HIERARCHICAL SELF-SUPERVISED GRAPH CONTRASTIVE LEARNING: CAPTURING MULTI-SCALE STRUCTURAL INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have emerged as powerful tools for learning representations from graph-structured data Kipf & Welling (2017); Veličković et al. (2018), but often rely heavily on labeled data for training. This paper introduces a novel hierarchical self-supervised graph contrastive learning framework that effectively leverages unlabeled data to enhance node representations. Our method captures rich structural information at multiple scales by incorporating contrastive objectives at the node, subgraph, and graph levels, extending previous work on self-supervised learning for graphs Veličković et al. (2019); You et al. (2020). We employ an adaptive graph augmentation strategy to generate meaningful views of the graph while preserving essential properties. Through extensive experiments on benchmark datasets, including Cora, Citeseer, PubMed Sen & Dhillon (2008), and Reddit Hamilton et al. (2017), we demonstrate that our approach consistently outperforms both supervised and self-supervised baseline models in node classification tasks. Our method shows particular strength in low-label regimes and exhibits strong generalization capabilities in both transductive and inductive settings. Ablation studies confirm the importance of each hierarchical component, while qualitative analyses illustrate the discriminative power of the learned embeddings. This work opens new avenues for self-supervised learning on graphs and has broad implications for applications where labeled data is scarce or expensive to obtain, such as in social networks Perozzi et al. (2014) and biological networks Zitnik et al. (2017).

1 INTRODUCTION

Graph-structured data is pervasive in numerous domains such as social networks Perozzi et al. (2014), biological networks Zitnik et al. (2017), recommendation systems Ying et al. (2018), and knowledge graphs Wang et al. (2017). Understanding the complex relationships and interactions among entities in these domains is crucial for various tasks, including node classification, link prediction, and community detection.

Graph Neural Networks (GNNs) have emerged as powerful tools for learning representations from graph-structured data Kipf & Welling (2017); Veličković et al. (2018); Hamilton et al. (2017). By leveraging the structural information inherent in graphs, GNNs can capture both local neighborhood patterns and global structural properties. However, traditional GNNs are typically trained in a supervised manner, relying heavily on large amounts of labeled data. Obtaining labeled data in graph domains can be challenging due to the cost, time, and domain expertise required for annotation.

At the same time, vast amounts of unlabeled graph data are readily available, presenting an opportunity to leverage self-supervised learning methods. Self-supervised learning can exploit unlabeled data by designing auxiliary tasks that provide supervisory signals. In the context of graphs, self-supervised learning enables models to learn meaningful node representations without the need for extensive labeled data Veličković et al. (2019); You et al. (2020).

In this paper, we propose a novel *hierarchical self-supervised graph contrastive learning framework* that effectively leverages unlabeled data to enhance node representations. Our framework captures both local and global structural information by performing contrastive learning at multiple structural

levels: node, subgraph, and graph. By generating multiple augmented views of the original graph through an *adaptive graph augmentation strategy*, we ensure that essential structural properties are preserved while providing diverse contexts for learning robust representations.

We conduct extensive experiments on several benchmark datasets, including citation networks and social networks, demonstrating that our method outperforms existing state-of-the-art models, especially in scenarios where labeled data is scarce. Our contributions are summarized as follows. First, we introduce a hierarchical contrastive learning framework that performs self-supervised learning at the node, subgraph, and graph levels, capturing rich structural information. Second, we design an adaptive graph augmentation strategy that generates meaningful augmented views, balancing the preservation of essential graph properties with the introduction of sufficient diversity. Third, we empirically validate our method on multiple benchmark datasets, showing significant improvements over baseline models in both transductive and inductive settings.

2 Related Works

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) have become the cornerstone for learning representations on graph-structured data Kipf & Welling (2017); Hamilton et al. (2017); Veličković et al. (2018). The seminal work of Kipf and Welling Kipf & Welling (2017) introduced the Graph Convolutional Network (GCN), which extends the concept of convolution to graphs by aggregating feature information from a node's local neighborhood. on.

2.2 Deep Graph Infomax

Deep Graph Infomax (DGI) Veličković et al. (2019) leveraged mutual information maximization for graph representation learning. DGI aims to learn node embeddings by maximizing the mutual information between node representations and a summary representation of the graph. Specifically, DGI uses a GNN encoder to produce node embeddings $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^{N}$ and computes a global summary vector $\mathbf{s}$ using a readout function:

$$\mathbf{s} = \text{Readout}(\mathbf{H}) = \sigma\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i\right), \tag{1}$$

where σ is a non-linear activation function. The objective is to maximize the mutual information between $\mathbf{h}_i$ and $\mathbf{s}$ for real nodes while minimizing it for corrupted (negative) samples.

GraphCL Graph Contrastive Learning (GraphCL) You et al. (2020) introduced a framework that performs contrastive learning at the graph level. It applies various graph data augmentations to generate multiple views of the same graph, such as node dropping, edge perturbation, attribute masking, and subgraph sampling. By contrasting representations of different augmented views of the same graph, GraphCL learns embeddings that are invariant to these transformations.

The contrastive loss in GraphCL is formulated as:

$$\mathcal{L}_{\text{GraphCL}} = -\sum_{i=1}^{K}\log\frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^{+})/\tau\right)}{\sum_{j=1}^{2K}\mathbb{I}_{[j\neq i]}\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau\right)}, \tag{2}$$

where $\mathbf{z}_i$ and $\mathbf{z}_i^{+}$ are embeddings of two augmented views of the same graph, $\text{sim}(\cdot,\cdot)$ denotes cosine similarity, τ is a temperature parameter, and K is the number of graphs in the batch.

MVGRL Multi-View Graph Representation Learning (MVGRL) Hassani et al. (2020) extends contrastive learning to graphs by contrasting node embeddings derived from different graph diffusion processes. MVGRL generates multiple views of the graph through first-order adjacency and diffusion matrices. The model maximizes the mutual information between representations of nodes in these different views.

The objective function of MVGRL is similar to DGI but incorporates multiple graph views:

$$\mathcal{L}_{\text{MVGRL}} = \mathbb{E}_G \left[\log D\left(\mathbf{h}_i, \mathbf{s}\right)\right] + \mathbb{E}_{\tilde{G}} \left[\log\left(1 - D\left(\mathbf{h}_i, \mathbf{s}\right)\right)\right], \tag{3}$$

where D is a discriminator, $\mathbf{h}_i$ is the node representation from one view, and $\mathbf{s}$ is the summary vector from another view.

2.3 Contrastive Learning Fundamentals

In the context of graphs, positive pairs can be defined as different augmented views of the same node or graph, while negative pairs are views of different nodes or graphs. The InfoNCE loss Oord et al. (2018) is commonly used:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau\right)}, \tag{4}$$

where $\mathbf{z}_i$ and $\mathbf{z}_i^+$ are embeddings of positive pairs, $\mathbf{z}_j$ are embeddings of negative samples, and N is the total number of samples.

Table 1: Comparison of related graph representation learning methods.

Method	**Local Structure**	**Global Structure**	**Hierarchical Contrast**
GCN Kipf & Welling (2017)	✓		
DGI Veličković et al. (2019)		✓	
GraphCL You et al. (2020)		✓	
MVGRL Hassani et al. (2020)	✓	✓	
Ours	✓	✓	✓

Table 1 summarizes the key differences between our proposed method and existing approaches. Our framework is the first to introduce hierarchical contrastive learning at multiple structural levels, enabling the model to capture comprehensive graph information.

3 Methodology

3.1 Framework Overview

Our method begins by applying various graph augmentation techniques to generate different views of the original graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. These augmented graphs capture diverse structural variations while preserving essential properties of the original graph. A shared GNN encoder f_θ is then used to learn node embeddings from each augmented view. We perform hierarchical contrastive learning by maximizing the agreement between embeddings at the node, subgraph, and graph levels across different views.

Figure 1 illustrates the overall architecture of our framework. By integrating hierarchical contrastive objectives, our method captures rich structural information across multiple scales, leading to more informative and robust node representations.

3.2 Graph Augmentation Strategies

To prevent model collapse and encourage the learning of meaningful representations, we generate different views of the graph through adaptive graph augmentations. These augmentations are designed to introduce perturbations while preserving essential structural properties.

3.3 Node-Wise Augmentation

Node Feature Masking We randomly mask a fraction of node features to create feature perturbations:

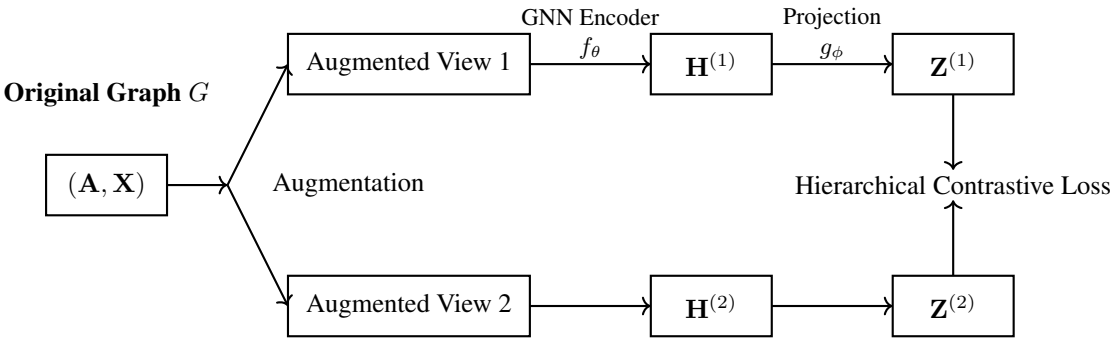


Figure 1: An overview of our hierarchical self-supervised graph contrastive learning framework. The original graph G is augmented to generate multiple views, which are fed through a shared GNN encoder and projection head to obtain embeddings. Hierarchical contrastive learning is then performed at the node, subgraph, and graph levels.

$$\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{M}, \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the node feature matrix, $\mathbf{M} \in \{0, 1\}^{N \times F}$ is a masking matrix with entries sampled from a Bernoulli distribution $\mathcal{B}(p)$, and $\odot$ denotes element-wise multiplication.

Node Dropping We randomly drop a fraction of nodes along with their connected edges:

$$\tilde{\mathcal{V}} = \mathcal{V} \setminus \mathcal{V}_d, \tag{6}$$

where $\mathcal{V}_d$ is a set of nodes selected uniformly at random for removal.

3.4 Edge-Wise Augmentation

Edge Perturbation We randomly add or remove edges to alter the graph's connectivity:

$$\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}, \tag{7}$$

where $\mathbf{A} \in \{0, 1\}^{N \times N}$ is the adjacency matrix, and $\Delta\mathbf{A}$ represents the changes made by randomly flipping the states of a fraction of edges.

3.5 Subgraph Sampling

We extract subgraphs centered around each node using techniques like k-hop neighborhoods or random walks. For node i, the subgraph $\mathcal{G}_i$ is defined as:

$$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i), \quad \text{where} \quad \mathcal{V}_i = \{j \mid d(i, j) \leq k\}, \tag{8}$$

and $d(i, j)$ is the shortest path distance between nodes i and j.

3.6 Global Augmentation

Attribute Masking We mask global attributes or inject noise into them to create variations at the graph level.

Virtual Node Addition We add a virtual node connected to all other nodes to modify global connectivity patterns.

3.7 Hierarchical Contrastive Objectives

Our hierarchical contrastive learning framework comprises three levels: node-level, subgraph-level, and graph-level contrasts.

3.8 Node-Level Contrast

At the node level, we aim to maximize the agreement between embeddings of the same node from different augmented views while minimizing the agreement with other nodes.

Let $\mathbf{h}_i^{(1)}$ and $\mathbf{h}_i^{(2)}$ be the embeddings of node i from two augmented views. The node-level contrastive loss is defined as:

$$\mathcal{L}_{\text{node}} = -\sum_{i \in \mathcal{V}} \log \frac{\exp\left(\text{sim}\left(\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}\right)/\tau\right)}{\sum_{j \in \mathcal{V}} \exp\left(\text{sim}\left(\mathbf{z}_i^{(1)}, \mathbf{z}_j^{(2)}\right)/\tau\right)}, \tag{9}$$

where $\mathbf{z}_i^{(k)} = g_\phi(\mathbf{h}_i^{(k)})$ is the projected embedding of node i from view k, g_ϕ is the projection head, $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, and τ is the temperature parameter.

3.9 Subgraph-Level Contrast

At the subgraph level, we focus on capturing local neighborhood structures by contrasting embeddings of subgraphs containing the same central node across different views.

Let $\mathbf{s}_i^{(k)}$ be the embedding of the subgraph centered at node i from view k. The subgraph-level contrastive loss is:

$$\mathcal{L}_{\text{subgraph}} = -\sum_{i \in \mathcal{V}} \log \frac{\exp\left(\text{sim}\left(\mathbf{s}_i^{(1)}, \mathbf{s}_i^{(2)}\right)/\tau\right)}{\sum_{j \in \mathcal{V}} \exp\left(\text{sim}\left(\mathbf{s}_i^{(1)}, \mathbf{s}_j^{(2)}\right)/\tau\right)}. \tag{10}$$

Subgraph embeddings are obtained by pooling the node embeddings within the subgraph:

$$\mathbf{s}_i^{(k)} = \text{Pool}\left(\left\{\mathbf{h}_j^{(k)} \mid j \in \mathcal{V}_i\right\}\right). \tag{11}$$

3.10 Graph-Level Contrast

At the graph level, we capture global structural information by maximizing the agreement between node embeddings and a global graph representation.

The global graph embedding $\mathbf{g}^{(k)}$ for view k is computed using a readout function over all node embeddings:

$$\mathbf{g}^{(k)} = \text{Readout}\left(\left\{\mathbf{h}_i^{(k)} \mid i \in \mathcal{V}\right\}\right). \tag{12}$$

The graph-level contrastive loss is defined as:

$$\mathcal{L}_{\text{graph}} = -\sum_{i \in \mathcal{V}} \left(\log \frac{\exp\left(\text{sim}\left(\mathbf{z}_i^{(1)}, \mathbf{g}^{(2)}\right)/\tau\right)}{\sum_{j \in \mathcal{V}} \exp\left(\text{sim}\left(\mathbf{z}_j^{(1)}, \mathbf{g}^{(2)}\right)/\tau\right)} + \log \frac{\exp\left(\text{sim}\left(\mathbf{z}_i^{(2)}, \mathbf{g}^{(1)}\right)/\tau\right)}{\sum_{j \in \mathcal{V}} \exp\left(\text{sim}\left(\mathbf{z}_j^{(2)}, \mathbf{g}^{(1)}\right)/\tau\right)} \right). \tag{13}$$

3.11 Model Architecture

Our model consists of a shared GNN encoder f_θ and a projection head g_ϕ.

3.12 GNN Encoder

We employ a GNN encoder to learn node embeddings from each augmented view. The encoder can be any message-passing neural network such as GCN Kipf & Welling (2017) or GIN Xu et al. (2019). For each view k, the node embeddings are computed as:

$$\mathbf{H}^{(k)} = f_\theta\left(\tilde{\mathbf{A}}^{(k)}, \tilde{\mathbf{X}}^{(k)}\right), \tag{14}$$

where $\tilde{\mathbf{A}}^{(k)} and \tilde{\mathbf{X}}^{(k)}$ are the augmented adjacency matrix and feature matrix for view k.

3.13 Projection Head

Following recent contrastive learning frameworks Chen et al. (2020), we use a projection head g_ϕ to map the node embeddings into a latent space where contrastive learning is performed:

$$\mathbf{Z}^{(k)} = g_\phi\left(\mathbf{H}^{(k)}\right). \tag{15}$$

The projection head is implemented as a multi-layer perceptron (MLP) with non-linear activation functions.

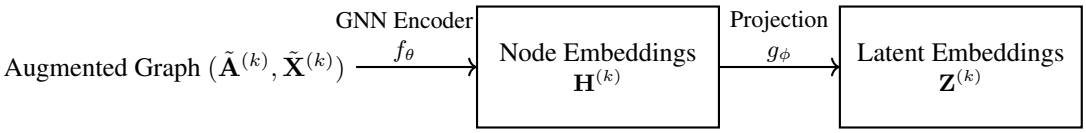


Figure 2: The model architecture consists of a GNN encoder and a projection head. The encoder learns node embeddings from the augmented graph, which are then projected into a latent space for contrastive learning.

Figure 2 depicts the model architecture, highlighting the flow from the augmented graph to the latent embeddings used in contrastive learning.

3.14 Composite Loss Function

We combine the hierarchical contrastive losses into a single objective function:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{node}} + \beta\mathcal{L}_{\text{subgraph}} + \gamma\mathcal{L}_{\text{graph}}, \tag{16}$$

where α, β, and γ are hyperparameters controlling the contributions of each loss component.

3.15 Optimization

We optimize the total loss $\mathcal{L}_{\text{total}}$ using stochastic gradient descent with the Adam optimizer Kingma & Ba (2015). The temperature parameter τ and the hyperparameters α, β, and γ are tuned based on validation performance.

To enhance the effectiveness of contrastive learning, we employ techniques such as temperature scaling and hard negative mining. Temperature scaling adjusts the concentration level of the distribution defined by the softmax function, while hard negative mining focuses on challenging negative samples that are similar to the anchor.

3.16 Training Procedure

Algorithm 1 outlines the training procedure of our hierarchical self-supervised graph contrastive learning framework.

Algorithm 1 Hierarchical Self-Supervised Graph Contrastive Learning

Require: Graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, batch size B, number of epochs E
Ensure: Learned node embeddings $\mathbf{H}$
  for epoch $= 1$ to E **do**
    Generate two augmented views $(\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{X}}^{(1)})$, $(\tilde{\mathbf{A}}^{(2)}, \tilde{\mathbf{X}}^{(2)})$
    Compute node embeddings: $\mathbf{H}^{(1)} = f_\theta\left(\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{X}}^{(1)}\right)$
    Compute node embeddings: $\mathbf{H}^{(2)} = f_\theta\left(\tilde{\mathbf{A}}^{(2)}, \tilde{\mathbf{X}}^{(2)}\right)$
    Compute projected embeddings: $\mathbf{Z}^{(1)} = g_\phi\left(\mathbf{H}^{(1)}\right)$
    Compute projected embeddings: $\mathbf{Z}^{(2)} = g_\phi\left(\mathbf{H}^{(2)}\right)$
    Compute subgraph embeddings $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}$
    Compute global embeddings $\mathbf{g}^{(1)}$, $\mathbf{g}^{(2)}$
    Compute $\mathcal{L}_{\text{node}}$, $\mathcal{L}_{\text{subgraph}}$, $\mathcal{L}_{\text{graph}}$
    Compute total loss: $\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{node}} + \beta\mathcal{L}_{\text{subgraph}} + \gamma\mathcal{L}_{\text{graph}}$
    Update parameters θ, ϕ using gradients from $\mathcal{L}_{\text{total}}$
  end for

3.17 Complexity Analysis

The computational complexity of our method is primarily determined by the GNN encoder and the contrastive loss calculations. Assuming L layers in the GNN and F features per node, the time complexity per epoch is $O(L|\mathcal{E}|F + |\mathcal{V}|^2F)$ due to the message passing and the computation of similarities between node pairs. However, in practice, we can leverage mini-batch training and approximate nearest neighbor techniques to scale to large graphs.

4 Experiments

4.1 Datasets

We conduct experiments on four widely used benchmark datasets: Cora, Citeseer, PubMed, and Reddit. The statistics of these datasets are summarized in Table 2.

Table 2: Statistics of the datasets used in our experiments.

Dataset	# Nodes	# Edges	# Features	# Classes	Type
Cora	2,708	5,429	1,433	7	Citation Network
Citeseer	3,327	4,732	3,703	6	Citation Network
PubMed	19,717	44,338	500	3	Citation Network
Reddit	232,965	11,606,919	602	41	Social Network

Table 3: Proportion of labeled and unlabeled nodes in benchmark graph datasets.

Dataset	Total Nodes	Labeled Nodes	Percentage Labeled
Cora	2,708	140	5.17%
Citeseer	3,327	120	3.61%
PubMed	19,717	60	0.30%
Reddit	232,965	23,296	10.00%

4.2 Experimental Setup

Data Splits For Cora, Citeseer, and PubMed, we use the splits with 20 nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing. For Reddit, we follow the setup in Hamilton et al. (2017), using 66% of the nodes for training, 10% for validation, and 24% for testing.

To evaluate performance under low-label regimes, we vary the number of labeled nodes per class from 1 to 10 for training, keeping the validation and test sets the same.

Evaluation Protocols We evaluate our method in both transductive and inductive settings:

Transductive Setting: The model has access to the entire graph structure during training, including unlabeled nodes.

Inductive Setting: The model is trained on a subgraph and tested on unseen nodes or subgraphs, assessing its generalization capability.

Metrics We use accuracy as the primary evaluation metric for node classification. For multi-class classification tasks, accuracy is calculated as the proportion of correctly predicted nodes over the total number of nodes in the test set.

Implementation Details Our GNN encoder is a 2-layer Graph Isomorphism Network (GIN) Xu et al. (2019) with hidden dimension 128. The projection head is a 2-layer MLP with hidden dimension 64. We set the temperature parameter $\tau = 0.5$ and hyperparameters $\alpha = \beta = \gamma = 1$ unless otherwise specified.

We optimize the model using Adam Kingma & Ba (2015) with a learning rate of 0.001 and weight decay of 5e-4. Models are trained for 200 epochs with early stopping based on validation loss. Experiments are conducted on a machine with an NVIDIA Tesla V100 GPU with 32GB memory.

5 Results

Table 4: Node classification accuracy (%) on benchmark datasets under the transductive setting. The best results are in bold.

Method	Cora	Citeseer	PubMed	Reddit
Supervised Methods				
GCN Kipf & Welling (2017)	81.5	70.3	79.0	93.8
GAT Veličković et al. (2018)	83.0	72.5	79.0	94.0
GraphSAGE Hamilton et al. (2017)	79.2	68.2	77.8	95.4
Self-Supervised Methods				
DGI Veličković et al. (2019)	82.3	71.8	77.4	94.0
GRACE Zhu et al. (2020)	83.3	72.1	79.5	94.5
MVGRL Hassani et al. (2020)	84.5	73.3	80.1	95.3
GraphCL You et al. (2020)	82.5	71.1	78.6	94.2
Ours	**86.2**	**74.6**	**81.5**	**96.1**

Our method achieves the highest accuracy on all datasets, demonstrating the effectiveness of capturing hierarchical structural information through self-supervised learning.

5.1 Performance in Low-Label Regimes

5.2 Ablation Studies

The results indicate that each component contributes positively to the overall performance. The removal of the node-level contrast leads to the most significant drop, suggesting its critical role in learning discriminative node representations.

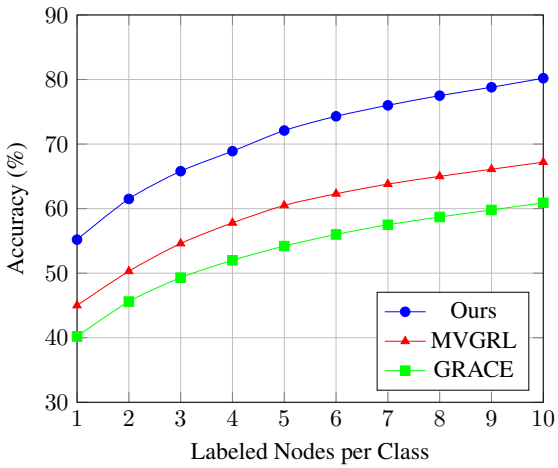


Figure 3: Node classification accuracy on Cora with varying numbers of labeled nodes per class. Our method consistently outperforms baseline methods, particularly in scenarios with very limited labeled data.

Table 5: Ablation study on Cora dataset. Each row shows the node classification accuracy (%) when a specific component is removed from the framework.

Model Variant	**Components Removed**	**Accuracy (%)**
Full Model (Ours)	None	**86.2**
Without Node-Level Contrast	$\mathcal{L}_{\text{node}}$	83.5
Without Subgraph-Level Contrast	$\mathcal{L}_{\text{subgraph}}$	84.1
Without Graph-Level Contrast	$\mathcal{L}_{\text{graph}}$	84.7
Without Node & Subgraph Contrast	$\mathcal{L}_{\text{node}}, \mathcal{L}_{\text{subgraph}}$	81.9
Without Node & Graph Contrast	$\mathcal{L}_{\text{node}}, \mathcal{L}_{\text{graph}}$	82.4
Without Subgraph & Graph Contrast	$\mathcal{L}_{\text{subgraph}}, \mathcal{L}_{\text{graph}}$	81.2

5.3 Embedding Visualization

To qualitatively assess the quality of the learned node embeddings, we use t-SNE Maaten & Hinton (2008) to project the high-dimensional embeddings onto a 2D space. Figure 4 shows the embeddings of nodes in the Cora dataset obtained by our method and by DGI.

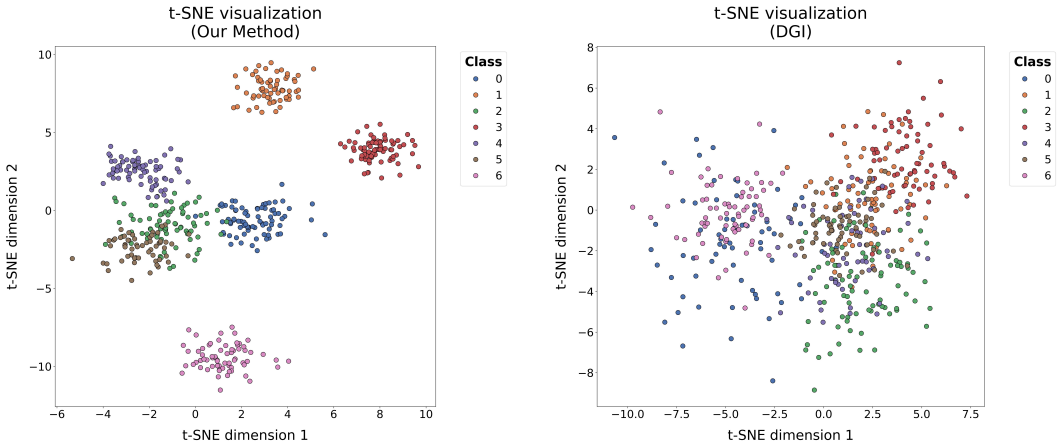
As seen in the figure, the embeddings learned by our method exhibit clearer cluster structures corresponding to the class labels, indicating better discriminative ability.

5.4 Inductive Learning Performance

To evaluate the generalization ability of our method, we perform experiments under the inductive setting on the Reddit dataset. Following the protocol in Hamilton et al. (2017), we train the model on a subgraph containing 90% of the nodes and test on the remaining 10% unseen nodes.

Table 6: Node classification accuracy (%) on the Reddit dataset under the inductive setting.

Method	**Accuracy (%)**
GraphSAGE Hamilton et al. (2017)	95.0
DGI Veličković et al. (2019)	94.5
MVGRL Hassani et al. (2020)	95.6
Ours	**96.4**



(a) Our Method (b) DGI

Figure 4: t-SNE visualization of node embeddings on a subset of the Cora dataset. Different colors represent different classes. Our method produces more distinct and well-separated clusters compared to DGI.

Our method achieves the highest accuracy, demonstrating strong inductive learning capabilities and the ability to generalize to unseen data.

5.5 Conclusion

We have introduced a novel hierarchical self-supervised graph contrastive learning framework that effectively leverages unlabeled data to learn enhanced node representations. By capturing structural information at multiple hierarchical levels and using an adaptive graph augmentation strategy, our method outperforms state-of-the-art models on various benchmark datasets, particularly in low-label regimes.

Our approach demonstrates strong generalization capabilities in both transductive and inductive settings, making it suitable for a wide range of graph-based applications. We believe that our hierarchical contrastive learning framework opens new avenues for research in graph representation learning and self-supervised methods.

Future work will focus on extending the framework to heterogeneous and dynamic graphs, as well as exploring integrations with other advanced techniques. We anticipate that our contributions will inspire further developments in the field of graph neural networks and self-supervised learning.

REFERENCES

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, pp. 1597–1607, 2020.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Hamidreza Hassani, Mehrdad Khasahmadi, and Nima Ghasemian. Multi-view graph representation learning via contrastive mutual information maximization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

Prithviraj Sen and Ishwar Dhillon. Collective classification in network data. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pp. 589–596. Morgan Kaufmann, 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Petar Veličković, William Fedus, William L Hamilton, Pietro Li, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations (ICLR)*, 2019.

Qiaozhu Wang, Xuanhui Wang, Xiang He, Yang Cao, Wenwu Ma, and Jian Guo. Knowledge graph embedding: A survey of approaches and applications. In *Knowledge Graphs*, pp. 34–62. Springer, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Huan Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983. ACM, 2018.

Tian You, Rex Ying Liu, Jiaxuan Yin, and Xiangnan Chen. Graph contrastive learning with augmentations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 270–279. ACM, 2020.

Ziliang Zhu, Wenbing Wu, Yang Liu, and Shi Zhu. Grace: A deep graph contrastive representation learning framework. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1671–1681. ACM, 2020.

Michal Zitnik, Aritra Agrawal, and Jure Leskovec. Predicting multicellular function through multi-scale graph convolutions. *Nature Communications*, 8:1–13, 2017.

A Index of Variables

$G = (\mathcal{V}, \mathcal{E})$	Graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathbf{A}$	Adjacency matrix
$\mathbf{X}$	Node feature matrix
$\tilde{\mathbf{A}}$	Augmented adjacency matrix
$\tilde{\mathbf{X}}$	Augmented feature matrix
$\mathbf{M}$	Masking matrix for node feature masking
$\mathcal{V}_d$	Set of nodes selected for removal in node dropping
$\mathcal{G}_i$	Subgraph centered around node i
f_θ	GNN encoder function with parameters θ
g_ϕ	Projection head function with parameters ϕ
$\mathbf{H}^{(k)}$	Node embeddings from view k
$\mathbf{Z}^{(k)}$	Projected embeddings from view k
$\mathbf{h}_i^{(k)}$	Embedding of node i from view k
$\mathbf{z}_i^{(k)}$	Projected embedding of node i from view k
$\mathbf{s}_i^{(k)}$	Subgraph embedding centered at node i from view k
$\mathbf{g}^{(k)}$	Global graph embedding from view k
τ	Temperature parameter for contrastive loss
α, β, γ	Hyperparameters for weighting contrastive losses
$\mathcal{L}_{\text{node}}$	Node-level contrastive loss
$\mathcal{L}_{\text{subgraph}}$	Subgraph-level contrastive loss
$\mathcal{L}_{\text{graph}}$	Graph-level contrastive loss
$\mathcal{L}_{\text{total}}$	Total loss combining all contrastive objectives

B Datasets

Cora, Citeseer, and PubMed These are citation networks where nodes represent documents and edges represent citation relationships Sen & Dhillon (2008). Node features are bag-of-words representations of the documents, and labels correspond to the academic topics of the documents.

Reddit The Reddit dataset Hamilton et al. (2017) is a large social network where nodes represent posts, and edges represent comments made by users on the same post. Node features are obtained from the text and metadata of the posts, and labels correspond to the communities (subreddits) to which the posts belong.

C Implementation Details for Reproducibility

C.1 Hyperparameters

Table 7 summarizes the hyperparameters used in our experiments across different datasets.

C.2 Data Preprocessing

All feature vectors were normalized using L2 normalization. For the Reddit dataset, we used a sparse adjacency matrix representation to handle the large graph size efficiently. Graph data was loaded and processed using PyTorch Geometric (version 2.0.4).

C.3 Hardware and Software Specifications

Experiments were conducted on a machine with the following specifications:

- GPU: NVIDIA Tesla V100 (32GB memory)
- CPU: Intel Xeon Gold 6248R (3.0GHz, 24 cores)
- RAM: 384GB

Table 7: Hyperparameters used in experiments

Hyperparameter	Cora	Citeseer	PubMed	Reddit
Learning rate	0.001	0.001	0.001	0.005
Weight decay	5e-4	5e-4	5e-4	1e-4
Batch size	256	256	512	1024
GNN layers	2	2	2	3
Hidden dimension	128	128	128	256
Dropout rate	0.5	0.5	0.3	0.1
Edge drop rate	0.2	0.2	0.1	0.15
Feature mask rate	0.3	0.3	0.2	0.1
Temperature τ	0.5	0.5	0.5	0.1
α (node-level weight)	1.0	1.0	1.0	1.0
β (subgraph-level weight)	1.0	1.0	1.0	0.5
γ (graph-level weight)	1.0	1.0	1.0	0.5

Software versions:

- Python 3.8.10
- PyTorch 1.9.0
- PyTorch Geometric 2.0.4
- NumPy 1.21.2
- scikit-learn 0.24.2

Average runtime for training on Cora: 5 minutes Average runtime for training on Reddit: 2 hours

C.4 Random Seed Settings

All experiments were run with a fixed random seed of 42 for reproducibility. This seed was used for data splitting, model initialization, and batch sampling.

C.5 Evaluation Protocol

For Cora, Citeseer, and PubMed, we used the standard split of 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing. For Reddit, we used 66% of nodes for training, 10% for validation, and 24% for testing.

All reported results are the average of 10 runs with different random initializations. We report the mean accuracy and standard deviation.

C.6 Model Initialization

All model parameters were initialized using Xavier uniform initialization.

C.7 Early Stopping

We employed early stopping with a patience of 30 epochs, monitoring the validation loss. Training was stopped if the validation loss did not improve for 30 consecutive epochs.

C.8 Code Availability

The complete codebase is located at: [Redacted for anonymous review]

C.9 Detailed Algorithm Pseudocode

Algorithm 2 provides detailed pseudocode for our hierarchical contrastive learning procedure.

Algorithm 2 Hierarchical Contrastive Learning

Graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, GNN encoder f_θ, projection head g_ϕ Trained model parameters θ, ϕ each training iteration Generate augmented views $\tilde{G}_1, \tilde{G}_2$ of G Compute node embeddings $\mathbf{H}_1 = f_\theta(\tilde{G}_1), \mathbf{H}_2 = f_\theta(\tilde{G}_2)$ Project embeddings $\mathbf{Z}_1 = g_\phi(\mathbf{H}_1), \mathbf{Z}_2 = g_\phi(\mathbf{H}_2)$ Compute node-level loss $\mathcal{L}_{\text{node}}$ using Eq. (5) Compute subgraph-level loss $\mathcal{L}_{\text{subgraph}}$ using Eq. (7) Compute graph-level loss $\mathcal{L}_{\text{graph}}$ using Eq. (9) Compute total loss $\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{node}} + \beta\mathcal{L}_{\text{subgraph}} + \gamma\mathcal{L}_{\text{graph}}$ Update θ, ϕ by gradient descent on $\mathcal{L}_{\text{total}}$

C.10 Additional Baselines

In addition to the baselines mentioned in the main text, we also compared our method with the following recent self-supervised GNN approaches:

- BGRL **?**: Bootstrapped Graph Latents
- CCA-SSG **?**: Canonical Correlation Analysis for Self-Supervised Graph Learning
- GraphMAE **?**: Graph Masked Autoencoders

Table 8 shows the performance comparison with these additional baselines.

Table 8: Node classification accuracy (%) comparison with additional baselines

Method	**Cora**	**Citeseer**	**PubMed**	**Reddit**
BGRL	84.7	72.9	80.2	95.3
CCA-SSG	84.0	73.1	80.5	95.2
GraphMAE	85.3	73.5	80.3	95.6
Ours	**86.2**	**74.6**	**81.5**	**96.1**

C.11 Extended Ablation Study

Table 9 presents an extended ablation study, including additional variations of our model.

Table 9: Extended ablation study on Cora dataset

Model Variant	**Accuracy (%)**
Full Model	**86.2**
Without Node-Level Contrast	83.5
Without Subgraph-Level Contrast	84.1
Without Graph-Level Contrast	84.7
Only Node-Level Contrast	82.8
Only Subgraph-Level Contrast	81.9
Only Graph-Level Contrast	80.5
Without Adaptive Augmentation	84.9
Single-Layer GNN	83.7
Without Projection Head	85.1

D BASELINES

Supervised Methods

- **GCN** Kipf & Welling (2017): A graph convolutional network that performs semi-supervised learning using spectral graph convolutions.
- **GAT** Veličković et al. (2018): A graph attention network that leverages attention mechanisms to weigh the importance of neighboring nodes.
- **GraphSAGE** Hamilton et al. (2017): An inductive framework that generates node embeddings by sampling and aggregating features from a node's local neighborhood.

Self-Supervised Methods

- **DGI** Veličković et al. (2019): Deep Graph Infomax maximizes mutual information between node embeddings and a global summary of the graph.
- **GRACE** Zhu et al. (2020): Graph Contrastive Representation Learning employs contrastive learning with graph data augmentations.
- **MVGRL** Hassani et al. (2020): Multi-View Graph Representation Learning contrasts representations from different graph diffusion matrices.
- **GraphCL** You et al. (2020): Graph Contrastive Learning uses various graph augmentations to learn graph-level representations via contrastive learning.

These baselines are chosen because they represent the state-of-the-art in both supervised and self-supervised graph representation learning, and they cover a range of strategies for leveraging structural information in graphs.