mathematics

IMPACT FACTOR 2.592

CITESCORE 2.9

Article

Subgraph Adaptive Structure-Aware Graph Contrastive Learning

Zhikui Chen, Yin Peng, Shuo Yu, Chen Cao and Feng Xia

Special Issue

Advances in Machine Learning Applied to Intelligent Systems and Data Analytics

Edited by

Dr. Linlin You, Dr. Ivan Lee and Dr. Zhicong Chen

MDPI

https://doi.org/10.3390/math10173047

mathematics

MDPI

Article

Subgraph Adaptive Structure-Aware Graph Contrastive Learning

Zhikui Chen [1], Yin Peng [1], Shuo Yu [2,*], Chen Cao [3] and Feng Xia [4]

[1] School of Software, Dalian University of Technology, Dalian 116620, China
[2] School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China
[3] Information Networking Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[4] Institute of Innovation, Science and Sustainability, Federation University Australia, Ballarat, VIC 3353, Australia
* Correspondence: yushuo@dlut.edu.cn

Abstract: Graph contrastive learning (GCL) has been subject to more attention and been widely applied to numerous graph learning tasks such as node classification and link prediction. Although it has achieved great success and even performed better than supervised methods in some tasks, most of them depend on node-level comparison, while ignoring the rich semantic information contained in graph topology, especially for social networks. However, a higher-level comparison requires subgraph construction and encoding, which remain unsolved. To address this problem, we propose a subgraph adaptive structure-aware graph contrastive learning method (PASCAL) in this work, which is a subgraph-level GCL method. In PASCAL, we construct subgraphs by merging all motifs that contain the target node. Then we encode them on the basis of motif number distribution to capture the rich information hidden in subgraphs. By incorporating motif information, PASCAL can capture richer semantic information hidden in local structures compared with other GCL methods. Extensive experiments on six benchmark datasets show that PASCAL outperforms state-of-art graph contrastive learning and supervised methods in most cases.

Keywords: graph contrastive learning; subgraph learning; network motif; unsupervised node classification; social network

MSC: 68T07; 05C62

Citation: Chen, Z.; Peng, Y.; Yu, S.; Cao, C.; Xia, F. Subgraph Adaptive Structure-Aware Graph Contrastive Learning. *Mathematics* **2022**, *10*, 3047. https://doi.org/10.3390/math10173047

Academic Editors: Mikhail Goubko, Pedro A. Castillo Valdivieso and Francesco Calimeri

Received: 3 June 2022
Accepted: 19 August 2022
Published: 24 August 2022

1. Introduction

Nowadays, deep learning technologies such as federated learning and reinforcement learning are widely used in various fields [1,2]. However, for graph learning, graph neural networks (GNNs) have gradually become the mainstream methods [3], e.g., GAT [4] and GraphSAGE [5], which have received considerable attention due to their outstanding performance in various tasks. Although GNNs have achieved great success, most of the existing GNNs are supervised methods and commonly rely on a large amount of labeled data. This is also one of the most widely acknowledged limitations of GNNs. Figure 1a,c represent the node embeddings of test nodes of GCN [6] and GCNII [7] when trained with 20 samples per class. Figure 1b,d, respectively, show the embeddings of test nodes when trained with 40% labeled data. Comparing the left and right columns of Figure 1, we can find that, depending on whether it is GCN or GCNII, the more labeled data used to train the model, the higher the quality of node embeddings learned by the model, which greatly limits the performance of GNNS in downstream tasks. Nowadays, although there are many data acquisition methods [8,9], we can easily obtain massive data for training models, but the data quality is often unsatisfactory, especially for social data [10]. On the one hand, the problem of incomplete data is widespread in practice [11]. On the other hand, data annotation is too expensive due to the fact that it requires lots of expertise in many areas [12]. In these contexts, it is difficult for GNNs to achieve excellent performance due

to the inability to obtain enough labeled data. Therefore, it is extremely meaningful and necessary to develop unsupervised graph representation learning methods.

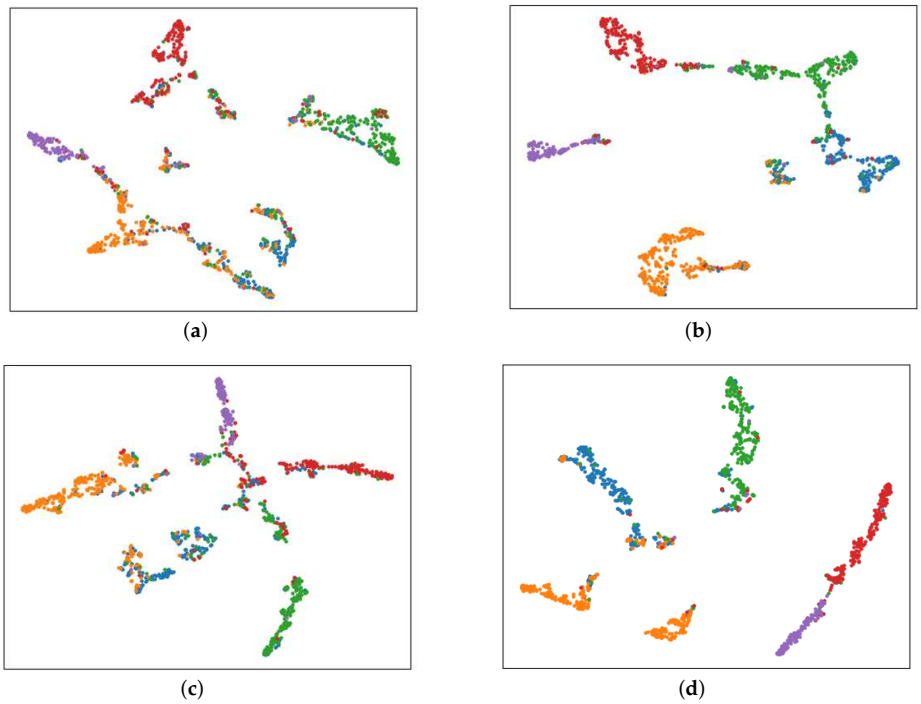


Figure 1. The testing node embeddings learned by GCN and GCNII on acmv9 when trained with 20 nodes per class and 40% nodes per class, visualized by t-SNE. (**a**) GCN: 20 nodes per class; (**b**) GCN: 40% nodes; (**c**) GCNII: 20 nodes per class; (**d**) GCNII: 40% nodes.

The main idea of previous unsupervised graph representation learning methods is to reconstruct the graph topology, such as CSADW [13], VGAE [14], etc. However, these methods overemphasize the proximity of graphs and perform unsatisfactorily in some contexts [15]. Unlike traditional grid data, such as images and texts, graphs are a non-euclidean form of structure data containing complex relational structures. Such structures generally have specific meanings in different graphs. For example, a triangle structure can be used to represent the ternary closure in a social network. Meanwhile, it can also represent a special chemical structure in a chemical molecular network. Therefore, subgraph-aware methods have been proposed to enhance the effectiveness of downstream tasks in graph representation learning.

Graph contrastive learning (GCL) is the most representative unsupervised graph learning method currently [16–18]. Unlike other deep learning techniques [19], the intuition behind GCL is to learn prior knowledge from the data itself by comparing different views of the original graph, which can be explained by mutual information (MI) and triplet loss [20]. However, most of the existing GCL methods require node–node-level comparison. In some scenarios, such as social networks, it is difficult to adequately capture the semantic information hidden in the local topology, resulting in sub-optimal performance. Although some subgraph-based GCL methods (e.g., [21,22]) have been proposed, the subgraph construction methods they employed failed to capture significant semantic information. Specifically, they usually use nearest neighbors or random walks to construct subgraphs. These methods are too simple to fully capture the structural information in some complex networks. Moreover, some of them have limitations in generalization because these methods can only be used for specific downstream tasks such as graph classification [23].

Our work. To solve the above problems, we propose a motif-based GCL method, entitled the subgra**p**h **a**daptive **s**tructure-aware graph **c**ontr**a**stive **l**earning model (PASCAL), used for unsupervised node classification tasks in this paper. Concretely, we first adaptively

extract subgraphs for each node based on its motif information to capture rich semantic information hidden in the local structure. Then, we use the feature masking and edge dropping augmentation strategies to generate two different graph views. Next, we use our proposed subgraph aggregation method to calculate subgraph embeddings. The subgraph embeddings are then regarded as node features fed to the next layer of GNNs. Finally, the graph encoder is optimized by maximizing the mutual information between the same nodes in different graph views. We conduct extensive experiments on various academic and social network datasets. Compared with previous methods, our proposed PASCAL performs better in most cases. The contributions of this work are summarized as follows:

- **Rich sentiment information representation.** We propose an effective motif-based graph contrastive learning method, called PASCAL, for unsupervised node classification tasks. PASCAL employs motifs to formulate certain patterns containing rich sentiment information, which significantly enhances the effectiveness of graph contrastive learning.
- **Subgraph aggregation and encoding strategy.** We propose a motif-based subgraph aggregation and encoding strategy, which is a play-and-pug component. The performance of models that imports the component can be significantly improved.
- **Explainable motif-aware model.** To prove the reliability and interpretability of the model, we analyze the Pearson correlation coefficient between the number of motifs and learned attention weights, and the learned attention weights are in line with our intuition. Moreover, we visualize the node embeddings by t-SNE, showing that our model is explainable and trustworthy.
- **Excellent performance on various datasets.** The effectiveness of PASCAL is shown through numerous experiments on various datasets including dblpv7, amazon-computers, etc. For the unsupervised node classification task, all methods are executed 20 times. No matter the optimal or average performance, PASCAL greatly outperforms all methods, including SOTA contrastive learning methods, under most settings.

In the following, we first introduce existing works and the preliminary of these works in Sections 2 and 3, respectively. Then, we show the details of PASCAL in Section 4. Section 5 introduces the experimental results. The effectiveness of our proposed motif-based subgraph aggregation strategy for semi-supervised models is implemented in Section 6. Moreover, we also analyze the learned attention weights in Section 6, which shows that our model is explainable.

2. Related Work

2.1. Graph Contrastive Learning

The main idea of graph contrastive learning is to maximize the mutual information between the anchor node and negative nodes. Deep graph infomax (DGI) [15] firstly learns node embeddings by maximizing the similarity between node embeddings and graph embeddings. However, DGI is a graph-level model which requires calculating the whole graph embedding. It is too expensive for large-scale graphs, thereby some node-level models are proposed [20,24]. Two strategies, i.e., adaptive negative sampling and data augmentation, have gradually proved their significance in enhancing the effectiveness of GCL. Adaptive augmentation can adaptively select the optimal one from a set of multiple augmentation strategies [25]. It also can be employed to dynamically design the optimal parameters for a special strategy instead of pre-defining them [16,26]. As for negative sampling, selecting the optimal negative sample (or high-quality ones) to calculate the contrast loss is the most significant method [27]. For example, SelfGNN [28] introduces the bootstrap your own latent (BYOL) mechanism into graph contrastive learning to avoid explicit negative sampling.

2.2. Motif-Based Graph Learning

The main purpose of motif-based graph learning is to capture high- or local-level structural information to improve model performance [29], such as [30,31]. An open question of motif-based graph learning is how to integrate motifs and graph learning

methods in a reasonable way. Some methods optimize existing models based on graph motifs [32–34]. In particular, Xia et al. [32] propose a motif-based high-order clustering algorithm that can effectively improve the clustering efficiency for large social networks. Some other methods regard graph motifs as auxiliary information to preprocess input graph [35–38]. For instance, Zhang et al. [36] designs a motif-based clustering algorithm, which divides the graph into several small networks for traffic speed prediction in the large urban traffic networks.

In general, under the unsupervised setting, it is more critical to capture the semantic information hidden in the graph topology as supervised signals to solve the problem of scarcity of labeled data. Therefore, we design an unsupervised model that can better capture graph structure information by combining graph contrastive learning with motif.

3. Preliminaries

3.1. Notation

Table 1 summarizes all notations used in this paper. Note that all bold notations represent a matrix or vector.

Table 1. Notations used in this paper.

Notation	Description
Network Related:	
$\mathcal{G}, \mathcal{G}^1, \mathcal{G}^2$	Input graph and two augmented graph views
V	The node set of $\mathcal{G}$
E	The edge set of $\mathcal{G}$
$\|V\|$	The number of nodes in $\mathcal{G}$
$\mathbf{A}, \mathbf{A}^1, \mathbf{A}^2$	The adjancy matrix of $\mathcal{G}$,$\mathcal{G}^1$,$\mathcal{G}^2$
$\mathbf{X}, \mathbf{V}, \mathbf{U}$	The feature matrix of $\mathcal{G}$,$\mathcal{G}^1$,$\mathcal{G}^2$
F	The dimention of nodes' input feature
C	The number of node categories
$\mathbf{Y} \in \mathbb{R}^{\|V\| \times C}$	The label matrix of $\mathcal{G}$
$\mathbf{H}^l$	The node embedding matrix in layer l
Motif Related:	
$\mathcal{M}$	Motif set we define in this paper
$M \in \mathcal{M}$	Some type of motif
m^t	The instance of t type motif
$\mathbf{M}$	Motif prototype vector
$\mathbf{m}$	The motif embedding
$\mathbf{P}$	Motif information of each node
$\mathbf{Q}$	The number matrix of each type of motif per node
$S^i, \mathbf{S}^i$	The motif-based subgraph centered on node v_i and its embedding
$\mathbf{S}^l$	The subgraph embedding matrix in layer l
Operation:	
$f_\phi(\cdot)$	The graph encoder with parameter ϕ
$\mathcal{T}_\alpha(\cdot)$	Augmentation function with parameter α
$\mathcal{J}(\cdot)$	The final loss function of the model
$\delta(\cdot,\cdot)$	The function that calculate the MI between inputs
$\mu(\cdot,\cdot)$	The cosine similarity function
$g(\cdot)$	Projection function
$D(\cdot)$	Contrastive loss of a positive pair
$Agg(\cdot)$	Aggregator for aggregating multi vectors
$Mean(\cdot)$	Mean Aggregator
$Att(\cdot)$	Aggregator based on attention mechanism

3.2. Problem Definition

Given an undirected graph $\mathcal{G} = (V, \mathbf{A}, \mathbf{X})$, where $V = \{v_1, v_2, \ldots, v_N\}$, $\mathbf{A} \in \{0,1\}^{N \times N}$, $\mathbf{X} \in \mathbb{R}^{|V| \times |F|}$ represent the node set, adjacency matrix, and node feature matrix, respectively. The goal of unsupervised node classification models is to train a graph encoder $f_\phi(\cdot)$

without using node labels. $\mathbf{H} = f_\phi(\mathbf{X}, \mathbf{A})$ represents the final learned node embedding matrix, which can be used to predict the label of nodes by a linear classifier or support vector machine trained by labeled data, i.e., $\widetilde{Y} = g_w(\mathbf{H})$, where $g_w(\cdot)$ represents the classifier.

3.3. Network Motif

The network motif is a special kind of low-order structure that hides rich semantic information and frequently occurs in the network [39]. Table 2 shows all types of third-order and fourth-order network motif. The third- and fourth-order network motifs indicate that the motif has three and four nodes respectively. In this paper, we use the first five predefined motifs [35] as auxiliary information for subgraph generation and aggregation, and their ids in the Table 2 range from 1 to 5. Moreover, we also regard ●—● as a special kind of motif in experiments. It is an edge in a graph, but herein we denote it as a second-order motif, so there are total six kinds of motifs used for subgraph generation and aggregation.

Table 2. All third-order and fourth-order motifs.

id	1	2	3	4	5	6	7	8
Motif								

4. The Design of PASCAL

We propose a motif-based graph contrastive learning method called PASCAL. As shown in Figure 2, PASCAL mainly consists five components, which are described in detail in Sections 4.1–4.5.

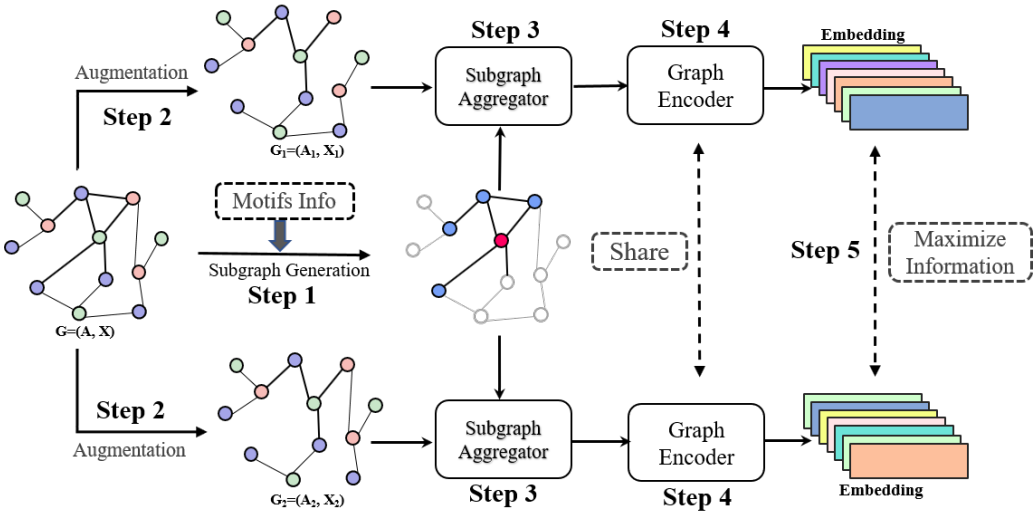


Figure 2. The over all architecture of the proposed PASCAL for unsupervised node classification, which uses feature masking as augmentation strategy. The model mainly consists 5 components: (1) subgraph generator; (2) augmentation; (3) subgraph aggregator; (4) graph encoder; and (5) comparator.

4.1. Subgraph Generator

In this work, we use pre-statistical node motif information to adaptively construct subgraphs for each node separately. As shown in Figure 3, we first find all motifs related to target node v_i, denoted by $M_i = \{m_1, \ldots, m_n\}$, where $m_i = \{v_i, v_j \mid v_j \in V, i \neq j\}$ represents a motif that containing v_i. Subsequently, we incroporate all of these motifs together as the motif-based subgraph centered on node v_i. The final subgraph centered on node v_i is represented as:

$$S_i = \{n_i \mid n_i \in m_j, m_j \in M_i\} \tag{1}$$

where n_i represents the ith node in motifs. Algorithm 1 is the pseudocode of the subgraph construction.

Algorithm 1 Subgraph construction.

Input: Target node v_i, the set of motifs containing the target node $M_i = \{m_1, \ldots, m_n\}$, node set $\mathcal{V}$, edge set $\mathcal{E}$.

1: **for** m in M_i **do**
2: Add all nodes of m to $\mathcal{V}$
3: Add all edges of m to $\mathcal{E}$
4: **end for**
5: Subgraph $G = (V', E')$, where $V' = \{v_i | v_i \in \mathcal{V}\}$ and $E' = \{e_j | e_j \in \mathcal{E}\}$

Output: Subgraph G

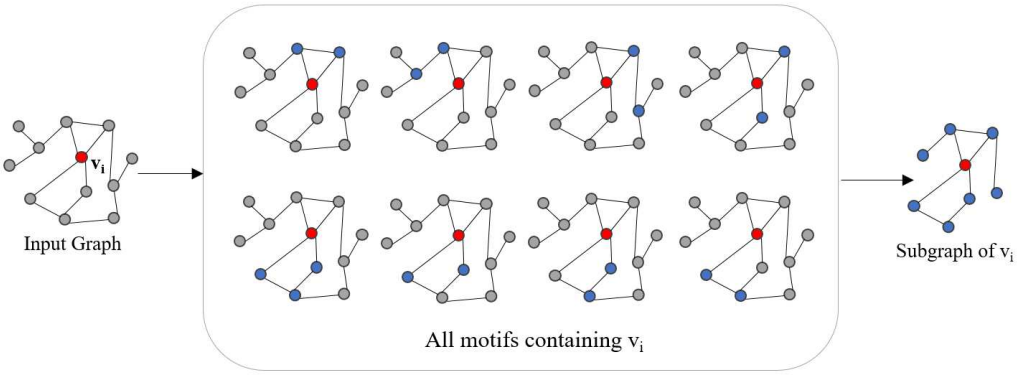


Figure 3. The process of generating subgraph for node v_i on the basis of its motif information. The red node represents the target node v_i and the blue nodes represent the nodes that appear in the same motif as v_i. For all networks in the middle box, the blue and red nodes represent all motifs containing the target node.

4.2. Augmentation

We use two augmentation strategies that are commonly used in GCL, feature masking and edge dropping, to generate two different graph views.

Edge Dropping. All edges of the input graph are dropped with a fixed probability. Formally, given a graph $\mathcal{G} = (V, E, \mathbf{A}, \mathbf{X})$, we first randomly sample a mask matrix $\mathbf{R} \in \{0,1\}^{|V|\times|V|}$, which follows a Bernoulli distribution $R_{ij} \sim \mathcal{B}(1 - p_r)$ if $\mathbf{A}_{ij} = 1$ for the input graph, or otherwise $R_{ij} = 0$. The p_r represents the probability of dropping edges. The adjacency matrix of the augmented graph is computed by Equation (2).

$$\mathcal{A} = A \circ R \tag{2}$$

where $\circ$ represents element-wise product.

Feature Masking. We randomly mask some dimensions of the input node features with zeros. To be specific, $\mathbf{X}$ represents the original feature matrix, we first randomly sample a vector $\tilde{\boldsymbol{m}} \in \{0,1\}^F$, where each dimension of it independently follows a Bernoulli distribution with probability $1 - p_m$, i.e., $\tilde{m}_i \sim \mathcal{B}(1 - p_m), \forall i$. The feature matrixes of the two views are computed by Equation (3).

$$\begin{aligned} \boldsymbol{V} &= [\boldsymbol{x}_1 \circ \tilde{\boldsymbol{m}}_{\boldsymbol{11}}; \boldsymbol{x}_2 \circ \tilde{\boldsymbol{m}}_{\boldsymbol{12}}; \cdots; \boldsymbol{x}_{1N} \circ \tilde{\boldsymbol{m}}_{\boldsymbol{1N}}]^T \\ \boldsymbol{U} &= [\boldsymbol{x}_1 \circ \tilde{\boldsymbol{m}}_{\boldsymbol{21}}; \boldsymbol{x}_2 \circ \tilde{\boldsymbol{m}}_{\boldsymbol{22}}; \cdots; \boldsymbol{x}_{2N} \circ \tilde{\boldsymbol{m}}_{\boldsymbol{2N}}]^T \end{aligned} \tag{3}$$

Algothrim 2 summarizes the graph augmentation process of PASCAL.

Algorithm 2 Graph augmentation.

Input: Input graph $G = (V, E, \mathbf{A}, \mathbf{X})$, drop edge probability p_r, and mask feature probability p_m.

Construct two empty network $\mathcal{G}_1, \mathcal{G}_2$
for $i = 1$ to 2 **do**
  Sample a mask matrix $\mathbf{R} \in \{0,1\}^{|V|\times|V|}$, where $R_{ij} \sim \mathcal{B}(1-p_r)$
  $\mathbf{A}' = \mathbf{A} \circ \mathbf{R}$
  $\mathbf{X}'$ is the augmented feature matrix
  for node v in V **do**
    Sample a mask vector $\tilde{\boldsymbol{m}} \in \{0,1\}^F$, where each dimension of it independently follows a Bernoulli distribution with probability $1 - p_m$
    $\mathbf{X}'_v = \mathbf{X}_v \circ \tilde{\boldsymbol{m}}$, representing the augmented feature of v
  end for
  Augmented graph $\mathcal{G}_i = (V, E', \mathbf{A}', \mathbf{X}')$
end for

Output: Augmented graph $\mathcal{G}_1, \mathcal{G}_2$

4.3. Subgraph Aggregator

How to construct and encode subgraphs is the key to subgraph-level GCL. In this work, we design a motif-based subgraph aggregator to calculate the subgraph embeddings, which are regared as node features fed into the graph encoder. Specifically, for each node v_i, the motifs set containing v_i is represented as $M_i = \{M_{i1}, M_{i2}, \ldots, M_{it}\}$, where $M_{ij} = \{m_{ij}^1, m_{ij}^2, \ldots, m_{ij}^n\}$, and t is the number of motif types. The subscript j represents the different kinds of motif defined in Section 3.3. Our proposed motif-based subgraph aggregate strategy consists of the following three steps:

(1) For each motif $m_{ij}^t \in M_{ij}$, we use a sum aggregator $Sum(\cdot)$ to compute the motif embedding, i.e., $\mathbf{m_{ij}^t} = Sum(m_{ij}^t) = \sum_{v_i \in m_{ij}^t} \mathbf{v}_i$.

(2) After Step 1, we can obtain all motif embeddings of type j containing v_i, denoted by $\mathbf{M_{ij}} = \{\mathbf{m_{ij}^1}, \ldots, \mathbf{m_{ij}^n}\}$. Then, we use a mean aggregator $Mean(\cdot)$ to aggregate all motif embeddings.The prototype of the j type motif containing v_i is represented by $\mathbf{m_{ij}} = Mean(\mathbf{M_{ij}})$.

(3) For all kinds of motif, Steps 1 and 2 are repeated. After obtaining all six kinds of motif embeddings containing v_i, denoted by $\mathbf{M_i}$, we use an aggregator $Agg(\cdot)$ to compute the final embedding of the subgraph that centered on v_i, denoted by $\mathbf{s_i} = Agg(\mathbf{M_i})$.

For the function $Agg(\cdot)$ used in Step 3, a mean or attention aggregator can be used. Given the motif prototypes, $\mathbf{M} \in \mathbb{R}^{|\mathcal{M}|\times N}$ where $|\mathcal{M}|$ represents the number of motif types and N represents the dimension of node embeddings. The formal definitions are, respectively, shown as follows:

- **Mean Aggregator.** The formula of $Mean(\cdot)$ is as follows:

$$Mean(\mathbf{M}) = \frac{\sum_{i=1}^{|M|} \mathbf{M_i}}{|M|} \tag{4}$$

- **Attention Aggregator.** We employ the attention mechanism used in UDAGCN [40] (shown in Equation (5)).

$$Att(\mathbf{M}) = Softmax(f(\mathbf{M})) \cdot \mathbf{M} \tag{5}$$

where $f(\cdot)$ and $Softmax(\cdot)$ represent the linear and softmax function, respectively. Algorithm 3 shows the process of subgraph aggregation.

Algorithm 3 Subgraph aggregation.

Input: Pre-statistical motif information M, input graph $G = (V, E, \mathbf{X})$.

$\mathbf{S} \in \mathbb{R}^{|V| \times h}$ represents the subgraph embedding matrix
for node v in V **do**
  $M_v \in M$ represents the motif information of v
  for each type t of motifs **do**
    M_v^t represents a set of motifs of type t containing v.
    Calculate each motif embedding by $Sum(\cdot)$, i.e., $\mathbf{m} = Sum(m) = \sum_{v_i \in m} \mathbf{v}_i$
    Calculate the prototype of each motif by $Mean(\cdot)$
  end for
  Aggregate all motif prototypes as subgraph embedding, i.e., $\mathbf{S}_v = Agg(\{\mathbf{m_1}, .., \mathbf{m_n}\})$
end for

Output: Subgraph embedding matrix $\mathbf{S}$

Figure 4 shows the process of calculating the triangle motif's prototype of v_i. In the calculation of motif embeddings and motif prototypes, we use $Sum(\cdot)$ and $Mean(\cdot)$ for aggregation, respectively. Therefore, in practice, we can simplify the calculation process of subgraph embeddings to matrix multiplication.

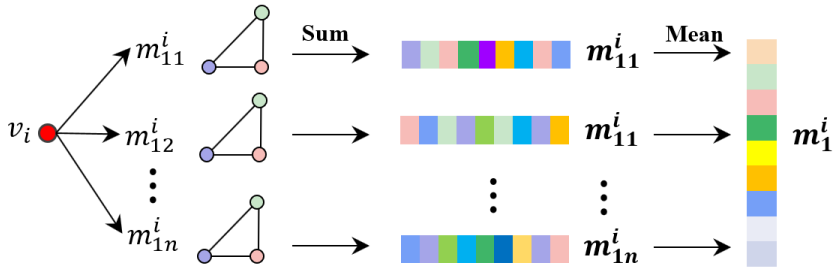


Figure 4. The process of calculating the triangle motif's prototype of v_i. m_{1j}^i represents the triangular motif containing the target node v_i. $\mathbf{m_{1j}^i}$ and $\mathbf{m_1^i}$ represent the motif embedding of m_{1j}^i and triangular motif prototype, respectively. "Sum" and "Mean" represent the computation of motif embedding and motif prototype using the su aggregator and mean aggregator, respectively.

Concretely, we define two matrices, $\mathbf{P} \in N^{|\mathcal{M}| \times |V| \times |V|}$ and $\mathbf{Q} \in N^{|\mathcal{M}| \times |V|}$, which represent all involved nodes and the number of motif type, respectively. $\mathbf{P}_{tij}$ represents the number of v_j appearing in the motif of type t containing v_i. Likewise, $\mathbf{Q}_{ti}$ denotes the number of motifs of type t containing v_i. The computing process of subgraph embeddings is formulated in Equation (6).

$$\mathbf{S} = Agg(\mathbf{PX} \ / \ \mathbf{Q}) \tag{6}$$

4.4. Graph Encoder

Two graph encoders, called PASCAL-concat and PASCAL-replace, respectively, are designed.

- **PASCAL-concat** adds an aggregation layer before message passing to compute the subgraph embedding S^l, which are regarded as node embeddings fed into the message passing layer. The feature update formulas for each layer of the graph encoder are as follows:

$$\begin{aligned} \mathbf{S^l} &= Agg(\mathbf{PH^l} \ / \ \mathbf{Q}) \\ \mathbf{H^{l+1}} &= \sigma(\mathbf{AS^l W^l}) \end{aligned} \tag{7}$$

- **PASCAL-replace** uses the subgraph aggregation to replace the original neighbor aggregation in GNN. Therefore, the adjacency matrix is useless in the graph encoder, as shown in Equation (8).

$$\mathbf{H^{l+1}} = \sigma(Agg(\mathbf{PH^l} \ / \ \mathbf{Q})\mathbf{W^l}) \tag{8}$$

where $\mathbf{W^l}$ is the learnable weight matrix of layer l.

For PASCAL-concat, we use both feature masking and edge dropping augmentation strategies at the same time. However, as the adjacency matrix is not used in PASCAL-replace, only the feature masking augmentation strategy is used.

4.5. Comparator

To train a graph encoder capturing rich local semantic information in an unsupervised manner, similar to GRACE [20], we define a contrastive objective to maximize the mutual information of the same node in two different graph views. Formally, we use $f_\phi(\cdot)$ to represent our motif-based graph encoder, and $\mathcal{G}^1 = (\mathbf{V}, \mathbf{A}^1), \mathcal{G}^2 = (\mathbf{U}, \mathbf{A}^2)$ denotes the two graph views, respectively. For better comparison, we use a projection function $g_\gamma(.) : \mathbb{R}^{n \times d_h} \rightarrow \mathbb{R}^{n \times d_h}$ to map the node embeddings of the two graph views to the same contrast space. For any node v_i, its embeddings in two views are denoted by $\mathbf{u_i}$ and $\mathbf{v_i}$, which are treated as the anchor and positive sample, respectively. The pairwise objective for each positive pair $(\mathbf{u_i}, \mathbf{v_i})$ is defined as Equation (9).

$$\mathcal{D}(\mathbf{u_i}, \mathbf{v_i}) = \log \frac{e^{\delta(\mathbf{u_i}, \mathbf{v_i})/\tau}}{\sum\limits_{k=1}^{N} e^{\delta(\mathbf{u_i}, \mathbf{v_k})/\tau} + \sum\limits_{\substack{k=1 \\ k \neq i}}^{N} e^{\delta(\mathbf{u_i}, \mathbf{u_k})/\tau}} \tag{9}$$

where $\delta(\mathbf{u}_*, \mathbf{v}_*) = \mu(g_\gamma(\mathbf{u}_*), g_\gamma(\mathbf{v}_*))$, and τ is a temperature parameter. The distance function used in $\mu(\cdot, \cdot)$ is the cosine similarity. Moreover, instead of deliberately choosing negative samples for the anchor, we treat all other nodes in the two graph views as negative samples. As two views are symmetric, the definition of $\mathcal{D}(\mathbf{v}_*, \mathbf{u}_*)$ is similar to $\mathcal{D}(\mathbf{u}_*, \mathbf{v}_*)$. Therefore, the final loss of PASCAL is defined as follows:

$$\mathcal{J} = \frac{1}{2N} \sum_{i=1}^{N} [\mathcal{D}(\mathbf{u_i}, \mathbf{v_i}) + \mathcal{D}(\mathbf{v_i}, \mathbf{u_i})] \tag{10}$$

Overall, in PASCAL, given the input graph and pre-statistical motif information, we first extract subgraphs for each node, and then perform graph augmentation to obtain different graph views and encode subgraphs based on motif information, and finally optimize the graph encoder by maximizing the mutual information between the same node in different views. The pseudocode of PASCAL is summarized in Algorithm 4.

Algorithm 4 PASCAL-replace algorithm.

Input: Input graph $G = (\mathbf{A}, \mathbf{X})$, motif info $\mathbf{P}$ and $\mathbf{Q}$, graph encoders f_ϕ, projection function g_θ, discriminator Θ, loss J. and augmentation function $\mathcal{T}$.

for $epoch = 1$ to n **do**
  for $i = 1$ to 2 **do**
    $G_i = T_\alpha(G) = (A_i, X_i)$
    $H_i^0 = X_i$
    for $l = 0$ to k **do**
      $\mathbf{S}_i^l = Agg(\mathbf{P} \times \mathbf{H}_i^l \ / \ \mathbf{Q})$
      $\mathbf{H}_i^{l+1} = \sigma(\mathbf{A}\mathbf{S}^l\mathbf{W}^l)$
    end for
  end for
  $\nabla_{\theta,\phi}\mathcal{J} = \nabla_{\theta,\phi} \frac{1}{2N} \sum_{i=1}^{N} \left[D(\mathbf{H}_1^l, \mathbf{H}_2^l) + D\left(\mathbf{H}_2^l, \mathbf{H}_1^l\right) \right]$
end for

Output: $\mathbf{H} = f(G)$

5. Experiments

5.1. Experimental Settings

5.1.1. Datasets

To achieve comprehensive comparison, we conduct unsupervised node classification experiments on six datasets, which can be categorized into two groups: academic and social networks.

- **Academic networks.** Citationv1, DBLPv7, and ACMv9 are three citation networks extracted from Microsoft Academic Graph, DBLP Computer Science Bibliography, and the Association for Computer Machinery, respectively [41]. These three datasets have five types of node labels. Totally, they have 8779, 5469, and 8769 nodes, with 13,590, 8090, and 14,798 edges, respectively.
- **Social networks.** Polblogs [42] is a directed network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance, which contains two categories of 1224 nodes, and 16,718 edges. In this paper, we treat it as an undirected graph. Amazon-computers and Amazon-photo [43,44] are segments of the Amazon co-purchase graph, which contain 10 and 8 kinds of nodes, respectively. The nodes and edges, respectively, represent the goods and the frequency by which two goods are bought together.

Detailed information of the six datasets is summarized in Table 3. The data of last five rows represent the average number of each motif per node, from which we can find that the five kinds of motifs defined in this paper frequently occur in networks.

Table 3. Dataset statistics. "**#Node** " and "**#Edge**" represent the total number of nodes and edges. "**#Classes**" is the number of node types. "**#M1_AVG**"~"**#M2_AVG**" represent the average number of motifs with per node in the motifs with id 1~5 in Table 2.

Dataset	Acmv9	Dblpv7	Citationv1	Polblogs	Computers	Photo
#Node	8779	5469	8769	1224	13,752	7650
#Edge	13,590	8098	14,798	16,718	245,861	119,081
#Classes	5	5	5	2	10	8
#M1_AVG	1.85	1.70	1.97	248	333	39
#M2_AVG	17.42	22.85	33.10	2545	8278	3915
#M3_AVG	1.23	1.26	1.26	1382	1947	1741
#M4_AVG	3.45	3.53	4.50	9074	26,887	13,875
#M5_AVG	3.04	3.05	3.30	3689	8822	4045

5.1.2. Baselines

The baselines can be categorized into two types: unsupervised and supervised methods. For supervised baselines, we choose GCN [6], SGC [45], GCNII [7], and MORE [35]. As for unsupervised methods, we regard GAE [14], GRACE [20], MVGRL [24], and DGI [15] as baselines. The details of these methods are as follows:

- GCN [6]: It is a classic semi-supervised GNNs method which learns the latent graph representation by extending the convolutional neural network to graph structure data and is widely used in various fields.
- SGC [45]: SGC transforms the nonlinear GCN into a simple linear model, which reduces the extra complexity of GCNs by repeatedly eliminating the nonlinearity between GCN layers and folding the resulting function into a linear transformation.
- GCNII [7]: It solves the over-smoothing problem of GNNs by using residual connection and identity mapping, which greatly improve the performance of GNNs.
- MORE [35]: MORE is a motif-based graph learning method, which regards the motif information as additional attribute information of nodes, used for social networks. The general idea of it is close to this work and its performance on social networks is very comparative.

- GAE [14]: GAE is an unsupervised graph learning method based on autoencoders, which learns node representation by reconstructing graph structure.
- DGI [15]: Different from traditional reconstruction-based unsupervised methods, DGI learns node embedding by maximizing the mutual information between the input and the output. DGI is groundbreaking graph contrastive learning algorithm and it also has top-ranked performance.
- GRACE [20]: It is a cutting-edge unsupervised graph representation learning method based on contrastive learning. GRACE is also the fundamental basis of our proposed method.
- MVGRL [24]: MVGRL uses graph diffusion for graph augmentation, and then compares the node and graph embedding of different views. It is one of the SOTA graph contrastive learning methods.

5.1.3. Experimental Details

To ensure the fairness of experiments, all unsupervised methods used in this paper employ a linear classifier to predict the label of nodes. We set the maximum epoch to 2000 and tolerance to 20, respectively. After the model is fitted, we use 10% data to train the classifier, and the remaining 90% data are used for testing. If there is no special statement, the graph encoder is a two-layer GNN, and the node embedding dimension is 128 for all datasets. We use the Adam with a learning rate of 0.001 to optimize the model. The classifier used to predict the node label uses a linear classifier or a support vector machine.

For supervised algorithms, the hyperparameters on all datasets are recommended by the original paper, and the node embedding dimension is 128. As for the dataset division, to facilitate comparison, we adopt the classic dataset division method used in GCN [6], that is, 20 samples of each class are used for training, 500 samples are used for validation, and another 1000 samples are used for testing. To prevent overfitting, we set tolerance to 20, and the maximum epoch to 1000. Our code is developed based on Python3.7 and Pytorch 1.7.0+cu101. Our model is trained by a V100 with 32G memory.

5.2. Unsupervised Node Classification

For PASCAL, we use the mean aggregator shown in Section 4.2 as the embedding aggregator. All methods are executed 20 times on each dataset, and the comparison results on six datasets are summarized in Table 4.

Table 4. Summary of node classification results on 6 datasets. All experiments are carried out 20 times, and "best" and "avg", respectively, represent the best and average performance of the model. "PASCAL-replace" and "PASCAL-concate" represent the two PASCAL variants mentioned in Section 4.4 using different types of encoders. Bold numbers represent the best results on different datasets.

Methods	acmv9		dblpv7		citationv1		Computers		Photo		Polblogs	
	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg
GCN	66	62.85	67.6	64.94	67.4	66.09	81.6	72.68	91.4	86.59	95.3	94.84
SGC	58.4	56.64	60.4	59.04	63.3	61.13	64.1	59.43	84.1	81.7	95.6	94.79
GCNII	72.3	70.18	73.1	71.66	74.4	71.66	63.6	62.46	70.1	62.3	95.5	94.97
MORE	70.8	67.15	66.7	64.69	65.6	62.87	74.2	72.89	85.9	85.46	**95.8**	**95.66**
GAE	50.08	47.62	52.31	50.3	53.19	51.45	82.15	79.61	88.94	88.32	92.3	91.6
DGI	70.46	68.47	70.47	69.02	74.58	73.28	87.25	86.78	92.67	92.26	94.28	92.46
GRACE	70.72	69.17	70.4	67.9	74.63	73.18	77.78	72.35	85.01	82.21	91	89.4
MVGRL	71.19	69.79	72.74	70.58	77.25	76.18	88.02	**87.47**	93.49	**93.09**	92.52	90.61
PASCAL-replace	53.49	52.94	57.78	57.05	**85.39**	**85.07**	**90.08**	85.07	**94.81**	89.75	94.81	94.28
PASCAL-concat	**75.57**	**75.03**	**73.13**	**72.22**	79.03	78.36	81.26	80.92	89.01	88.71	95	94.58

Some findings can be obtained according to the experimental results in Table 4. First, compared with all unsupervised methods, our method achieves the best results in acmv9,

dblpv7, citationv1, and polblogs, with an average performance improvement of almost 4% compared with GRACE. Although our method performs weaker than the SOTA unsupervised method on the computers and photo datasets, it still outperforms the GRACE using the same type of comparison framework, which proves the effectiveness of PASCAL. Second, under the dataset division settings used in GCN [6], our unsupervised method is superior to all supervised methods on each dataset. Third, we can find that the performance of PASCAL-replace is much worse than that of PASCAL-concat. The reason behind this phenomenon is that PASCAL-replace does not use graph adjacency matrix, which means that it only employs feature mask during data augmentation. Therefore, the two augmented views of PASCAL-replace are in low distinction so that the two views cannot be well contrasted. The semi-supervised experimental results in Section 6 further support our conjecture.

5.3. Ablation Studies

In this section, we discuss the performance of different variants of PASCAL.

Mean-Agg v.s Att-Agg. When aggregating prototypes of different motifs, we can use the mean aggregator or the attention aggregator. To compare the impact of different aggregators on model performance, we choose the attention mechanism used in UDAGCN [40] to aggregate multiple vectors, and both methods use the concat as the main framework. The experimental results are shown in Figure 5. We find that the model performed comparably in the two different aggregation methods, which shows that our proposed motif-based subgraph aggregation strategy is effective and reliable.

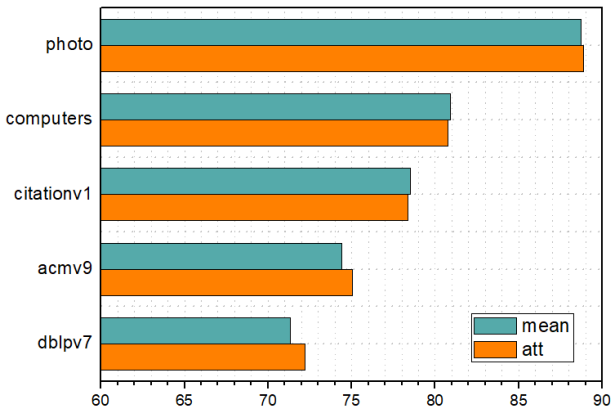


Figure 5. The performance comparison when using different aggregators to formulate motif prototypes. "att" and "mean" represent the attention-based and mean-based aggregators, respectively. The horizontal axis represents the classification accuracy, and the vertical axis represents the dataset.

Motifs. In this part, we study the impact of different variants of our proposed motif-based subgraph aggregation strategy on model performance. Concretely, we explore four different combinations between "second-order" and "degree-agg". Here, "second-order" indicates whether to use second-order motif when generating subgraphs, i.e., ●—●. "degree-agg" means using a degree-based aggregation to calculate motif embeddings, instead the $Sum(\cdot)$ aggregator used in Section 4.3. If using $Sum(\cdot)$ to calculate motif embeddings, when two motifs A and B(the motifs with id 1 and 2 in Table 2) consist of the same three nodes at the same time, the embedding of them are same. Thus, we introduce a degree-based motif aggregation method. Specifically, the degree of nodes in the motif is regarded as weight, and the weighted sum of all node embeddings is regarded as the motif embedding.

For example, supposing two motifs A and B consist of the same three nodes v_1, v_2, v_3, represented as m_a and m_b. The motif embeddings of them are calculated by:

$$\begin{aligned} \mathbf{m_a} &= \mathbf{v_1} + 2\mathbf{v_2} + \mathbf{v_3} \\ \mathbf{m_b} &= 2(\mathbf{v_1} + \mathbf{v_2} + \mathbf{v_3}) \end{aligned} \tag{11}$$

As shown in Table 5, we can find that: (1) no matter which combination we use, our model performs better than GRACE; (2) in most cases, the combination of using second-order motif without using degree-agg has the best performance; and (3) using the second-order motif on most datasets can slightly improve the performance of the model.

Table 5. The performance of different variants of PASCAL. "second-order" indicates whether to use the second-order motif when constructing the subgraph. "degree-agg" indicates whether to consider node degree when calculating motif embedding. Here check mark means consider it and cross means not use it. "PASCAL-concat-mean" represents the PASCAL-concat variant in Section 4.4, which uses the mean aggregator in Section 4.3 to aggregate different types of motif prototypes. Bold numbers represent the best results on different datasets.

Methods	Second-Order	Degree-Agg	acmv9		citationv1		dblpv7		polblogs	
			best	avg	best	avg	best	avg	best	avg
GRACE	–	–	70.72	69.17	74.63	73.18	70.4	67.9	91	89.4
PASCAL-concat-mean	✓	✓	73.58	72.81	78.02	77.43	71.2	70.3	94.7	94.34
	✓	×	**75.57**	**75.03**	**79.03**	**78.36**	**73.13**	**72.22**	**95**	**94.58**
	×	✓	72.83	72.15	77.34	76.89	69.66	68.49	94.92	94.35
	×	×	74.5	73.65	78.19	77.64	70.16	69.04	94.86	94.51

Classifier. In all previous experiments, the unsupervised methods employ a simple linear classifier to predict node labels. In this section, we compare the effect of different classifiers on the performance of the model. Specifically, we compare the performance of models using the linear classifier with that using the support vector machine (SVM) and the support vector machine (SVM), and the results are shown in Table 6. We can find that: (1) for both GRACE and PASCAL, using more powerful SVM can significantly improve model performance; (2) even if using SVM, the performance of GRACE is weaker than the PASCAL-concat using linear classifier, which shows the power of our proposed model.

Table 6. The performance of GRACE and PASCAL-concat with different classifier. "Linear" and "SVM" represent the use of linear and SVM as node classifiers, respectively. Bold numbers represent the best results on different datasets.

Datasets	GRACE		PASCAL-Concat	
	Linear	SVM	Linear	SVM
acmv9	69.17	74.11	75.03	**77.56**
citationv1	73.18	75.83	78.36	**80.19**
dblpv7	67.9	71.37	72.22	**73.6**
computers	72.35	76.89	80.92	**82.96**
photo	82.21	84.55	88.71	**89.03**
polblogs	89.4	89.12	**94.58**	94.25

6. Discussion

Complexity Analysis. Here, we briefly analyze the time complexity of PASCAL and compare it with GCN and GRACE. Let $|E|$ represent the edge number in the graph; d be the embedding size; b and m, respectively, denote the batch size and the node number in a batch; γ denote the edge keep rate in PASCAL; and L represent the number of layers of the encoder. We compare them from four aspects, and Table 7 summarizes the comparison results.

- Preprocessing: GCN and GRACE do not need to preprocess data, while PASCAL needs to collect the motif information of each node which is one of disadvantages of it. However, the motif information only needs to be analyzed once; the cost of preprocessing is, therefore, acceptable.
- Adjacency Matrix: For GCN, the adjacency matrix has only $2|E|$ non-zero elements since no augmentation is required. GRACE and PASCAL are typical contrastive learning methods that need to generate two augmented views, so there are two adjacency matrices containing $2\gamma|E|$ non-zero elements.
- Encoder: All three models use a two-layer encoder architecture, so the time complexity is consistent.
- Loss: For GCN, the time complexity is $O(2bd)$. For GRACE and PASCAL, we only use a small amount of data to train a simple linear classifier, so the time complexity mainly depends on the contrastive loss. Both GRACE and PASCAL use other nodes in the other perspective as negative samples, so the complexity of contrastive loss is $O(bd + bmd)$.

In general, the time complexity of contrastive learning is higher than that of GCN. Compared with GRACE, the complexity of PASCAL is higher than the additional data preprocessing. However, compared with the significant performance of PASCAL, the time cost of data preprocessing is negligible.

Table 7. Analysis of time complexity.

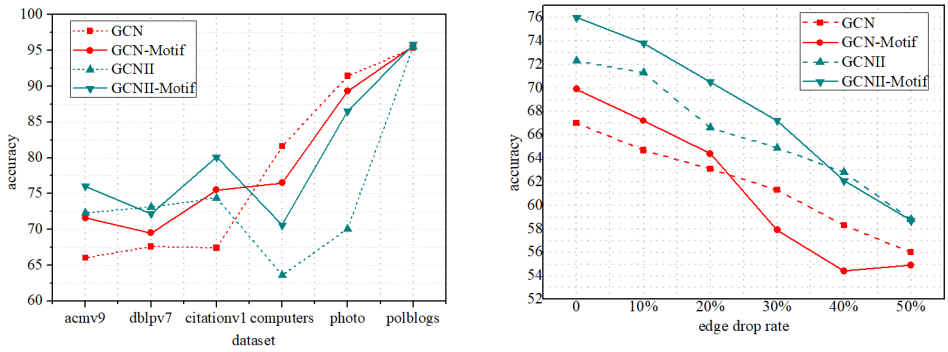
Datasets	GCN	GRACE	PASCAL
Preprocessing	-	-	O(2|E|)
Adjacency Matrix	O(2|E|)	O(4γ|E|)	O(4γ|E|)
Encoder	O(2|E|Ld)	O(8γ|E|Ld)	O(8γ|E|Ld)
Loss	O(2bd)	O(bd+bmd)	O(bd+bmd)

Attention weight. In the attention variant of PASCAL, we use the attention mechanism to aggregate different types of motifs. Here, we analyze the learned attention weights to explore something interesting. Specifically, we consider the relation of attention weights and the number of motifs using the Pearson correlation coefficient, which are used to measure the correlation between two variables. The results are summarized in Table 8. From Table 8, we can find that the number of nodes with correlation coefficients greater than 0.9 in acmv9, citationv1, and dblpv7 is much larger than that in the other three datasets, which means that the types with more motifs in the computers, photo, and polblogs may be assigned smaller weights. Actually, this phenomenon is normal. The distribution of the number of motifs in Table 3 shows that the distribution of motifs in these three datasets is more uneven. In this case, if the correlation coefficient is large, it will overly attenuate the influence of other motifs on the model, potentially reducing model performance. Therefore, the results in Table 8 are intuitive, indicating that the attention weights learned by the model are meaningful.

Table 8. The statistics of Pearson correlation coefficient between the number of motifs and the learned attention weights. "**#<0**" represents the number of nodes with coefficient less that 0, and the same for others. "**#Node**" represents the total number of nodes of datasets.

Datasets	#<0	#>0.3	#>0.5	#>0.9	#Node
acmv9	4060	2569	2112	1486	8779
citationv1	4496	2374	2007	1524	8769
dblpv7	2352	1658	1403	1074	5469
computers	11,432	3394	1914	52	13,752
photo	6784	1105	499	17	7650
polblogs	586	364	262	74	1224

Semi-supervised node classification. To further verify the power of our proposed motif-based subgraph aggregation strategy, we integrate it into GCN and GCNII in a concat manner for semi-supervised node classification tasks. Specifically, we first perform the motif-based subgraph aggregation on the output of the previous layer. Then, it is regarded as the updated node embeddings fed into the next GNN layer. Figure 6a,b, respectively, show the performance of the four models on the five datasets and incomplete Acmv9. From Figure 6a, we can find that on most of the datasets, the GCN-Motif and GCNII-Motif perform better, especially GCNII-Motif, which once again verifies the effectiveness of our proposed strategy. The performance of all models degrades as the ratio of missing edges increases. When the ratio of edge dropping is low, the performance of the improved model is still better than the original model. However, when the ratio of edge loss is too large ($\geq$30%), the performance of the improved model is comparable to or even worse than that of the original model, which is in line with our intuition. As the improved model is more dependent on the graph topology, the performance of the model will be affected more seriously if the original graph structure is excessively destroyed. Table 9 summarizes the GCN-Motif performance under the two integration modes of replace and concat. Unlike the unsupervised framework, in the semi-supervised framework, GCN-Motif-replace and GCN-Motif-concat perform comparably, which supports our conjecture in Section 5.2. Note that we use the Mean-Agg as the motif prototype aggregator for all experiments in this section.



(**a**) (**b**)

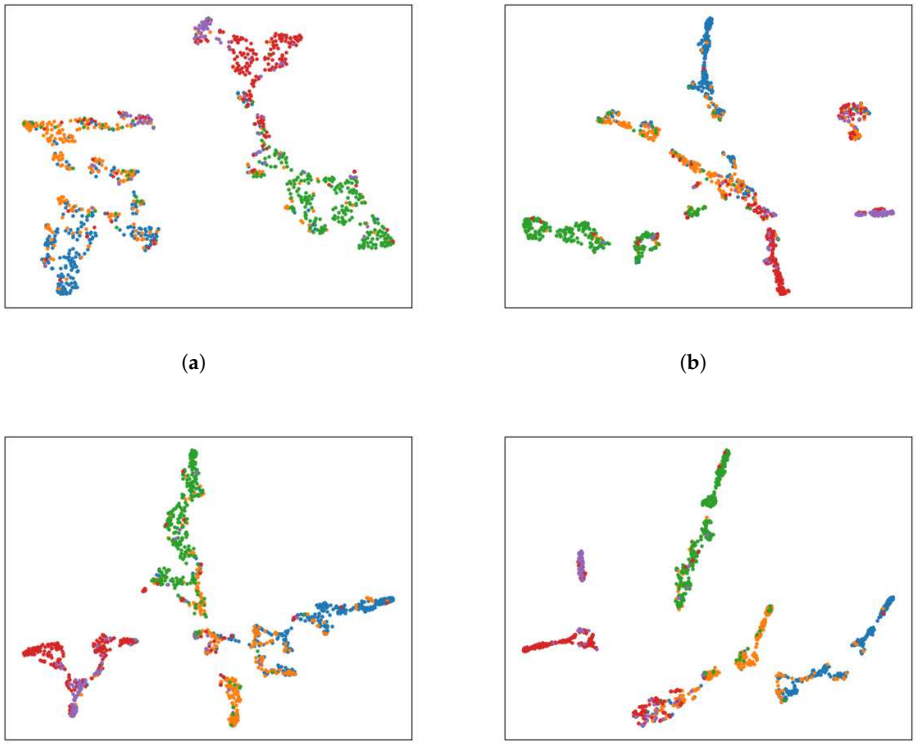
Figure 6. Performance comparison before and after integrating our proposed motif-based subgraph aggregation strategy. "GCN" and "GCNII" are two classic GNN models, respectively. "GCN-Motif" and "GCNII-Motif", respectively, represent the improved model based on the subgraph aggregation and encoding strategy proposed in this paper. The horizontal axis of (**b**) represents the random edge rate. (**a**) Node classification on the full dataset; (**b**) node classification on incomplete Acmv9.

Table 9. Performance of GCN when integrating different variants of our proposed motif-based subgraph aggregation strategy. "best" and "avg" represent the optimal and average performance over 20 experiments, respectively. Bold numbers represent the best results on different datasets.

Methods	acmv9		dblpv7		citationv1	
	best	avg	best	avg	best	avg
GCN	66	62.85	67.6	64.94	67.4	66.09
GCN-Motif-Replace	70.3	69.2	63.2	62.28	70.6	69.56
GCN-Motif-Concat	**71.6**	**69.28**	**69.5**	**68.3**	**75.5**	**72.6**

Visualization. To more intuitively show the effect of our proposed motif-based subgraph aggregation strategy on the GNNs framework, we use the tSNE algorithm to visualize the test set node embeddings learned by the model. Figure 7a,c show the test node embeddings of GCN and GCNII on citationv1, respectively. Figure 7b,d show the learned embeddings of GCN-Motif and GCNII-Motif, combined with our proposed subgraph

strategy on citationv1. Comparing the first and second columns of Figure 7, we find that the nodes of each category in the second column are more concentrated, and the classification boundaries are more obvious, which indicates that the quality of learned node embeddings is significantly improved through integrating our proposed subgraph strategy.



(**a**) (**b**)

(**c**) (**d**)

Figure 7. Visualization of test node embeddings of GCNII and GCNII-Motif on acmv9 and citationv1 by tSNE. xxx-Motif represents the GNNs integrated with our motif-based subgraph aggregation strategy. (**a**) GCN on citationv1; (**b**) GCN-Motif on citationv1; (**c**) GCNII on citationv1; (**d**) GCNII-Motif on citationv1.

7. Conclusions

In this work, we propose a structure-aware graph contrastive learning model called PASCAL which considers the subgraph-level embedding. PASCAL adaptively constructs and encodes subgraphs based on the nodes' motif information, and further uses them as the input of the GNN encoder to capture rich semantic information hidden in the local structure. Extensive experiments on six social and web benchmark datasets show the outperformance of PASCAL.

Although PASCAL performs well in unsupervised node classification tasks, it is not flawless. The motifs used in PASCAL are predefined, as it underperforms on some datasets such as Amazon Photo. The reason behind this phenomenon is the different distribution of motif types and numbers in different datasets. The five motifs predefined in this study may not be applicable to Amazon Photo. In future work, we will study how to automatically design and select the optimal motifs, which can significantly improve the generalization of PASCAL.

Author Contributions: Conceptualization, Z.C., Y.P. and F.X.; investigation, Y.P. and F.X.; methodology, Y.P., S.Y. and C.C.; supervision, Z.C., S.Y. and F.X.; validation, Z.C. and Y.P.; writing—original draft, Y.P. and Z.C.; writing—review and editing, Z.C., S.Y. and C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Key Research and Development Program of China under Grant No. 2021ZD0112400, the National Natural Science Foundation of China under Grant No. 62102060 and No. 62076047, and the Fundamental Research Funds for the Central Universities under Grant No. DUT22RC(3)060.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used in this paper are public, and you can download them from https://drive.google.com/drive/folders/1a1jSiw-2rxv9GClcE5WrEMA-GqTqUK-l?usp=sharing (accessed on 14 March 2022).

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

Abbreviations

The following abbreviations are used in this manuscript:

GCL	graph contrastive learning
GNN	graph neural network
GCN	graph convolution network
MI	mutual information
SOTA	state-of-the-art
PASCAL	subgraph adaptive structure-aware graph contrastive learning
DGI	deep graph infomax
BYOL	bootstrap your own latent

References

1. You, L.; Liu, S.; Chang, Y.; Yuen, C. A Triple-Step Asynchronous Federated Learning Mechanism for Client Activation, Interaction Optimization, and Aggregation Enhancement. *IEEE Internet Things J.* **2022**. [CrossRef]
2. Nasiri, E.; Berahmand, K.; Li, Y. A New Link Prediction in Multiplex Networks Using Topologically Biased Random Walks. *Chaos Solitons Fractals* **2021**, *151*, 111230.: 10.1016/j.chaos.2021.111230. [CrossRef]
3. Xia, F.; Sun, K.; Yu, S.; Aziz, A.; Wan, L.; Pan, S.; Liu, H. Graph Learning: A Survey. *IEEE Trans. Artif. Intell.* **2021**, *2*, 109–127. [CrossRef]
4. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
5. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1025–1035.
6. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
7. Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y. Simple and Deep Graph Convolutional Networks. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 1725–1735.
8. You, L.; Tunçer, B.; Xing, H. Harnessing Multi-Source Data about Public Sentiments and Activities for Informed Design. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 343–356. [CrossRef]
9. You, L.; Zhao, F.; Cheah, L.; Jeong, K.; Zegras, P.C.; Ben-Akiva, M. A Generic Future Mobility Sensing System for Travel Data Collection, Management, Fusion, and Visualization. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4149–4160. [CrossRef]
10. You, L.; Motta, G.; Liu, K.; Ma, T. CITY FEED: A Pilot System of Citizen-Sourcing for City Issue Management. *ACM Trans. Intell. Syst. Technol.* **2016**, *7*, 1–25. [CrossRef]
11. Lin, Z.; Kang, Z.; Zhang, L.; Tian, L. Multi-View Attributed Graph Clustering. *IEEE Trans. Knowl. Data Eng.* **2021**. [CrossRef]
12. You, L.; Motta, G.; Sacco, D.; Ma, T. Social Data Analysis Framework in Cloud and Mobility Analyzer for Smarter Cities. In Proceedings of the 2014 IEEE International Conference on Service Operations and Logistics, and Informatics, Qingdao, China, 8–10 October 2014; pp. 96–101.
13. Berahmand, K.; Nasiri, E.; Rostami, M.; Forouzandeh, S. A Modified DeepWalk Method for Link Prediction in Attributed Social Network. *Computing* **2021**, *103*, 2227–2249. [CrossRef]
14. Kipf, T.N.; Welling, M. Variational Graph Auto-Encoders. *arXiv* **2016**, arXiv:1611.07308.
15. Velickovic, P.; Fedus, W.; Hamilton, W.L.; Liò, P.; Bengio, Y.; Hjelm, R.D. Deep Graph Infomax. *ICLR (Poster)* **2019**, *2*, 4.
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.
17. Hafidi, H.; Ghogho, M.; Ciblat, P.; Swami, A. Graphcl: Contrastive Self-Supervised Learning of Graph Representations. *arXiv* **2020**, arXiv:2007.08025.
18. Pan, E.; Kang, Z. Multi-View Contrastive Graph Clustering. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 2148–2159.

19. You, L.; Tunçer, B.; Zhu, R.; Xing, H.; Yuen, C. A Synergetic Orchestration of Objects, Data, and Services to Enable Smart Cities. *IEEE Internet Things J.* **2019**, *6*, 10496–10507. [CrossRef]
20. Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; Wang, L. Deep Graph Contrastive Representation Learning. *arXiv* **2020**, arXiv:2006.04131.
21. 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, 6–10 July 2020; pp. 1150–1160.
22. Wang, C.; Liu, Z. Learning Graph Representation by Aggregating Subgraphs via Mutual Information Maximization. *arXiv* **2021**, arXiv:2103.13125.
23. Zhang, S.; Hu, Z.; Subramonian, A.; Sun, Y. Motif-Driven Contrastive Learning of Graph Representations. *arXiv* **2020**, arXiv:2012.12533.
24. Hassani, K.; Khasahmadi, A.H. Contrastive Multi-View Representation Learning on Graphs. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 4116–4126.
25. You, Y.; Chen, T.; Shen, Y.; Wang, Z. Graph Contrastive Learning Automated. *arXiv* **2021**, arXiv:2106.07594.
26. Suresh, S.; Li, P.; Hao, C.; Neville, J. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. *arXiv* **2021**, arXiv:2106.05819.
27. Lin, S.; Zhou, P.; Hu, Z.-Y.; Wang, S.; Zhao, R.; Zheng, Y.; Lin, L.; Xing, E.; Liang, X. Prototypical Graph Contrastive Learning. *arXiv* **2021**, arXiv:2106.09645.
28. Kefato, Z.T.; Girdzijauskas, S. Self-Supervised Graph Neural Networks without Explicit Negative Sampling. In Proceedings of the International Workshop on Self-Supervised Learning for the Web (SSL'21), at WWW'21, Singapore, 15–17 April 2021.
29. Yu, S.; Feng, Y.; Zhang, D.; Bedru, H.D.; Xu, B.; Xia, F. Motif Discovery in Networks: A Survey. *Comput. Sci. Rev.* **2020**, *37*, 100267. [CrossRef]
30. Peng, H.; Li, J.; Gong, Q.; Ning, Y.; Wang, S.; He, L. Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. In Proceedings of the AAAI, New York, NY, USA, 7–12 February 2020.
31. Cui, Z.; Cai, Y.; Wu, S.; Ma, X.; Wang, L. Motif-Aware Sequential Recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 11–15 July 2021; pp. 1738–1742.
32. Xia, F.; Yu, S.; Liu, C.; Li, J.; Lee, I. CHIEF: Clustering with Higher-Order Motifs in Big Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 990–1005 . [CrossRef]
33. Yu, S.; Xia, F.; Sun, Y.; Tang, T.; Yan, X.; Lee, I. Detecting Outlier Patterns with Query-Based Artificially Generated Searching Conditions. *IEEE Trans. Comput. Soc. Syst.* **2020**, *8*, 134–147. [CrossRef]
34. Zhang, K.; Yu, S.; Wan, L.; Li, J.; Xia, F. Predictive Representation Learning in Motif-Based Graph Networks. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Adelaide, Australia, 2–5 December 2019; pp. 177–188.
35. Xu, J.; Yu, S.; Sun, K.; Ren, J.; Lee, I.; Pan, S.; Xia, F. Multivariate Relations Aggregation Learning in Social Networks. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Wuhan, China, 1–5 August 2020; pp. 77–86.
36. Zhang, C.; Zhang, S.; James, J.; Yu, S. An Enhanced Motif Graph Clustering-Based Deep Learning Approach for Traffic Forecasting. In Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
37. Yu, S.; Xia, F.; Xu, J.; Chen, Z.; Lee, I. Offer: A Motif Dimensional Framework for Network Representation Learning. In Proceedings of the Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual, 19–23 October 2020; pp. 3349–3352.
38. Yu, S.; Xia, F.; Zhang, C.; Wei, H.; Keogh, K.; Chen, H. Familiarity-Based Collaborative Team Recognition in Academic Social Networks. *IEEE Trans. Comput. Soc. Syst.* **2021**, 1–14. [CrossRef]
39. Bedru, H.D.; Yu, S.; Xiao, X.; Zhang, D.; Wan, L.; Guo, H.; Xia, F. Big Networks: A Survey. *Comput. Sci. Rev.* **2020**, *37*, 100247. [CrossRef]
40. Wu, M.; Pan, S.; Zhou, C.; Chang, X.; Zhu, X. Unsupervised Domain Adaptive Graph Convolutional Networks. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 1457–1467.
41. Shen, X.; Dai, Q.; Mao, S.; Chung, F.; Choi, K.-S. Network Toher: Node Classification via Cross-Network Deep Network Embedding. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1935–1948. [CrossRef]
42. Adamic, L.A.; Glance, N. The Political Blogosphere and the 2004 US Election: Divided They Blog. In Proceedings of the 3rd International Workshop on Link Discovery, Chicago, IL, USA, 21–25 August 2005; pp. 36–43.
43. Shchur, O.; Mumme, M.; Bojchevski, A.; Günnemann, S. Pitfalls of Graph Neural Network Evaluation. *arXiv* **2018**, arXiv:1811.05868.
44. Chien, E.; Peng, J.; Li, P.; Milenkovic, O. Adaptive Universal Generalized PageRank Graph Neural Network. In Proceedings of the International Conference on Learning Representations, Virtual, 26 April–1 May 2020.
45. 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, Long Beach, CA, USA, 9–15 June 2019; pp. 6861–6871.