, SUBLIME [30] generates a target from the original data and maximizes the agreement between the two graphs. Also, the CoGSL model [31] extracts two basic views from the original graph and fuses the estimated views into the final view.

In contrast to the aforementioned graph structure learning methods, our goal is to explore graphs learned across different views that contain more than one type of node and relation.

### 2.3. Contrastive learning

Contrastive learning-based models have achieved great success and have attracted numerous works by comparing positive and negative samples [32,33]. DGI [34] constructs local patches and global summaries as positive pairs and uses infomax theory to contrast them. HeCo [35] is based on heterogeneous graphs and captures both local and high-order structures simultaneously, allowing cross-view contrastive learning to extract both positive and negative examples. These models are often used in sentiment analysis, where the positive and negative examples are represented as vectors. Recently, TIFA-GCL [36] found that the promotion of graph contrastive learning comes mainly from nodes with less annotated information and focuses on the uneven distribution of annotated information. DualGraph [37] investigates semi-supervised graph classification and proposes a framework to better use unlabeled graphs. To reduce overfitting due to too few labels, GraphSSL [38] improves the performance of graph classification

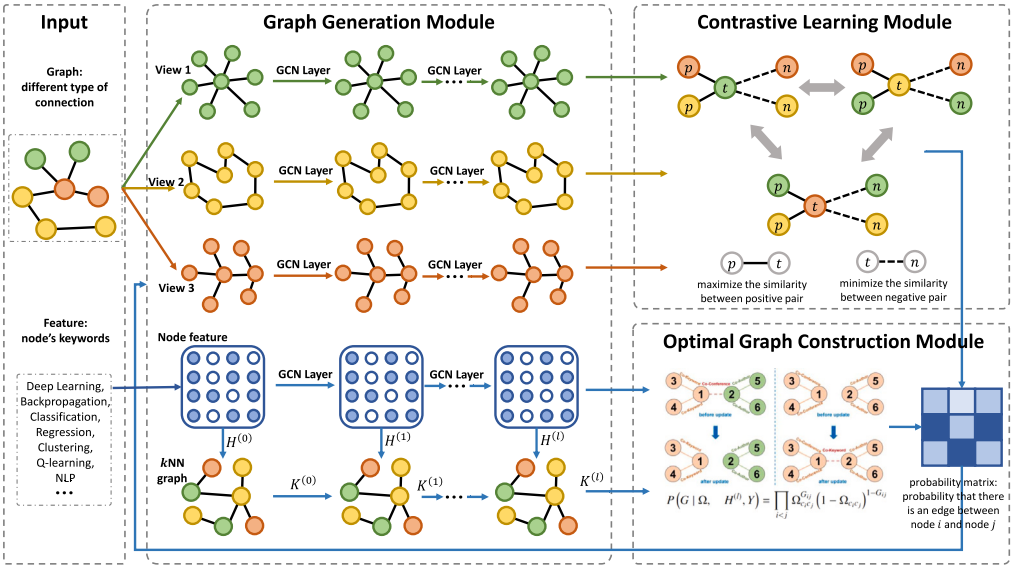



**Fig. 2.** Overall framework of our proposed model RUNCL, where takes three node relations as an example.

tasks by performing data augmentation on the original graphs. AD-GCL [39] proposes a method to enhance the ability of graph convolutional networks to resist adversarial attacks.

However, most contrastive learning models rely on the input graphs as ground truth information, which poses a challenge when dealing with uncertain or incomplete graph structures. This can limit the ability of the models to generalize to new, unseen graph structures.

## 3. The proposed model: RUNCL

In this section, we will provide a detailed explanation of RUNCL, a neural network designed for semi-supervised node classification with contrastive learning. The complete structure of the model is illustrated in Fig. 2.

### 3.1. Problem formulation

The graph $G = (V, E)$ is utilized in this study, where $V$ and $E$ represent the sets of nodes and edges, respectively. The semi-supervised node classification task requires the association of labels with only a subset of the nodes, specifically $V_L = \{v_1, v_2, \ldots, v_l\}$, denoted as the labeled nodes. These labeled nodes are assigned corresponding labels in the set $Y_L = \{y_1, y_2, \ldots, y_l\}$, where $y_i$ represents the label of node $v_i$. The remaining nodes in the graph are unlabeled.

### 3.2. Overview

Our proposed model comprises three main modules: the **graph generation module**, which generates a set of candidate graphs by sampling from a pre-defined distribution; the **optimal graph construction module**, which constructs an optimal graph by selecting the best candidate graph based on a pre-defined criterion; and the **contrastive learning module**, which learns node embeddings using contrastive learning on the constructed optimal graph. (1) The graph generation module's inputs are divided into two parts. The first part of the input is node features extracted from the original data and the $k$NN graph constructed based on the node features. The second part of the input is node relationship graphs, which ensures that our model can comprehensively capture the relationships between the nodes. (2) We used the optimal graph construction module to select the $k$NN graph that best reflects the relationships between nodes. (3) We employ contrastive learning across the node relationship graphs. To improve the performance of the model, it is necessary to redefine the positive samples of a node and devise a specific optimization strategy.

Our model alleviates the problem of missing or incorrect node relationships in three ways. (1) The model uses the original and newly generated node labels to complement the missing node relationships. Then, RUNCL can construct new links between nodes and remove incorrect links. (2) Multiple $k$NN graphs are used for node relationship inference, avoiding the noise in the original graph or a particular $k$NN graph that may affect downstream tasks. Moreover, by observing multiple $k$NN graphs, the connection can be reestablished based on the features of the two nodes. (3) With the iterative optimization of the model, the positive and negative samples generated by the contrastive learning module can generate more accurate positive and negative samples as labels, which finally learn more accurate node embeddings.

### 3.3. Graph generation module

In the graph generation module, we aim to refine each node's representation by incorporating its neighbors' information. We start by inputting the original node connections, denoted as $A$, and node features, represented as $X$, into the GCN (Graph Convolutional Network) layer. This step forms the initial representation $H$. The workings of the $k$-th layer in this process can be expressed mathematically as follows:

$$H^{(k)} = \mathrm{ReLU}\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(k-1)}W^{(k)}\right). \tag{1}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix with added self-connections, $I_N$ is the identity matrix signifying self-connections to preserve a node's features, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $W^{(k)}$ is the weight matrix for the $k$-th layer, and $ReLU$ denotes the activation function. The matrix $H^{(k)}$ refers to the node representations in the $k$th layer, where $H^{(0)} = X$ is the node feature matrix. The final layer of the GCN utilizes row-wise softmax. We optimize the GCN parameters $\Theta = \left(W^{(1)}, W^{(2)}, \ldots, W^{(l)}\right)$ using gradient descent. This recursive update enriches the node representations by learning from the local neighborhood structures.

In order to produce the set $K$, we adjust the GCN parameters $\Theta$ to create a series of $k$-nearest neighbor graphs, represented as $\{K^{(0)}, K^{(1)}, \ldots, K^{(l)}\}$. These are formed using the node representations $H = \{H^{(0)}, H^{(1)}, \ldots, H^{(l)}\}$, where each $K^{(i)}$ signifies the adjacency matrix for the $k$-NN graph derived from $H^{(i)}$. The original node relationships $A$ are also factored in as they provide critical baseline observations for the most informative neighborhood graph. As a result, we merge $A$ with our $k$-NN graphs to compile a comprehensive set of neighborhood information graphs, denoted as $K = \{A, K^{(0)}, K^{(1)}, \ldots, K^{(l)}\}$, to enable a more robust graph structure for our model.

### 3.4. Optimal graph construction module

To optimize the understanding of node relationships within our model, we have amassed a comprehensive collection of neighborhood information graphs, denoted as $K$. By applying Bayesian inference to these graphs, we calculate the posterior distribution of graph structures. This process allows us to determine that the observation graph most accurately represents the underlying node relationships. Consequently, by examining these refined neighborhood information graphs, our model can achieve more precise observational outcomes.

Our model incorporates the Stochastic Block Model (SBM), a prominent framework used in community detection that helps in graph modeling by considering the probabilities of connections between nodes. To estimate the most informative neighborhood graph $K$, the model presupposes its optimality. Following [29], we use the probability distribution $P(G|\Omega, H^{(l)}, Y)$ to derive the best graph $G$. The SBM operates on the principle that the likelihood of a link between any two nodes depends on the communities they belong to, without influence from other nodes or factors. In this context, $\Omega$ represents the model's parameters, detailing the likelihood of edges forming within and across communities. For example, if node $v_i$ is part of community $c_i$ and node $v_j$ is part of community $c_j$, the probability of an edge between them is denoted as $\Omega_{c_i c_j}$. Thus, $\Omega$ quantifies the connection probabilities within the same community versus between different communities. The probability of creating a particular graph $G$ can be calculated with these parameters $\Omega$, alongside the node predictions $H$ and labels $Y$:

$$P\left(G \mid \Omega, \quad H^{(l)}, Y\right) = \prod_{i<j} \Omega_{c_i c_j}^{G_{ij}} \left(1 - \Omega_{c_i c_j}\right)^{1-G_{ij}}. \tag{2}$$

In our method, we assume that the community designations $c_i$ and $c_j$ for nodes are independently assigned. When considering the graph $G$, the probability of an edge being present between any two nodes, say $v_i$ and $v_j$, depends solely on the probability $\Omega_{c_i c_j}$. This probability is directly linked to the community categories $c_i$ and $c_j$ that the nodes belong to, and it is not affected by any extraneous factors. The idea behind this approach is that we can use the labels of a node to replace its community identification, thereby improving the model's accuracy.

In our semi-supervised node classification framework, we incorporate labeled and unlabeled data to enhance learning. The model's accuracy is quantified by the cross-entropy loss computed over all labeled samples $Y_L$. We aim to reduce this loss to maintain the classification model's effectiveness. The cross-entropy loss function is pivotal in this process, adjusting the model's parameters towards more accurate predictions:

$$L_y(A, X, Y_L) = -\sum_{v_i \in V} y_i \ln z_i. \tag{3}$$

This loss function is typical in the optimization process of Graph Convolutional Networks (GCN), where the goal is to fine-tune the parameters $\Theta$ using stochastic gradient descent to minimize the error.

Furthermore, to refine the accuracy of our predictions, our model leverages labels to refine the outcomes. This correction process draws from the initial model input and insights from the contrastive learning module. Further details on this integration and its impact on prediction accuracy will be discussed in the following section.

### 3.5. Contrastive learning module

In our framework, nodes from the k-nearest neighbors graph $K$ and the optimized neighborhood information graph $G$ are utilized alongside node $a_i$ from the set of relationship graphs $A^{(m)}{}_{m=1}^{M}$. These nodes are input into a multi-layer perceptron (MLP) with an Exponential Linear Unit (ELU) activation function to project them into a space where we can calculate the contrastive loss:

$$k_{i-}p = W^{(2)}ELU(W^{(1)}k_i + b^{(1)}) + b^{(2)}, \tag{4}$$

$$g_{i-}p = W^{(2)}ELU(W^{(1)}g_i + b^{(1)}) + b^{(2)}, \tag{5}$$

$$a^{(m)}i-p = W^{(2)}ELU(W^{(1)}a^{(m)}i + b^{(1)}) + b^{(2)}. \tag{6}$$

Here, $ELU$ represents an exponential linear unit, a type of non-linear activation function. The parameter sets $W^{(2)}, W^{(1)}, b^{(2)}, b^{(1)}$ are shared across different views for embedding consistency.

Contrastive loss computation requires defining positive and negative samples distinctly. Unlike computer vision where all other images might be considered negatives, in our graph context, we focus on the node's local neighborhood to select positive samples. This method leverages local structural information to improve embedding accuracy, by considering nodes with strong mutual connections as positives.

To assess the connectivity between nodes $i$ and $j$, we count the number of direct links using the function $N_i(j)$:

$$N_i(j) = \sum_{n=1}^{M} \mathbb{I}(j \in V). \tag{7}$$

Here, $\mathbb{I}()$ is an indicator function that returns 1 if the condition inside is true. We define the top $t$ connected nodes to node $i$ as positives, $P_i$, with all others as negatives, $N_i$. The loss functions for the positive and negative sample generation are then given by:

$$L_i = -\log \frac{\sum_{j\in P_i} \exp\left(\frac{\mathrm{sim}\left(k_{i\_}p, g_{j\_}p\right)}{\tau}\right)}{\sum_{k\in\{P_i\cup N_i\}} \exp\left(\frac{\mathrm{sim}\left(k_{i\_}p, g_{i-}p\right)}{\tau}\right)}, \tag{8}$$

$$L_a = -\log \frac{\sum_{j\in P_i} \exp\left(\frac{\mathrm{sim}\left(k_{i\_}p, a_{j\_}p\right)}{\tau}\right)}{\sum_{k\in\{P_i\cup N_i\}} \exp\left(\frac{\mathrm{sim}\left(k_{i\_}p, a_{i-}p\right)}{\tau}\right)}. \tag{9}$$

The function $sim$ computes cosine similarity between two vectors, and the temperature parameter $\tau$ helps prevent the model from getting stuck in local optima during training. The overall loss function of the model is the sum of the losses from the label prediction and the contrastive learning module:

$$L = L_y + L_i + L_a. \tag{10}$$

This approach fine-tunes the model's parameters to better capture the complex relationships within the graph data, aiming to enhance the predictive performance.

## 4. Experimental results and analysis

This section evaluates how effective RUNCL is in semi-supervised node classification tasks. Additionally, we investigate the underlying mechanism of RUNCL to understand how it works better. The implementation of our approach is based on the original codebase GEN[1] [29] and HeCo.[2] [35] All experiments were conducted with the following settings: Operating system: Windows 10 64-bit 21H2. CPU: AMD Ryzen 5 5600X 6-Core Processor. GPU: NVIDIA GeForce RTX 3060. Software versions: Python 3.9; Pytorch 1.10.1; Numpy 1.21.2; SciPy 1.7.3; NetworkX 2.6.3; scikit-learn 1.0.2.

Training the GEN model and our model on the Cora, Citeseer, Chameleon, and Actor datasets required approximately 20-50 min for 20 epochs. In contrast, both models needed around 120 min to complete 20 epochs for the Pubmed and Squirrel datasets.

### 4.1. Datasets

Table 1 provides a brief overview of six datasets used in our study. These characteristics are vital for understanding the structural of each dataset.

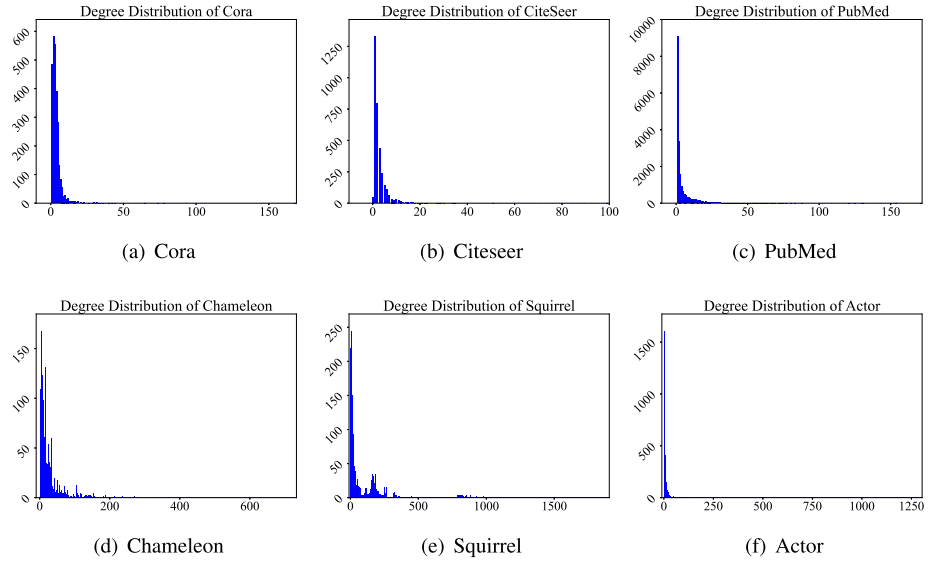



**Fig. 3.** Nodes degree distribution. The $x$-axis represents the degree, while the $y$-axis denotes the number of nodes.

**Table 1**
Statistics of the datasets.

| Dataset | Nodes | Edges | Classes | Features | Avg Degree | Max Degree | Density |
|---|---|---|---|---|---|---|---|
| **Cora** | 2708 | 5429 | 7 | 1433 | 3.90 | 168 | 0.00144 |
| **CiteSeer** | 3327 | 4732 | 6 | 3703 | 2.74 | 99 | 0.00085 |
| **PubMed** | 19717 | 44338 | 3 | 500 | 4.50 | 171 | 0.00023 |
| **Chameleon** | 2277 | 36101 | 5 | 2325 | 27.58 | 732 | 0.01213 |
| **Squirrel** | 5201 | 217073 | 5 | 2089 | 76.30 | 1904 | 0.01468 |
| **Actor** | 7600 | 33544 | 5 | 931 | 7.03 | 1303 | 0.00093 |

Additionally, we have generated degree distribution graphs for all six datasets to analyze the data structure in each dataset, as illustrated in Fig. 3. The $x$-axis represents the degree, while the $y$-axis denotes the number of nodes, offering insights into connectivity distribution across each dataset's nodes.

- **Cora**, **Citeseer** and **Pubmed** [18]: There exist three commonly used citation network datasets: Cora, Citeseer, and Pubmed. These datasets consist of papers, where each paper is represented as a node in the network, and the citation relationships between papers are represented as edges in the network. The labels for each paper correspond to academic fields, while the node features are represented as bag-of-words representations of the papers [29].
- **Chameleon** and **Squirrel** [13]: The Wikipedia dataset contains two page-page networks: Chameleon and Squirrel. In these networks, nodes correspond to web pages, and edges represent hyperlinks between the pages. Each node has a set of features that correspond to informative nouns in the page, and the label of each node represents the monthly traffic of that page. These datasets are commonly used in graph-based machine learning research as benchmarks for evaluating the performance of models on real-world graph datasets [29].
- **Actor co-occurrence network** [13]: The Actor collaboration network is a graph that represents the collaboration of actors in movies. It is a subgraph of the larger director-actor-writer network, which includes the collaboration of directors, actors, and writers. In this graph, nodes represent actors, and edges represent their collaborations in movies. The node features represent the keywords associated with each actor, while the labels indicate the type of the actor, such as "actor", "actress", "producer", etc [29].

### 4.2. Baselines

To verify the effectiveness of RUNCL, we conducted a comparative analysis with ten baseline models. These models included spectral-based methods such as SGC and GCN, spatial-based methods like Graph Attention Networks (GAT), Approximate Personalized Propagation of Neural Predictions (APPNP), and GraphSAGE, and graph structure learning-based methods like Local

[1] https://github.com/BUPT-GAMMA/Graph-Structure-Estimation-Neural-Networks
[2] https://github.com/liun-online/HeCo

**Table 2**
Model hyperparameters.

| Dataset | nhid1 | nhid2 | dropout | lr | epoch | k |
|---|---|---|---|---|---|---|
| **Cora** | 768 | 512 | 0.7 | 0.01 | 200 | 7 |
| **Citeseer** | 768 | 512 | 0.7 | 0.01 | 200 | 8 |
| **Pubmed** | 768 | 512 | 0.7 | 0.01 | 200 | 6 |
| **Chameleon** | 768 | 512 | 0.5 | 0.005 | 200 | 11 |
| **Squirrel** | 768 | 512 | 0.5 | 0.005 | 200 | 15 |
| **Actor** | 768 | 512 | 0.5 | 0.005 | 200 | 8 |

Degree Centrality-based Sampling (LDS), Proximity Graph Neural Network (Pro-GNN), and Geometric Matrix Completion with Graph Convolutional Networks (Geom-GCN). Additionally, we tested two different variants of RUNCL to evaluate the efficacy of the various modules employed. The baseline models are described below:

- **SGC** [40]: This model is a simplified variant of the Graph Convolutional Network, which aims to streamline the computation process by eliminating certain nonlinearities and weight matrices.
- **GCN** [18]: A scalable approach for graph-structured data that can handle large amounts of unlabeled data, using a variant of convolutional neural networks that operate directly on the graph structure.
- **ChebNet** [17]: This model extends convolutional neural networks beyond regular grids, such as images or videos, to high-dimensional irregular domains, such as graphs or social networks.
- **GAT** [23]: A neural network works on graph-structured data and can assign different weights to nodes based on their relationships with other nodes in the network, allowing for more nuanced analysis of complex networks.
- **APPNP** [25]: This model improves upon graph convolutional network by using personalized PageRank to better identify relevant information in graph-structured data.
- **GraphSAGE** [20]: A framework that generates node embeddings using node feature information. This allows it to produce high-quality embeddings even with limited labeled data, making it useful for tasks such as node classification and recommendation in social network analysis and other applications.
- **LDS** [12]: This model takes a unique approach to learning the graph structure and parameters of graph convolutional network simultaneously. By utilizing a bilevel programming approach, which allows LDS to optimize both the graph structure and GCN parameters at the same time.
- **Pro-GNN** [28]: A framework that uses the intrinsic properties of real-world graphs, such as being low-rank and sparse, to jointly learn a structural graph and a robust graph neural network model from a perturbed graph.
- **Geom-GCN** [13]: This model introduces a new method for geometric aggregation in graph neural networks, which improves the preservation of structural information and captures long-range dependencies in disassortative graphs.
- **GEN** [29]: An approach for fitting the mechanism of GNNs is presented by generating graphs with multifaceted observations injected into the posterior distribution of graphs in order to calculate the posterior distribution of the graphs.
- **GEM** [41]: A semi-supervised learning method from 2023, optimizing prediction accuracy, training efficiency, and inference speed simultaneously, ideal for resource-constrained environments.
- **RUNCL**$_{CO}$: A variant of RUNCL where the graph generation module is deleted.
- **RUNCL**$_{SU}$: A variant of RUNCL where the contrastive learning module is deleted.
- **RUNCL**: The proposed semi-supervised node classification model.

### *4.3. Parameters setting*

To thoroughly assess the RUNCL model, we begin by setting the parameters of the RUNCL model, as well as those of the baseline models, to random values and then evaluate the model's performance by optimizing it through Adam. We select three label rates for the training set: 20, 10, and 5 labeled nodes for each category. To prevent overfitting, we use $ReLU$ as the activation function and apply dropout with rates varying from 0.5 to 0.7. The proposed RUNCL model is implemented using the PyTorch deep learning library, while all baseline models are initialized with the same parameters as suggested by their papers. For our model, the hidden layer dimensions of each GCN are set to 768 and 512 in different experimental configurations. The model hyperparameters are summarized in Table 2.

### *4.4. Main results*

Table 3 shows the results of node classification from six datasets used in our study, listing the number of nodes with labels in each class as L/C. We used accuracy to evaluate each model's performance, with higher values indicating better classification effectiveness.

In experiments with fewer labels, RUNCL consistently showed superior results in all datasets. Specifically, it achieved notable improvements in classification accuracy on the **Citeseer** and **Cora** datasets. This improvement is due to the model's ability to learn the structure of neighborhood information in the node relationship graph effectively, thus enhancing its robustness against noise or sparseness in the original data.

**Table 3**
Node classification results (%) (bold = best).

| Datasets | L/C | SGC | GCN | ChebNet | GAT | APPNP | GraphSAGE | LDS | Pro-GNN | Geom-GCN | GEN | GEM | RUNCL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Cora** | 20 | 80.9 | 81.7 | 81.9 | 82.3 | 83.3 | 80.1 | 82.5 | 80.9 | 63.7 | 83.6 | 83.5 | **84.2** |
| | 10 | 75.5 | 74.6 | 72.5 | 76.9 | 75.6 | 72.9 | 77.1 | 76.9 | 47.1 | 77.8 | 78.2 | **78.7** |
| | 5 | 72.8 | 71.0 | 66.6 | 75.0 | 72.9 | 68.4 | 75.7 | 75.1 | 22.5 | 76.2 | 77.8 | **78.4** |
| **Citeseer** | 20 | 71.9 | 70.9 | 70.0 | 72.0 | 72.0 | 71.8 | 72.3 | 68.8 | 65.6 | 73.8 | 74.2 | **75.3** |
| | 10 | 68.6 | 66.6 | 67.3 | 68.4 | 70.2 | 68.0 | 70.4 | 69.1 | 48.8 | 72.4 | 73.9 | **74.3** |
| | 5 | 61.3 | 53.5 | 51.7 | 61.8 | 54.2 | 55.4 | 68.1 | 56.6 | 28.3 | 70.4 | 71.5 | **72.9** |
| **Pubmed** | 20 | 77.1 | 79.4 | 78.2 | 77.9 | 79.6 | 73.6 | 78.2 | 78.0 | 77.2 | 80.9 | 80.5 | **81.0** |
| | 10 | 71.9 | 73.7 | 71.5 | 71.1 | 73.5 | 70.6 | 74.4 | 72.7 | 69.3 | 75.6 | **76.6** | 76.0 |
| | 5 | 68.7 | 73.0 | 69.4 | 70.2 | 73.8 | 70.2 | 72.8 | 70.6 | 68.4 | 74.9 | 74.5 | **75.4** |
| **Chameleon** | 20 | 49.1 | 49.1 | 37.0 | 43.7 | 46.1 | 43.7 | 49.4 | 50.3 | 35.7 | 50.4 | 51.4 | **52.3** |
| | 10 | 44.4 | 44.2 | 32.5 | 41.7 | 39.4 | 41.7 | 44.9 | 45.5 | 31.6 | 45.6 | 46.0 | **46.2** |
| | 5 | 39.3 | 39.5 | 33.2 | 35.9 | 36.9 | 35.9 | 40.5 | 41.0 | 28.5 | 41.4 | 41.8 | **42.1** |
| **Squirrel** | 20 | 34.7 | 35.0 | 21.2 | 28.3 | 33.6 | 28.3 | 30.1 | 33.4 | 25.4 | 35.5 | **36.9** | 36.2 |
| | 10 | 31.8 | 33.0 | 18.8 | 25.9 | 31.4 | 25.9 | 29.4 | 23.9 | 21.6 | 33.4 | 33.8 | **34.3** |
| | 5 | 29.1 | 31.3 | 18.1 | 24.9 | 27.0 | 24.9 | 27.1 | 28.2 | 22.6 | 32.7 | 33.1 | **33.5** |
| **Actor** | 20 | 22.0 | 21.7 | 26.7 | 28.9 | 29.7 | 28.9 | 27.0 | 21.5 | 20.7 | 35.3 | 36.1 | **36.6** |
| | 10 | 22.0 | 20.8 | 22.3 | 22.2 | 28.0 | 22.2 | 25.7 | 22.2 | 20.7 | 31.3 | 32.4 | **33.4** |
| | 5 | 24.2 | 21.8 | 21.4 | 23.1 | 22.4 | 23.1 | 23.8 | 20.9 | 24.1 | 30.5 | 31.2 | **32.6** |

**Table 4**
Node classification results of the variants (%).

| Datasets | L/C | RUNCL$_{CO}$ | RUNCL$_{SU}$ | RUNCL |
|---|---|---|---|---|
| **Cora** | 20 | 82.0 | 82.9 | **84.2** |
| | 10 | 75.4 | 78.1 | **78.4** |
| | 5 | 73.2 | 76.2 | **78.7** |
| **Citeseer** | 20 | 71.7 | 73.4 | **75.3** |
| | 10 | 68.0 | 72.7 | **74.3** |
| | 5 | 58.3 | 70.3 | **72.9** |

Our model excelled against both GCN and GAT across all datasets, showcasing its capacity to build optimal graphs. The combined approach of graph structure estimation and GCN parameter optimization led to a mutual enhancement, improving overall performance. The model also outperformed the baseline, GEN, on all datasets, confirming its ability to create less biased graphs and generate quality samples for node classification through contrastive learning.

Table 4 illustrates the node classification results of different variations. When solely relying on contrastive learning, as in RUNCL$_{CO}$, the model falls short as it cannot update node relationships to minimize biases. Compared to this, RUNCL performs better, signifying the effectiveness of employing contrastive learning across multiple viewpoints.

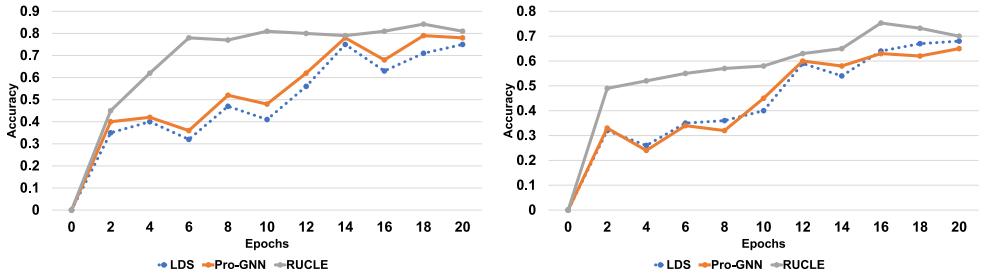
Observations indicate that traditional GNN methods witness a significant performance dip with a decreasing label rate, unlike models using graph structure learning methods like GEN and RUNCL. For example, in the **Citeseer** dataset, where the accuracy of GCN is 70.9% when 20% of the labels are available. However, as the percentage of labeled data decreases to 5%, the accuracy of GCN drastically drops to 53.5%. While the accuracy of RUNCL only decreases from 75.3 to 72.9. As a result, we observe a more significant enhancement in the performance of this model at lower label rates, specifically at 10% and 5%. This improvement can be attributed to the contrastive learning module, which can generate high-quality positive and negative samples, thereby mitigating the impact of limited labeled data in the dataset.

### *4.5. Model analysis*

To provide further insight into the performance improvement of RUNCL's each module, we analyze the change of accuracy in **Cora** and **Citeseer** datasets, Table 4 presents the outcomes obtained from the different variations.

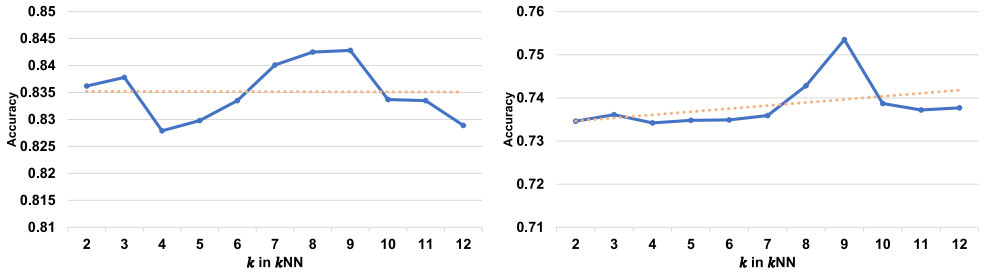
***Effectiveness of the Graph Generation Module.*** We analyze the effectiveness of the graph generation module. In every dataset, RUNCL$_{CO}$ is composed of two inputs: the node feature graph and the initial node relationships graph. The accuracy comparison reveals that RUNCL outperforms RUNCL$_{CO}$. This is mainly due to the fact that real-world graphs in complex systems are commonly prone to errors, which inevitably have an adverse impact on the overall accuracy of the model. For example, missing links exist in protein interaction graphs since the experimental error is inevitable. In the Cora dataset, incorrect links between two authors with no collaborative relationship due to author ambiguation problems. With the graph generation module, the model can generate a new node relationship graph using the neighborhood information to ensure that the updated node relationship graph is able to describe the relationships between different nodes more accurately.

***Effectiveness of the Contrastive Learning Module.*** In RUNCL, we leverage both the node feature graph and node relationship graph to jointly supervise and learn embeddings. In this section, we investigate the efficacy of the contrastive learning module. In




(a) Cora (b) Citeseer

**Fig. 4.** Effectiveness of the multi-view module.




(a) Cora (b) Citeseer

**Fig. 5.** Effectiveness of the multi-layer attention module.

each dataset, it can be seen that when the label rate is 20, the RUNCL does not improve significantly compared to the RUNCL$_{SU}$. However, when the label rate drops to 5, RUNCL has the most significant improvement. This is because, for semi-supervised node classification, more labels imply higher accuracy. Moreover, the model uncovers node relationships that were previously overlooked in the initial relationship graph. This capability enables the generation of high-quality positive and negative samples that can serve as valuable labels.

***Convergence Speed.*** Fig. 4 illustrates the accuracy of the LDS, Pro-GNN on the RUNCL model for Cora and Citeseer. The vertical coordinate in the figure is the model accuracy, and the horizontal coordinate is the epochs. Based on both datasets, RUNCL has a faster convergence time and better accuracy, which proves the efficiency and effectiveness of RUNCL. Meanwhile, both LDS and Pro-GNN are subject to a lot of fluctuation in accuracy, but the accuracy of RUNCL has steadily improved. This shows that the model is robust after considering various information, such as the neighborhood information graph.

***Effectiveness of $k$-Nearest Neighbor.*** Fig. 5 demonstrates the influence of the top $k$ neighborhoods in the $k$-nearest neighbor graph. The $x$-axis represents the number of neighbors, while the $y$-axis shows the accuracy. The experiments were conducted independently on the **Cora** and **Citeseer** datasets. As observed in the figure, the accuracy initially increases as the number of neighbors increases, but then reaches a peak and starts to decrease. We believe this result occurred because a larger $k$ may introduce more noise, and a smaller $k$ may cause information loss.

Table 5 illustrates the performance difference between our model and the GEN model in two sub-datasets: low-degree and high-degree nodes. We utilized the Cora dataset, conducting training and testing on both our model and the GEN model. During model training, the training and validation sets remained consistent, with the only variation being the categorization of the test set into two subsets based on node degrees: low-degree nodes (the lowest 200-degree nodes in the test set) and high-degree nodes (the highest 200-degree nodes in the test set). Note that to highlight model improvement, the 'boost' column displays a comparison between our model and the GEN model, calculated as Boost $= \frac{\text{Our Model's Result}-\text{Baseline Model's Result}}{\text{Baseline Model's Result}}$.

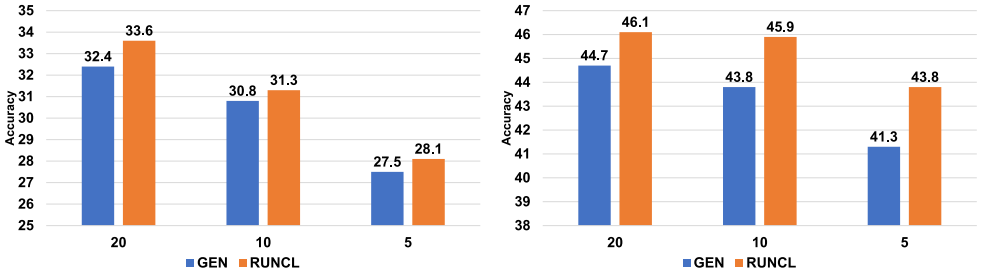
From the table, it is apparent that, overall, the model exhibits superior performance on the test set's low-degree nodes compared to the high-degree nodes. For instance, when the label rate is 20, and all test nodes are utilized, our model improve the result from 83.6% to 84.2%. However, when the test set comprises only low-degree nodes, the model's test result escalates from 44.7% to 46.1%.

Low-degree or isolated nodes typically possess limited information in graphs due to their sparse neighborhood connections, which can lead to information loss during the aggregation. This makes it challenging for models to learn effective feature representations for these nodes. However, the application of graph contrastive learning significantly mitigates this issue.

Contrastive learning in graphs allows the model to go beyond relying solely on nodes' direct neighborhood information. By constructing positive and negative samples, even isolated nodes can learn from other nodes that are either similar (positive samples) or non-similar (negative samples). This approach introduces additional contextual information for isolated or low-degree nodes, enriching the feature space available for the model to learn from.

**Table 5**
Performance difference between our model and the GEN model in two sub-datasets: low-degree and high-degree nodes (%).

| | L/C | ALL testset | High-degree nodes testset | Low-degree nodes testset | Boost |
|---|---|---|---|---|---|
| GEN | 20 | 83.6 | 32.4 | 44.7 | 0 |
| | 10 | 77.8 | 30.8 | 43.8 | 0 |
| | 5 | 76.2 | 27.5 | 41.3 | 0 |
| RUNCL | 20 | 84.2 | 33.6 | 46.1 | 3.13 |
| | 10 | 78.7 | 31.3 | 45.9 | 4.79 |
| | 5 | 78.4 | 28.1 | 43.8 | 6.05 |




(a) Performance on High-degree Nodes (b) Performance on Low-degree Nodes

**Fig. 6.** Performance difference between our model and the baseline model.

Additionally, Fig. 6 also reveals that with a reduced label rate, the enhancement in low-degree nodes becomes more pronounced. For example, considering the low-degree nodes' results, our model exhibits a 3.13% improvement when the label rate is 20. However, when the label rate decreases to 10, our model's performance escalates to a 4.79% improvement, and at a label rate of 5, it rises to a 6.05% improvement. This suggests that through contrastive learning, our model can forge a more extensive set of negative samples. Thus, when fewer labels are available in the training set, our model demonstrates a more significant improvement.

### *4.6. Datasets analysis*

In this section, we analyze how the consistency between node features and labels affects model accuracy across various datasets. Model accuracy tends to be higher in datasets where node features and labels closely align. For example, in the Cora dataset, papers are represented by node features like bag-of-words and labeled by academic fields such as Case-Based, Genetic Algorithms, and Theory. A paper's representation is deeply influenced by its node features, suggesting a strong correlation with its labels, leading to higher model accuracy.

Taking a specific paper, "P1728" in the Cora dataset illustrates this. It has five neighbors; three share the same label, while two do not, despite having closely related node features. Our model effectively updates node relationships and assigns appropriate labels based on these features, demonstrating its efficacy when there is a high feature-label match.

Contrastingly, in the Squirrel dataset, which associates nodes with Wikipedia topics, the correlation between node features and labels is less pronounced. For instance, considering page "P258" has neighbors sharing similar topics but having different visitors counts as labels. Here, updating node relationships solely based on features might be less effective, leading to subpar model outcomes. Thus, our model performs better in datasets like Cora, where node features and labels are more coordinated, compared to Squirrel, where such correlation is lacking.

## 5. Conclusion and future work

This paper introduces **RUNCL**: **R**elationship **U**pdating **N**etwork with **C**ontrastive **L**earning model, which addresses the node classification problem under limited labeled data for low-degree nodes in graph datasets. We leverage two views of the graph data, the node feature graph, and the node relationship graph, to capture local and high-order structures. Furthermore, contrastive learning is utilized to enhance the performance of the model. Our experiments demonstrate that our approach achieves superior results compared to most recent models, particularly in scenarios with limited labeled data. The model exhibited an improvement of 2.5% in the testset. Moreover, the model's performance boosted to 6% when the testset only included low-degree nodes. Looking ahead, we envision the following two areas for future work:

**Exploration of the Model's Functionality on Temporal Graph Data.** In future work, we aim to investigate the impact of temporal variations in graph structures on the performance of our RUNCL model. Given the presence of a time dimension in the graph data, the architecture of the graphs might evolve over time. Understanding how these dynamic changes influence the model's effectiveness and adaptability is crucial.

**Extension to Other Graph-Based Tasks.** Another promising direction could be extending the applicability of the RUNCL model to other graph-based tasks such as link prediction, graph clustering, and graph completion. Investigating how the model can be adapted and fine-tuned for various tasks will broaden its usability and practical relevance.

## CRediT authorship contribution statement

**Pengyu Zhang:** Writing – review & editing, Writing – original draft, Methodology. **Yong Zhang:** Writing – review & editing, Supervision, Methodology. **Xinglin Piao:** Methodology. **Yongliang Sun:** Methodology. **Baocai Yin:** Supervision.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yong Zhang reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This study received support from the National Natural Science Foundation of China, including grants No. 62072015, U19B2039, and U1811463.

## References

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81, http://dx.doi.org/10.1016/j.aiopen.2021.01.001, URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.
[2] C. Gao, S. Yin, H. Wang, Z. Wang, Z. Du, X. Li, Medical-knowledge-based graph neural network for medication combination prediction, IEEE Trans. Neural Netw. Learn. Syst. (2023) 1–12, http://dx.doi.org/10.1109/TNNLS.2023.3266490.
[3] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML '17, JMLR.org, 2017, pp. 1263–1272.
[4] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, J. Leskovec, Strategies for pre-training graph neural networks, 2019, arXiv preprint arXiv:1905.12265.
[5] R. Wang, C. Shi, T. Zhao, X. Wang, Y. Ye, Heterogeneous information network embedding with adversarial disentangler, IEEE Trans. Knowl. Data Eng. 35 (2) (2023) 1581–1593, http://dx.doi.org/10.1109/TKDE.2021.3096231.
[6] M.E. Newman, Estimating network structure from unreliable measurements, Phys. Rev. E 98 (6) (2018) 062321.
[7] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.
[8] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4116–4126.
[9] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[11] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.
[12] L. Franceschi, M. Niepert, M. Pontil, X. He, Learning discrete structures for graph neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 1972–1982.
[13] H. Pei, B. Wei, K.C.-C. Chang, Y. Lei, B. Yang, Geom-Gcn: Geometric graph convolutional networks, 2020, arXiv preprint arXiv:2002.05287.
[14] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[15] C. Gao, J. Zhu, F. Zhang, Z. Wang, X. Li, A novel representation learning for dynamic graphs based on graph convolutional networks, IEEE Trans. Cybern. 53 (6) (2023) 3599–3612, http://dx.doi.org/10.1109/TCYB.2022.3159661.
[16] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint arXiv:1312.6203.
[17] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, vol. 29, 2016.
[18] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[19] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: International Conference on Machine Learning, PMLR, 2016, pp. 2014–2023.
[20] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[21] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, 2015, arXiv preprint arXiv:1511.05493.
[22] Y. Zhang, Q. Liu, L. Song, Sentence-state LSTM for text representation, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 317–327.
[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[24] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.
[25] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, 2018, arXiv preprint arXiv:1810.05997.
[26] T. Martin, B. Ball, M.E. Newman, Structural inference for uncertain networks, Phys. Rev. E 93 (1) (2016) 012306.

[27] Y. Zhang, S. Pal, M. Coates, D. Ustebay, Bayesian graph convolutional neural networks for semi-supervised classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (no. 01) 2019, pp. 5829–5836.
[28] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, J. Tang, Graph structure learning for robust graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 66–74.
[29] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, X. Xie, Graph structure estimation neural networks, in: Proceedings of the Web Conference 2021, 2021, pp. 342–353.
[30] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, S. Pan, Towards unsupervised deep graph structure learning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1392–1403.
[31] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, C. Shi, Compact graph structure learning via mutual information compression, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1601–1610.
[32] P. Bachman, R.D. Hjelm, W. Buchwalter, Learning representations by maximizing mutual information across views, in: Advances in Neural Information Processing Systems, vol. 32, 2019.
[33] A. Van den Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv e-prints, arXiv–1807.
[34] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, ICLR (Poster) 2 (3) (2019) 4.
[35] X. Wang, N. Liu, H. Han, C. Shi, Self-supervised heterogeneous graph neural network with co-contrastive learning, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1726–1736.
[36] D. Chen, Y. Lin, L. Li, X. Ren, P. Li, J. Zhou, X. Sun, Rethinking the promotion brought by contrastive learning to semi-supervised node classification, 2020, arXiv preprint arXiv:2012.07437.
[37] X. Luo, W. Ju, M. Qu, C. Chen, M. Deng, X.-S. Hua, M. Zhang, Dualgraph: Improving semi-supervised graph classification via dual contrastive learning, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 699–712.
[38] J. Zeng, P. Xie, Contrastive self-supervised learning for graph classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, (no. 12) 2021, pp. 10824–10832.
[39] S. Suresh, P. Li, C. Hao, J. Neville, Adversarial graph augmentation to improve graph contrastive learning, Adv. Neural Inf. Process. Syst. 34 (2021) 15920–15933.
[40] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6861–6871.
[41] Y. Luo, G. Luo, K. Qin, A. Chen, Graph entropy minimization for semi-supervised node classification, 2023, arXiv:2305.19502.