mathematics

MDPI

Article

A Negative Sample-Free Graph Contrastive Learning Algorithm

Dongming Chen [1], Mingshuo Nie [1,*], Zhen Wang [1], Huilin Chen [2] and Dongqi Wang [1]

[1] Software College, Northeastern University, Shenyang 110819, China; chendm@mail.neu.edu.cn (D.C.); 2071322@stu.neu.edu.cn (Z.W.); wangdq@swc.neu.edu.cn (D.W.)
[2] College of Engineering, Computing and Cybernetics, Australian National University, Canberra, ACT 2601, Australia; u7326198@anu.edu.au
* Correspondence: niemingshuo@stumail.neu.edu.cn

Abstract: Self-supervised learning is a new machine learning method that does not rely on manually labeled data, and learns from rich unlabeled data itself by designing agent tasks using the input data as supervision to obtain a more generalized representation for application in downstream tasks. However, the current self-supervised learning suffers from the problem of relying on the selection and number of negative samples and the problem of sample bias phenomenon after graph data augmentation. In this paper, we investigate the above problems and propose a corresponding solution, proposing a graph contrastive learning algorithm without negative samples. The model uses matrix sketching in the implicit space for feature augmentation to reduce sample bias and iteratively trains the mutual correlation matrix of two viewpoints by drawing closer to the distance of the constant matrix as the objective function. This method does not require techniques such as negative samples, gradient stopping, and momentum updating to prevent self-supervised model collapse. This method is compared with 10 graph representation learning algorithms on four datasets for node classification tasks, and the experimental results show that the algorithm proposed in this paper achieves good results.

Keywords: complex networks; graph representation learning; self-supervised learning; data augmentation; comparative learning

MSC: 05C82

Citation: Chen, D.; Nie, M.; Wang, Z.; Chen, H.; Wang, D. A Negative Sample-Free Graph Contrastive Learning Algorithm. *Mathematics* **2024**, *12*, 1581. https://doi.org/10.3390/math12101581

Academic Editor: Michele Bellingeri

Received: 5 April 2024
Revised: 9 May 2024
Accepted: 16 May 2024
Published: 18 May 2024

1. Introduction

Graph neural networks (GNNs) are powerful deep learning tools used to model graph-structured data and have shown outstanding performance in various graph learning tasks. Despite the strong capabilities of GNNs, their effectiveness in deep graph learning largely depends on high-quality input training graphs and real labels. The performance of GNNs on real-world graphs is often fragile because of the lack of labeled training samples, which may lead to overfitting and difficulty in generalization, thus losing their ability to solve various downstream deep graph learning tasks. On the other hand, real-world graphs are typically complex and inevitably contain redundant, erroneous, or missing features and connections. Training GNN-based models directly in such cases may result in a severe performance drop.

In the negative sampling-based contrastive learning method, negative examples play a crucial role because a large number of negative samples are needed to ensure rich information in the learned node representations and to guarantee the quality of the model. This helps to avoid the problem of model collapse in representation learning methods. However, this leads to a higher demand for computational resources, and the algorithm's effectiveness is to some extent dependent on the method of generating negative samples. The generation of positive and negative samples is achieved through data augmentation, which introduces sample bias [1]. This bias arises from changes in the graph structure

during graph augmentation, and the subsequent graph neural network encoding relies on neighborhood aggregation to embed nodes. This means that node embedding are heavily influenced by neighboring nodes, especially those with smaller degrees. When neighborhood information is disrupted by data augmentation, it blocks the pathway for nodes to acquire information from neighbors, resulting in significant fluctuations in node embedding and biased augmented samples.

To address these issues, this paper proposes a negative sample-free graph contrastive learning algorithm that conducts graph contrastive learning at the vector level. This work aims to eliminate the need for negative samples and reduce the bias caused by graph augmentation operations. The contributions of this paper are as follows:

1. Converting the adjacency matrix into a diffusion matrix as a perspective for graph augmentation. Unlike other graph augmentation methods that disturb the graph structure, this approach aggregates neighbors from another perspective, enabling the model to learn rich graph information.
2. Since graph augmentation operates in the input space and the embedding distribution is often difficult to predict or control after GNN influence, this paper performs feature augmentation in the implicit space. By using covariance, the data distribution after feature augmentation in the implicit space can be better controlled.
3. A target function [2] is used in this work, which starts from the embedding data themselves rather than the samples. It also incorporates the idea of reducing redundancy. For the same input, the features obtained from the upper and lower branches should have a similarity-cross-correlation matrix with a main diagonal close to the identity matrix (i.e., the feature correlation matrix is close to the identity matrix), while the off-diagonal elements should be close to zero. The idea of the target function avoids the collapse problem in negative sample-free self-supervised models. Furthermore, it does not require weight sharing, the same architecture for the upper and lower branches, or inputs with the same properties, nor does it require a repository, etc.

2. Related Work

To improve the sufficiency and quality of training data, data augmentation is considered an effective tool that expands the given input data by slightly modifying existing data instances or generating synthetic new instances from existing ones. In recent years, the importance of data augmentation has been fully recognized in the computer vision and natural language processing fields. Recently, data augmentation techniques have also been explored in the graph field to push the performance boundaries of deep graph learning, and promising results have been demonstrated. In addition to traditional image or text data, graph-structured data has become more complex due to heterogeneous information modalities and complex graph properties, providing a wider design space and additional challenges for graph data augmentation.

However, due to the inherent non-Euclidean nature of graph data, it is difficult to directly apply data augmentation strategies designed for images to the GNN field. Here, this paper categorizes data augmentation strategies for graph data into three types: feature-based augmentation, structure-based augmentation, and sampling-based augmentation.

2.1. Feature-Based Augmentation

Feature masking (FM) [3–5]. The data augmentation technique based on node feature masking is to generate new graph data by randomly masking a part of the node's features, e.g., randomly selecting some nodes and then setting their features to zero or some other specific values to simulate the situation where some nodes are missing from the data as shown in Figure 1. This method can simulate the situation of some nodes with incomplete attributes in the real world, increase the diversity of the data, and improve the robustness of the model at the same time. In addition, the data augmentation technique based on node feature masking can also reduce the model's dependence on certain node features, thus improving the model's generalization performance.

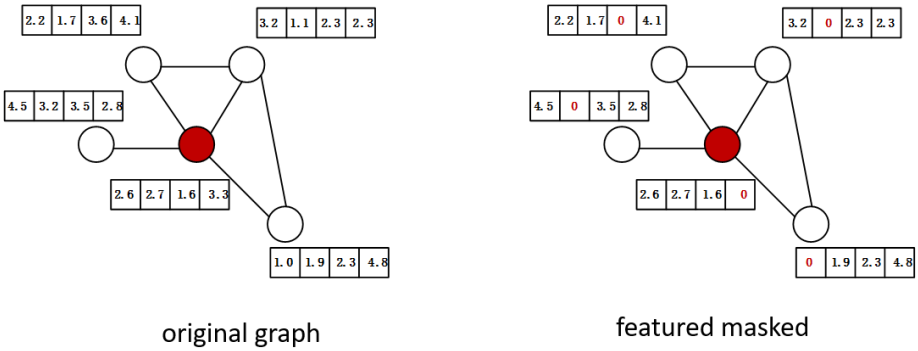

Figure 1. Masking node features.

Feature shuffling [6] is a technique used in graph neural networks (GNNs) to randomize the node and edge features, where the input features of the GNN will be randomized while other parts (such as the network topology) remain unchanged, and then the output of the GNN is computed as shown in Figure 2. The impact of each feature on the performance of the GNN can be measured by comparing the original inputs with the outputs after the features are randomly arranged.

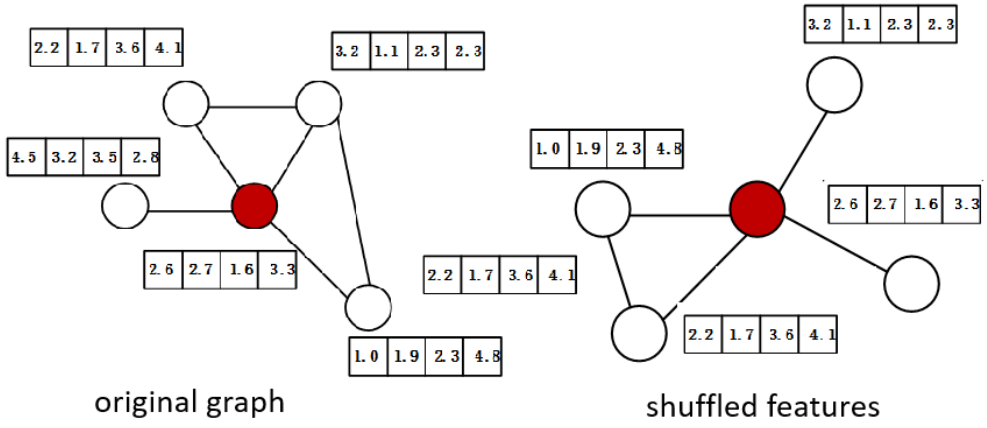

Figure 2. Shuffled features.

2.2. Structure-Based Augmentations

Edge perturbation (FD) [5] is a graph structure perturbation strategy that mainly changes the structure of a graph by randomly adding or removing a certain percentage of edges. Its goal is to introduce a certain amount of randomness while keeping the general structure of the graph unchanged, to test whether the semantics of the graph is robust to such differences in edge connection patterns. When applying edge perturbations, this paper also follows the default uniform distribution for adding/removing each edge.

Node dropping (ND). Randomly removes a portion of nodes from the graph to perturb the integrity of the graph, and the probability of removal of each node follows the default independent homogeneous distribution, constrained a priori to the fact that the removed nodes do not affect the semantic information of the entire graph.

2.3. Sampling-Based Augmentation

For a graph G = (X,A), sampling-based augmentation transforms both the feature matrix X and the adjacency matrix A. The underlying assumption is that the semantics of the graph will be preserved in its local structure, as shown in Figure 3.

In graph neural networks, sampling-based data augmentation is a common technique used to increase the diversity and quantity of the training dataset, thus improving the

generalization ability of the model. Specifically, this technique is used to train a model by randomly selecting several subgraphs (i.e., samples) in the original graph data and using them as inputs. These subgraph can be selected randomly, based on the properties of nodes or edges in the graph, or based on some specific rules. With the sampling approach, several different subgraph can be generated, allowing the model to learn more different graph features, thus improving its generalization ability. In addition, the sampling method can effectively reduce the size of the graph dataset, thus making training more efficient.

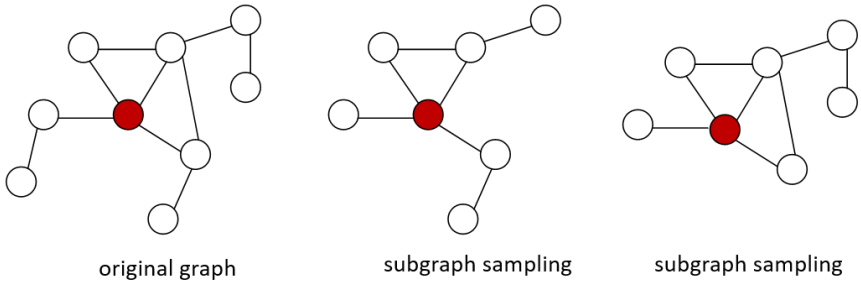

Figure 3. Sampling-based augmentation.

Uniform sampling [7] is a commonly used sampling method whose main goal is to randomly select a subraph from the graph. The uniform sampling process is to randomly select a node from the original graph as the center node of the subgraph, and from this center node, traverse the graph along the edges and collect neighboring nodes as needed. Maximum distance or other traversal constraints can be set as per requirement. The collected nodes and the edges between them are formed into a subgraph. Due to the random selection of nodes from the original graph, uniform sampling increases data diversity and helps to improve the generalization ability of the model.

The random walking sampling method [8] selects a subgraph from the graph by random walking. This method can be done by randomly selecting a node from the original graph as the starting point of the random walk. Starting from the current node, a neighboring node is randomly selected as the next node. Each walking selection will be based on the weights of the edges or other probability distributions. The above process is repeated until a predetermined walking length is reached. The nodes visited during the walking process and the edges between them are formed into a subgraph. Randomized walking sampling preserves local structural information in the graph because neighboring nodes are visited during the walking process. Randomized walking sampling can control the size and structure of the subgraph by adjusting parameters such as walking length and probability distribution. However, it has some limitations, such as incomplete information coverage and high computational complexity.

Importance sampling [9] is a method of selecting subgraph based on the importance of nodes or edges. Importance sampling has the advantage of extracting parts with higher information value, thus improving the learning effect, and by selecting only important nodes or edges, it can effectively reduce the size of the subgraph and reduce the computational complexity, but there is also the advantage that calculating the importance of the nodes or edges may require additional computational costs, especially for large-scale graphs. Moreover, importance sampling may ignore some non-important parts of the graph that may also be valuable for the learning task.

Knowledge-based sampling (KBS) [10] is a method that combines a priori knowledge or domain knowledge to select a subraph. The method utilizes existing knowledge to filter nodes and edges in the graph to extract more representative and valuable subgraphs, which in turn improves the performance of graph self-supervised learning. The algorithmic process is to collect a priori knowledge or domain knowledge, which can be about the properties of nodes and edges in the graph, or about the topology of the subgraph. Based on the collected knowledge, a weight or score is assigned to each node or edge. Sort the

nodes or edges according to the weights or scores. Select the nodes or edges with higher weights or scores to form the subgraph. A threshold can be set or a certain percentage of nodes or edges can be selected. It has the advantages of utilizing existing knowledge, extracting high-quality subgraph, and interpretability, but also has limitations of relying on the quality of knowledge, possible bias, and higher computational cost.

2.4. Evaluation Indicators

The F1-score is a commonly used metric for evaluating the performance of binary or multi-category classification models, especially when the dataset is unbalanced in terms of positive and negative samples. The F1-score is a harmonic mean of the precision and recall rates, which takes into account both the precision and recall rates of the model. The F1-score is the average of Precision and Recall. The F1-score is higher only when both Precision and Recall are higher. This makes the F1-score an important model performance evaluation metric, because in many cases, we would like to have a model with a high level of both Precision and Recall, rather than favoring one over the other. The F1-score provides us with an effective way to measure the combined performance of the model in terms of Precision and Recall with a single metric. This is particularly important in practice, as we often want models to strike a good balance between accuracy and recall.

The F1-score is also known as the balanced F-score, and it is calculated as shown in Equation (1):

$$F1 - score = \frac{2 * precision * recall}{precision + recall}, \tag{1}$$

For multi categorization problems, there are two commonly used ways to calculate the F1-score: Micro-F1 and Macro-F1.

Micro-F1 is to count the TP, FP, and FN obtained in all categories, then calculate the overall Precision and Recall, and then finally use these two to calculate the obtained F1-score. In this method, all categories are treated as a whole to calculate the F1-score, so the prediction results of each sample are treated equally, which is often a better way to evaluate the F1-score in the case of category imbalance. This situation is often a better way of evaluation.

Macro-F1, on the other hand, is obtained by first calculating the F1-score for each category and then averaging the F1-scores for all categories. In Macro-F1, all categories have the same weight and are considered equally important regardless of the number of samples in the category. Therefore, Macro-F1 gives more importance to models that achieve good performance in each category

In short, Micro-F1 considers the proportion of all samples that are correctly predicted and is more concerned with the predictive accuracy of the broad categories, whereas Macro-F1 focuses on the mean of the predictive accuracy of each category and places more importance on the balance of predictions between categories. The choice of whether to use Micro-F1 or Macro-F1 when dealing with data with imbalanced categories needs to be determined based on the specific task requirements.

3. A Graph Contrastive Learning Algorithm without Negative Samples

3.1. Definition of Symbols

This subsection gives specific definitions of the relevant concepts in the proposed algorithmic framework and the explanations corresponding to the symbols that appear, as shown in Table 1.

Definition 1 (*Adjacency matrix*)**.** *The adjacency matrix is used to represent the relationship between vertices, e.g., for an adjacency matrix, if there is an edge between the nodes, the element in the adjacency matrix is 1, otherwise it is 0. For undirected graphs, whose adjacency matrix is symmetric, the value of the element on the main diagonal is 0. The adjacency matrix can be used to quickly determine whether there is an edge connection between any two vertices.*

Definition 2 (*Graph diffusion matrix*)**.** *A graph diffusion matrix is defined as a symmetrically normalized adjacency matrix where each element represents the probability of transferring from one node to another and the sum of the weights of all the outgoing edges of each node is one.*

Definition 3 (*Intercorrelation matrix*)**.** *An intercorrelation matrix is a measure of similarity between two vectors or matrices.*

Table 1. Definition of related symbols.

Symbolic	Define
G	given graph
V	nodes set of the graph
E	edge set of a graph
X	eigenmatrix of the graph
A	adjacency matrix of the graph
d	degree of nodes
S	graph diffusion matrix
α	diffusion coefficient
W	weighting matrix
H	node representation
Z	output node representation
C	Intercorrelation matrix
λ	objective function hyperparameters
b	batch size
P	affine transformation
F	dimension

3.2. Algorithm Design and Implementation

Previous graph contrastive learning algorithms have achieved good results, where most of the models are based on positive and negative samples for comparison, and the negative samples to avoid the problem of model collapse. However, the effect of this method depends on the number of negative samples, and the algorithm uses a large number of negative samples, which requires more arithmetic resources, and puts forward higher requirements for hardware devices. In this paper, we propose graph contrastive learning method without a negative sample, named WNS-GCL, where the core idea is to train two neural networks by reducing the redundancy between their outputs, specifically by minimizing the distance between the outputs of the two networks of the two networks. The model framework is shown in Figure 4, where GAL denotes the graph augmentation layer.

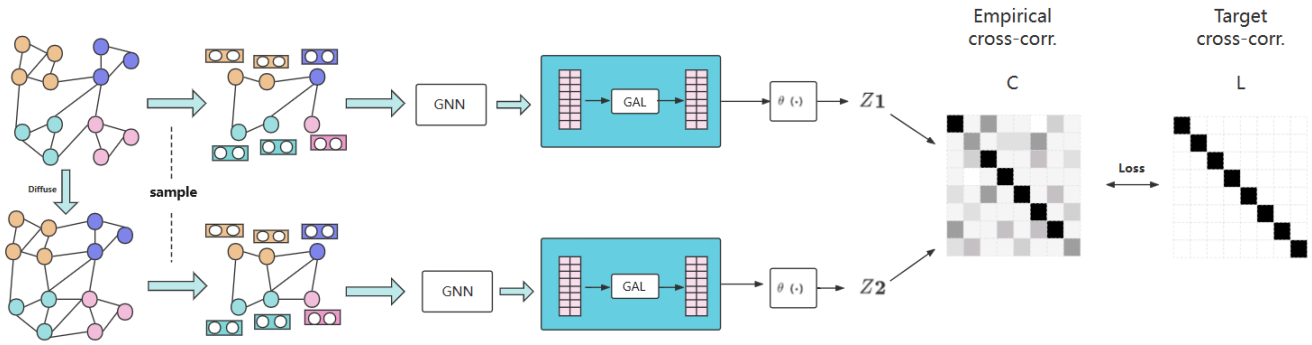

Figure 4. The overall framework of WNS-GCL model.

In the following, the components of the model are described in detail: graph data augmentation, graph encoder, projection head, and loss function.

3.2.1. Graph Data Augmentation

The graph data augmentation method used in this paper is the combined data augmentation (node feature masking and deletion of edges operation) strategy with better experimental results in Section 3, as well as the algorithm used in this paper, which is based on the randomized walking sampling (see the graph in Section 2.3).

Graph diffusion networks: acquiring graph representations can be categorized into two types: based on graph theory and based on node space structure. Graph diffusion networks (GDNs) are a synthesis of these two approaches and can integrate more neighbor information. Graph diffusion networks can effectively extract and integrate information from graph data by fusing these two approaches. The core of these networks is the graph diffusion process, which works by converting the adjacency matrix into a graph diffusion matrix, as shown in Equation (2):

$$S = \sum_{k=0}^{\infty} \theta_k T^k \,, \tag{2}$$

where θ_k is the weighting factor that determines the global-local information ratio, T is the generalized transition matrix, $\sum_{k=0}^{\infty} \theta_k = 1$, λ_i is the eigenvalues of T, and $\lambda_i \in [0, 1]$, to ensure convergence.

The two most commonly used methods in graph diffusion networks are Personalized PageRank (PPR) and Heat Kernel. The algorithm in this paper is the PPR method, which is used to generate the graph diffusion matrix, based on the idea of random walking. Moreover, PPR iteratively propagates the importance of the nodes in the graph by iterating the importance of the nodes so as to achieve a personalized ranking of each node. Here, $T_{rw} = AD^{-1}$ is the transition matrix ,$T_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is symmetric transition matrix, D is the degree matrix, $\theta_K^{PPR} = \alpha(1-\alpha)^K$s weighting factor, α is the transfer probability in random walking, and $\alpha \in (0,\ 1)$. The formula based on PPR diffusion is shown in Equation (3):

$$S^{PPR} = \alpha \left[I_n - (1-\alpha) D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right]^{-1} , \tag{3}$$

The algorithm in this paper adopts the graph diffusion network method, and the experimental effect is improved. The initial intention of the construction of the graph diffusion network is to optimize the information transfer mechanism of the graph neural network, so that the model can obtain more global information in the training process.

Matrix sketch [11]: after the GNN learns the node representation of the graph, it directly enhances the node feature representation. Here, the covariance is used as a constraint between the original feature matrix X and the augmented matrix $\tilde{X}$ (which needs to limit the size of $||X^T X - \tilde{X}^T \tilde{X}||$). In this paper, we convert the problem of feature augmentation matrix into a matrix sketching problem. Matrix sketching is where a given original feature matrix $X \in R^{n \times d}$ is passed through the following Equations (4) and (5), where Equation (4) has to satisfy Equation (5) in order to obtain the augmented feature matrix $\tilde{X} \in R^{k \times d}$:

$$X = PX + E \,, \tag{4}$$

$$||X^T X - \tilde{X}^T \tilde{X}||_2 \leq \varepsilon Tr(X^T X) \,, \tag{5}$$

where $P \in R^{k \times d}$ in Equation (4) denotes the transformation matrix, E is the random noise matrix, and ε is the error that controls the quality of the approximation $||X^T X - \tilde{X}^T \tilde{X}||_2$. The transformation matrix P is used here to generate a sketch of the matrix by means of random mapping [11].

Data augmentation is a stochastic process and the enhanced samples generated by random mapping need to satisfy Equation (6). Equation (6) is such that the probability that the covariance is constrained is at least guaranteed:

$$P(||X^T X - \tilde{X}^T \tilde{X}||_2 \leq \varepsilon Tr(X^T X)) \geq 1 - e^{-\frac{\varepsilon^2 k}{8}} \,, \tag{6}$$

3.2.2. Graph Encoder

The graph convolutional neural network encoder chosen for this algorithm encodes the same graph after subgraph sampling and inputs it into the encoder to learn the representation of the nodes, and the GNN encoder $f_i : R^{n\times n} \times R^{n\times d} \rightarrow R^{n\times d}$ extracts the hidden node features $H_i \in R^{n\times d}$ from the i-th augmented graph $(\tilde{A}_i, \tilde{X}_i)$. Usually, multiple encoders are applied to obtain hidden node features H_i for different views as follows:

$$H_1 = f_1(A_1, X_1), \cdots, H_k = f_k(A_k, X_k) \,, \tag{7}$$

The GNN encoder is implemented as a two-layer graph convolutional network (GCN):

$$GCN_l(X, A) = \sigma\left[D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X W_l\right] \,, \tag{8}$$

$$f(X, A) = GCN_2(GCN_1(X, A), A) \,, \tag{9}$$

where $A = A + I$ is the adjacency matrix with self-loop, D is the degree matrix, $\sigma(\cdot)$ is the activation function, $ReLu(\cdot)max(0, \cdot)$, and W_l is the matrix of the trainable weight layer.

3.2.3. Projection Head

The projection head $\theta(\cdot)$ is a small network that maps representations to the space in which the contrast loss is applied. It is implemented as a multilayer perceptron MLP with one hidden layer (consisting of two fully connected layers with bulk normalization (BN) and $ReLU$, and a third linear layer) to obtain $Z_i = \theta(H_i) = W^{(2)}\sigma W^{(1)} H_i$, where σ is the ReLU nonlinear activation function. Its role in the model is to: (1) eliminate two representations of different information and (2) extend the dimension in a nonlinear way so that the decorrelation of the embedded variables will reduce the dependence (and not only the correlation) between the two representations of the vector's variables.

3.2.4. Loss Function

The goal of this algorithm is to make the main diagonal of the intercorrelation matrix as close as possible to the unit matrix as a loss function, in which the embedding matrix is normalized along the batch dimensions, and after that the intercorrelation matrix $C \in R^{d\times d}$ of the two viewpoints is computed with the goal of optimizing it to make it close to the unit matrix. The optimized formula is:

$$\iota_{\beta\tau} \triangleq \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j\neq i} C_{ij}^2 \,, \tag{10}$$

where λ is a hyperparameter greater than 0 used to balance the first invariant term with the second redundant term and C is the intercorrelation matrix computed along the batch dimension between the outputs of two identical networks:

$$C_{ij} \triangleq \frac{\sum_b z_{b,i}^1, z_{b,j}^2}{\sqrt{\sum_b (z_{b,i}^1)^2}\sqrt{\sum_b (z_{b,j}^2)^2}} \,, \tag{11}$$

where b indexes the batch samples and i, j indexes the vector dimensions of the network output, C has dimension $d \times d$, and d is the feature dimension of the model output.

The second term of Equation (10), by trying to make the off-diagonal elements of the intercorrelation matrix closer to 0, aims at making the same dimensions of the two feature vectors extracted from the same sample after going through different networks as similar as possible, and to reduce redundancy between different dimensions.

This method draws on the strategy of reducing redundant information, and its core idea is to implement comparative learning in the embedded features, which is a significant difference since previous methods often perform this type of learning on samples. It strives to make different feature dimensions represent as diverse information as possible as a

way to enhance the expressiveness of the features, which further ensures the correlation between the two perspectives. At the same time, compared to loss functions commonly used in graph self-supervised learning, this approach does not rely on specific techniques, such as gradient stopping or momentum encoders.

4. Experimentation and Analysis

4.1. Experimental Dataset

In this section, four public benchmark datasets, i.e., Cora, Citeseer, Pubmed, and DBLP, which are widely used in the field of graph neural network research, are selected for experimental evaluation and the results are analyzed in comparison with the widely influential methods in recent years. Cora, Citeseer, and Pubmed belong to the citation network datasets [12], which consist of papers and their mutual citation relationships constitute a network, including citation relationships, co-authors, etc., which form the structure of the graph. In these datasets, each node represents an academic paper, the citation relationships between papers form edges, and each node contains word feature vectors and category labels, which represent the content features and subject categories of the paper, respectively. DBLP is a large indexing database of computer-based literature in the original XML format, which is organized to form a citation network dataset. The specific nodes, edges, and other information of these four datasets are shown in Table 2.

Table 2. Basic dataset information.

Dataset	Nodes	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
Pubmed	19,717	44,338	3	500
DBLP	17,716	105,734	4	1639

4.2. Experimental Environment

In this paper, all the experimental codes are written in Python language and the framework proposed in this paper as well as the benchmark models used in the experiments are implemented based on the Pytorch framework. The details of the environment on which the experiments depend are shown in Table 3.

Table 3. Experimental environment.

Hardware Environment	Parameters
OS	Ubuntu18.04
RAM	8 GB, 50 GB
CPU	Intel(R) Core(TM)i7-6700 CPU @3.40 GHz (Intel Corporation, Santa Clara, CA, USA)
GPU	NVIDIA RTX A2000 display memory 12G (NVIDIA Corporation, Santa Clara, CA, USA)
Development Tool	PyCharm
Development Language	Python 3.8

4.3. Evaluation Metrics

During the experiments, the performance of the learned node representations is evaluated by node classification as a downstream task, and Micro-F1 from the evaluation metrics in Section 2.4 is used as the evaluation metric.

4.4. Experimental Setup

In order to be able to demonstrate the effectiveness of their algorithms, the following ten algorithms that are representative of graph neural networks were chosen as comparison algorithms.

The Raw Features algorithm is based on the original node feature information. In the model, the feature vector of each node is directly used as the input, without any preprocessing or dimension reduction operation. The model represents each node as a vector, and the dimension of the vector usually corresponds to the dimension of the node features.

The Node2vec algorithm is based on learning node embedding from randomized walking sequences. The core idea of the algorithm is that by randomly walking over graph data, a large amount of data containing node sequences can be generated. These sequences can be considered as a "corpus" and can be used to train an embedding model that maps nodes into a low-dimensional vector space.

The core idea of the DeepWalk algorithm is to generate a large amount of data containing sequences of nodes by randomly walking over graph data, using a neural network-based approach to training embedding models, which makes use of negative sampling techniques in the Skip-gram model.

For the graph convolutional network (GCN), the core idea of the model is to utilize the idea of the convolutional neural network (CNN) to perform convolutional operations on graph data. Since the connectivity between nodes in graph data is non-Euclidean, the convolution operation needs to be redefined. GCN uses a Laplace matrix-based approach to define the convolution operation, which can be interpreted through the spectral graph theory.

The core idea of the SGC [13] model is to aggregate features from neighboring nodes to the central node using a linear transformation without using the nonlinear activation function and convolution operation in GCN. This linear transformation can be realized by multiple identical weight matrices, thus reducing the number of parameters and computation of the model.

GAE [14], a graph self-encoder, learns to extract the representation of node embedding from the input graph and reconstructs the original graph by treating the graph as an input to a self-encoder. The model is mainly divided into two parts: encoder and decoder. The core idea of the model is to use the representation of node embedding as an intermediate hidden layer of the self-encoder and learn the node embedding by minimizing the reconstruction error.

VGAE [14], a variational graph self-encoder, is an extension of the GAE model, uses a probabilistic generative model to model the representation of node embedding, and can learn embedding representations with randomness. The core idea of the model is to consider the representation of node embedding as a probability distribution and then learn how to sample the embedding representation of nodes from that distribution.

The DGI [15] model learns node representations by maximizing the mutual information between node features and graph structure, which in turn enables unsupervised graph node classification, node clustering, and other tasks. Its core idea is to utilize the idea of self-encoder, which takes the node features and graph structure as inputs, maps the nodes to a low-dimensional representation space through an encoder, and then reconstructs them through a decoder so that the reconstructed results are as close as possible to the original node features.

The GCA [16] model proposes a new graph comparison representation learning method with adaptive augmentation, which incorporates various priors on the topological and semantic aspects of the graph. The core idea is that an augmentation scheme based on a node centrality metric is designed at the topology level to highlight important connectivity structures. Node features are corrupted at the node attribute level by adding more noise to unimportant node features to force the model to recognize the underlying semantic information.

The COSTAMV [1] model proposes a new covariance-preserving feature space augmentation framework to perform augmentation operations on hidden features for use in graph contrastive learning.

In the model, Adam is uniformly used as an optimizer for training for the sake of experimental fairness. The learning rate is 0.001 for Citeseer, Pubmed, and DBLP datasets

and 0.0001 for Cora. The Drop Feature Rate and Edge Removing Rate are set differently depending on the dataset. The Hidden dimension is 256. The Epochs are 700, 1500, 1500, and 1000 for Cora, Citeseer, Pubmed, and DBLP, respectively. The Diffusion matrix parameter is fixed to 0.2. A summary of all the parameter settings is shown in Table 4.

Table 4. Algorithm parameter settings.

Dataset	Learning Rate	Drop Feature Rate	Edge Removing Rate	Training Epochs	Hidden Dimension	α_1
Cora	0.0001	0.3	0.4	700	256	0.2
Citeseer	0.001	0.4	0.2	1500	256	0.2
Pubmed	0.001	0.3	0.4	1500	256	0.2
DBLP	0.001	0.3	0.4	1000	256	0.2

4.5. Experimental Results and Analysis

4.5.1. Node Classification Task

The results of the node classification experiments comparing the model and the ten baseline models on the three citation network datasets, i.e., Cora, Citeseer, Pubmed, and DBLP, are shown in Table 5, with the highest experimental results in each dataset denoted in bold.

In order to better observe the experimental results, this section further represents the algorithm results in a bar chart as shown in Figure 5. From Table 5, it can be observed that all the algorithms in this paper achieve the best results compared to the baseline algorithm. On the Cora dataset, the algorithm improves by 3.3% on the Micro-F1 value over the COSTAMV algorithm, which has the best results. On the Citeseer dataset, the algorithm improves 0.1% on the Micro-F1 value over the COSTAMV algorithm, which has the best results. On the Pubmed dataset, the algorithm is similar to the results of the DGI and the GCA, but compared to the other algorithms, they show better performance; however, both DGI and GCA model training need a large number of negative samples, while the algorithm in this paper does not need negative samples for calculation. Thus, the algorithm does not need strong arithmetic power, and from this point of view, the algorithm is better than DGI and GCA algorithms. Finally, on the DBLP dataset, the algorithm improves by 0.9% compared with the most effective COSTAMV algorithm.

Although the algorithm slightly outperforms the suboptimal algorithm on the Citeseer, Pubmed, and DBLP datasets, the algorithm has a significant improvement on the Cora dataset, as well as the algorithm shows good algorithmic performance on all four datasets.

Table 5. Comparison of Micro-F1 values of node classification.

Method	Training Data	Cora	Citeseer	Pubmed	DBLP
Rawfeatures	X	64.8	64.6	84.8	71.6
Node2vec	A	74.8	52.3	80.3	78.8
DeepWalk	X,A	73.1	47.6	83.7	78.1
GCN	X,A,Y	82.8	72.0	84.9	82.7
SGC	X,A,Y	80.6	69.1	84.8	81.7
GAE	X,A	76.9	60.6	82.9	81.2
VGAE	X,A	78.9	61.2	83.0	81.7
DGI	X,A	82.6	68.8	86.0	83.2
GCA	X,A	82.8	71.5	86.0	83.1
COSTAMV	X,A	84.3	72.7	85.9	84.4
WNS-GCL	X,A	**87.1**	**72.8**	**86.0**	**85.3**

As can be seen in Figure 5, the performance obtained by using an objective function with no negative sample loss sometimes even exceeds that of a negative sample-based objective function, indicating a promising future direction, which provides a more efficient negative sample-free solution for subsequent graph contrastive learning. In contrast to

the negative sample-based objective, the Barlow Twins loss avoids the need for negative samples, thus significantly reducing the computational burden.

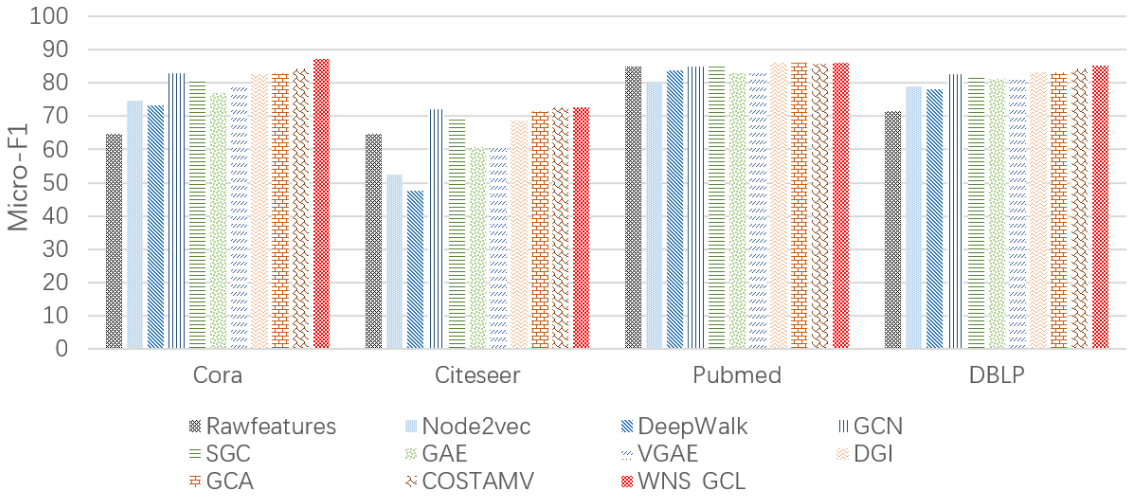

Figure 5. Micro-F1 of node classification experiment.

4.5.2. Ablation Experiments

The ablation experiments investigate the effect of different types of augmentation on the model, Feature Aug. (which refers to feature augmentation in implicit space) and Other Aug. (which refers to combinatorial augmentation performed on graphs as well as graph diffusion). In order to minimize factors other than the augmentation strategy that may affect the results, the other parameters of the model remain unchanged and only the augmentation method is replaced.

As shown in Table 6, it can be observed that the performance of each dataset is relatively low without using any data augmentation. This indicates that data augmentation methods have a positive effect on improving the model performance. When only feature augmentation (Feature Aug.) is used, the performance of each dataset is improved. This suggests that feature augmentation methods can help the model learn more effective feature representations and thus improve classification performance. When using only other types of data augmentation, the performance is also improved, especially on the DBLP dataset. This indicates that other types of data augmentation help the model to better capture the information in the graph structure. When both feature augmentation and other types of data augmentation are used, the highest performance is achieved on all datasets. This indicates that the combination of feature augmentation and other types of data augmentation methods can fully utilize the characteristics of the graph data to further improve the model performance. Therefore, the algorithm in this paper uses a combination of both augmentation methods.

Table 6. Comparison of Micro-F1 values of node classification with augmentation methods.

Other Aug.	Feature Aug.	Cora	Citeseer	Pubmed	DBLP
×	×	0.6912	0.6527	0.8424	0.7800
×	✓	0.7757	0.6916	0.8474	0.8049
✓	×	0.8162	0.7186	0.8525	0.8291
✓	✓	**0.8710**	**0.8710**	**0.8710**	**0.8710**

4.5.3. Parameter Sensitivity Analysis

This subsection focuses on the parameter sensitivity of the model. Specifically, the effects of changes in the node output dimension, learning rate, and diffusion coefficient size, as well as the parameters of the combined augmentation strategy on the node classification

performance, are investigated. Due to some similarities in different datasets, the Cora dataset is used as an example for illustration. In order to be fair, other parameters except test variables are kept constant during the experiment.

Node Output Dimension: the node output dimension directly affects the model performance. As shown in Figure 6, as the node representation dimension increases, the model performance first increases then stays the same, and then decreases. As the node output dimension increases from 128 to 512, the results of node categorization show an increasing trend up to 87.27%. This may be due to the fact that at lower dimensions, the model may have difficulty capturing the complex features and structural information of all the nodes in the graph. As the dimension increases, the model is able to learn richer node representations, which improves the classification results. When the node output dimensions continue to increase to 1024 and 2048, the effectiveness of node classification decreases instead. This may be due to the fact that too high dimension leads to model overfitting (i.e., the model is too sensitive to the training data) and the generalization performance on the test data decreases. Another reason could be that a too high dimension increases the computational complexity of the model, leading to a more difficult training process, which affects the final classification performance.

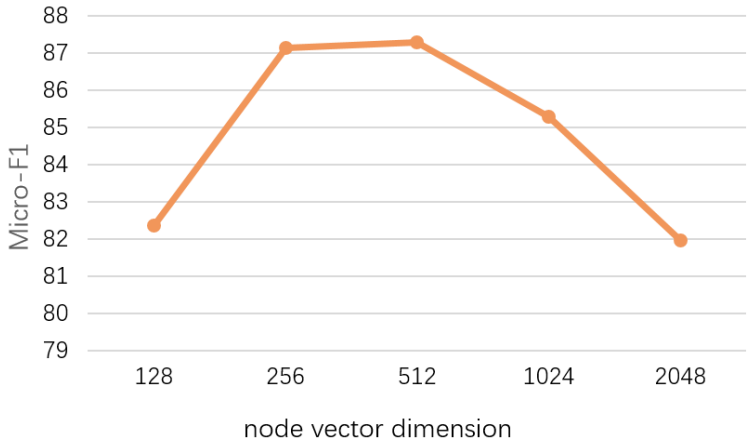

Figure 6. Influence of different output dimensions on node classification performance.

Learning rate: the learning rate has a certain impact on the node classification performance. Figure 7 shows the node classification performance with the size of the learning rate. It can be seen that as the learning rate increases, the node classification classification results show a weak increase followed by a decrease. When the learning rate increases from 0.0001 to 0.0003, the node classification accuracy improves and reaches a peak of 87.24%. This suggests that lower learning rates may prevent the model from converging to the optimal solution because smaller parameter updates require more iterations. On the contrary, a slightly higher learning rate allows the model to find a suitable local optimal solution more efficiently. When the learning rate was further increased to 0.0005, there was a slight decrease in classification accuracy. This decrease can be attributed to oscillations in the optimization process, which is caused by a learning rate that is too high to converge stably to a suitable local optimal solution. As the learning rate increases to 0.0007, 0.0009, and 0.001, the classification accuracy continues to decrease. This phenomenon may be caused by overfitting due to the high learning rate. As a result, the model is too sensitive to the training data, which reduces its generalization performance.

Diffusion coefficient: the effect of graph diffusion scale factor on node classification performance. When the input of the perspective is a graph diffusion matrix with different scales, the diffusion coefficient is analyzed. From Figure 8, it can be observed that with the gradual increase, the node classification performance shows a trend of increasing and then decreasing. At a diffusion coefficient of 0.2, the results of the node classification experiment reached the highest value of 87.13%. When the diffusion coefficient increases from 0.1 to 0.2,

the experimental results improve, which suggests that a larger diffusion coefficient in this range may help the model to capture more local and global information, thus improving the accuracy of node classification. However, when the diffusion coefficient continues to increase to 0.3 and above, we can see a gradual decrease in the node classification experimental results. This may be due to the fact that too large diffusion coefficients lead to an increase in noise in the information diffusion process, making it difficult for the model to capture effective features. In addition, too large a diffusion coefficient may lead to excessive diffusion of information, which makes local features become less obvious, thus affecting the classification performance.

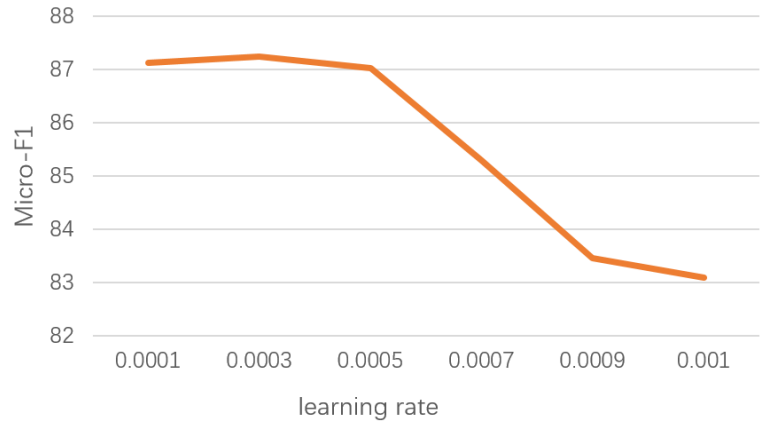

Figure 7. Influence of different learning rates on node classification performance.

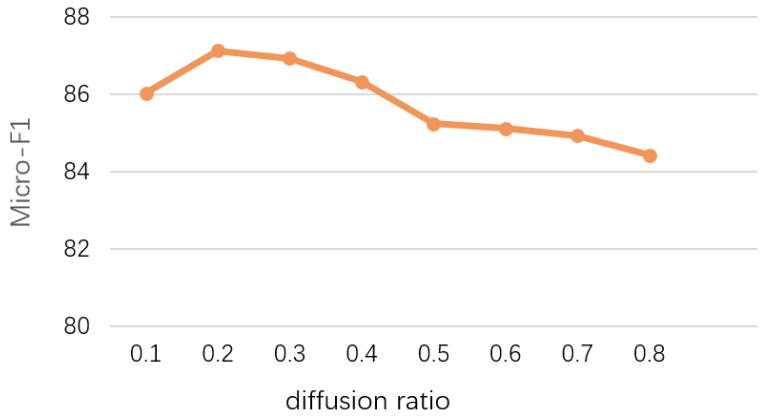

Figure 8. Influence of α on node classification performance.

Data augmentation method parameters: this section provides a sensitivity analysis of the main hyperparameters of the data augmentation algorithms, focusing on the impact of the control parameters pe and pf for the two data augmentation methods Feature Masking (FM) and Edge Removing (ER). This analytical process uses the Cora citation dataset and provides insights for the four parameters pe1 and pe2, as well as pf1 and pf2. In this section, node categorization experiments are conducted over a parameter range of 0.1 to 0.7, by which the impact of hyperparameters on accuracy can be more fully assessed. In order to control the augmentation at the topology level and node feature level, pe = pe1 = pe2 and pf = pf1 = pf2. In performing the sensitivity analysis, only these four parameters are adjusted in this section, keeping the other parameters stable. The related experimental results are shown in Figure 9. The horizontal axis represents the Edge Removing parameter probability and the vertical axis represents the Feature Masking parameter probability in Figure 9.

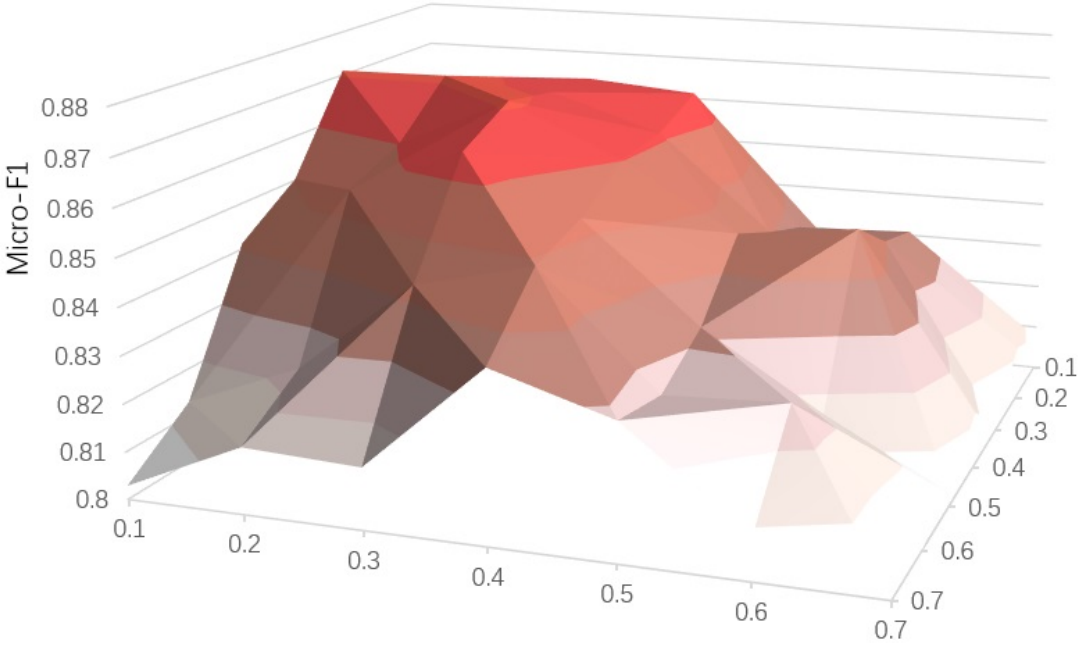

Figure 9. Influence of different hyperparameters on node classification performance.

Edge Removing parameter sensitivity analysis: as the Edge Removing parameter probability increases, we can observe the fluctuation of the classification accuracy under different Feature Masking parameters. At low Feature Masking parameter probabilities (e.g., 0.1–0.3), an increase in the Edge Removing parameter probability leads to a decrease in classification accuracy. This indicates that the model is more sensitive to the Edge Removing parameter, which may be due to the fact that removing more edges destroys the topology of the graph, making it difficult for the model to capture the structural information in the original data.

Sensitivity analysis of the Feature Masking parameter: with the increase in the probability of the Feature Masking parameter, we can observe the change of classification accuracy under different Edge Removing parameters. At low Edge Removing parameter probabilities (e.g., 0.1–0.3), an increase in the probability of the Feature Masking parameter leads to a decrease in classification accuracy. This indicates that the model is also more sensitive to the Feature Masking parameter, which may be due to the fact that masking more features makes it difficult for the model to obtain feature information in the original data.

Parameter combination sensitivity analysis: When the probability of Edge Removing and Feature Masking parameters increase simultaneously, the classification accuracy generally shows a decreasing trend. This indicates that the model is more sensitive under the simultaneous action of these two parameters, probably because the topology and node features of the graph are damaged at the same time, making it difficult for the model to capture the effective information in the original data. In practical applications, it is necessary to find a balance between these two parameters, so that the model can both learn the information from the original data and have a better generalization ability.

5. Conclusions

In this paper, the graph contrastive learning algorithm relies heavily on negative samples to ensure the effectiveness of the model, and the training algorithm requires a lot of arithmetic power. In order to solve the problem that the experimental results rely to a certain extent on the generation method of negative samples, as well as the problem of sample bias in node embedding caused by the data augmentation, the algorithm of this paper utilizes the way of matrix sketching in the implicit space for feature augmentation as well as using the objective function, which does not require negative samples and compares

the algorithms on the vector level. The objective function that does not require negative samples and compares at the vector level is used in four real network datasets to compare node classification experiments with 10 other baseline algorithms, and the Micro-F1 values are used to evaluate the effectiveness of the algorithms. The final experimental results prove that the algorithm in this paper generates higher-quality node embedding representations and achieves better results.

Author Contributions: Conceptualization, D.C.; Formal analysis, M.N., H.C. and D.W.; Funding acquisition, D.C. and D.W.; Methodology, D.C., M.N. and Z.W.; Project administration, D.C. and D.W.; Software, Z.W.; Supervision, D.C.; Visualization, M.N., Z.W. and H.C.; Writing—original draft, Z.W.; Writing—review and editing, D.C., M.N. and H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Applied Basic Research Project of Liaoning Province under Grant 2023JH2/101300185, in part by the Key Technologies Research and Development Program of Liaoning Province in China under Grant 2021JH1/10400079, and in part by the Natural Science Foundation of Liaoning Provincial Department of Science and Technology under Grant No. 2022-KF-11-04.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; King, I. COSTA: Covariance-preserving feature augmentation for graph contrastive learning. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2524–2534.
2. Bielak, P.; Kajdanowicz, T.; Chawla, N.V. Graph barlow twins: A self-supervised representation learning framework for graphs. *Knowl.-Based Syst.* **2022**, *256*, 109631. [CrossRef]
3. Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; Leskovec, J. Strategies for pre-training graph neural networks. *arXiv* **2019**, arXiv:1905.12265.
4. Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; Wang, L. Deep graph contrastive representation learning. *arXiv* **2020**, arXiv:2006.04131.
5. You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; Shen, Y. Graph contrastive learning with augmentations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 5812–5823.
6. Ren, Y.; Liu, B.; Huang, C.; Dai, P.; Bo, L.; Zhang, J. Heterogeneous deep graph infomax. *arXiv* **2019**, arXiv:1911.08538.
7. Zeng, J.; Xie, P. Contrastive self-supervised learning for graph classification. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 10824–10832. [CrossRef]
8. Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 23–27 August 2020; pp. 1150–1160.
9. Jiao, Y.; Xiong, Y.; Zhang, J.; Zhang, Y.; Zhang, T.; Zhu, Y. Sub-graph contrast for scalable self-supervised graph representation learning. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 222–231.
10. Zhang, M.; Hu, L.; Shi, C.; Wang, X. Adversarial label-flipping attack and defense for graph neural networks. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 791–800.
11. Liberty, E. Simple and deterministic matrix sketching. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 581–588.
12. Rozemberczki, B.; Kiss, O.; Sarkar, R. Karate Club: An API oriented open-source python framework for unsupervised learning on graphs. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual, 19–23 October 2020; pp. 3125–3132.
13. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying graph convolutional networks. In Proceedings of the International Conference on Machine Learning. PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6861–6871.
14. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
15. Veličković, P.; Fedus, W.; Hamilton, W.L.; Liò, P.; Bengio, Y.; Hjelm, R.D. Deep graph infomax. *arXiv* **2018**, arXiv:1809.10341.
16. Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; Wang, L. Graph contrastive learning with adaptive augmentation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2069–2080.