Graph Self-Supervised Learning:
Taxonomy, Frontiers, and Applications

Yixin Liu¹, Yizheng Zheng¹, Ming Jin¹, Feng Xia², Shirui Pan³

¹ Monash University
² RMIT University
³ Griffith University

June 18 1:30pm-3:30pm, 2023 (GMT +10)
# Tutorial outline

<table>
<thead>
<tr>
<th>Content</th>
<th>Presenter</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction and background</strong></td>
<td>Shirui Pan</td>
<td>20 min</td>
</tr>
<tr>
<td>1. Graph analytics and graph neural networks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Background of graph self-supervised learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Taxonomy of graph self-supervised learning</strong></td>
<td>Ming Jin</td>
<td>30 min</td>
</tr>
<tr>
<td>1. Uniform framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Categories of GSSL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Representative methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Frontiers of graph self-supervised learning</strong></td>
<td>Yizhen Zheng</td>
<td>30 min</td>
</tr>
<tr>
<td>1. Graph self-supervised learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Efficient graph self-supervised learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Automatic graph self-supervised learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Applications of graph self-supervised learning</strong></td>
<td>Yixin Liu</td>
<td>30 min</td>
</tr>
<tr>
<td>1. Recommender system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Outlier detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. More applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Future directions and conclusion</strong></td>
<td>Yixin Liu</td>
<td>10 min</td>
</tr>
<tr>
<td>1. Potential directions of graph self-supervised learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Conclusion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part 1: Introduction and background

- Graph analytics
- Graph neural networks
- Graph self-supervised learning: Background
What is graphs?

Example: A Social Network Graph

A Graph has **nodes/vertices** and **edges**.

**Nodes/vertices** → a person in the social network

**Edges** → Connection between people
Graphs in real-world applications

Social Networks

Bibliography Networks

Protein Interaction Networks

Knowledge Graphs

Chemical Compounds

Traffic Networks
Graph Analytics (1): Link Prediction

Friend Recommendation: Does Alice Know Bob in Facebook

Item Recommendation: Which Items will The User Like?
Graph Analytics (2): Community Detection

Typically a Graph Clustering Task
Graph Analytics (3): Node Classification

- d1 is democratic
- d2 is republican
- What can we say about d3 and d4?

-Graph from Jerry Zhu’s Tutorial in ICML 07
Graph Analytics (4): Graph Classification

- Example: Drug Activity Prediction in the Biological domain

It is active to Breast Cancer?
Graph Analytics: Many Others…

- Sampling
- Ranking
- Evolution
- Matching
- Visualization
- Social Influence
- ...

Traditional Machine Learning Pipeline

- **Network Feature Extractions**
  - Nodes: degree/PageRank score
  - Edges: # of common neighbors

- **Feature Vector Construction**
  - Network Feature + Content Feature

- **Machine Learning Tasks**
  - Classification
  - Clustering
  - Link Prediction

Disadvantages:
- Ineffective
- Shallow Method
- Multiple Steps
Graph neural networks (GNNs)

- Methods and Applications
  - Frontier of Deep Learning
  - Effective Representation for Graph Data
  - Wide applications

Graph neural networks (GNNs) are a type of neural network designed to operate on graph-structured data. They are particularly useful for tasks that involve relational data, such as social networks, biological networks, and recommendation systems. GNNs extend the capabilities of traditional machine learning models by incorporating the graph structure into the learning process, allowing them to capture complex relationships between data points.

Applications of GNNs include:
- Recommender Systems
- Community Detection
- Credit Assessment
- Traffic Flow Prediction
- EHR Data Analysis

Real Graphs are transformed through GNNs into Feature Representations, which are then applied to various domains, demonstrating the versatility of GNNs in solving complex problems.
Graph neural networks (GNNs)

A deep encoder which transfer the node in a graph into a latent vector

$$\text{ENC}(v) = \text{multiple layers of non-linear transformations of graph structure}$$

- Learn Better Representation for Graph Data

Big Picture of Graph Neural Networks
Motivation of graph self-supervised learning (GSSL)

Recent graph learning focuses on (semi-) supervised learning scenarios…

Reliance on labels!
Motivation

Recent graph learning focuses on (semi-) supervised learning scenarios…

Reliance on **labels** ➔ Problems:

- Problem 1: Expensive cost of data collection and annotation
Motivation

Recent graph learning focuses on (semi-) supervised learning scenarios…

Reliance on **labels**→ Problems:

- Problem 1: Expensive cost of data collection and annotation
- Problem 2: Pool generalization (over-fitting)
Motivation

Recent graph learning focuses on (semi-) supervised learning scenarios...

Reliance on labels → Problems:

- Problem 1: Expensive cost of data collection and annotation
- Problem 2: Pool generalization (over-fitting)
- Problem 3: Vulnerable to label-related adversarial attacks

Motivation

Recent graph learning focuses on (semi-) supervised learning scenarios…

Reliance on **labels** ➔ Problems:

• Expensive cost of data collection and annotation

• Pool generalization

• Vulnerable to label-related adversarial attacks

**How to address these problems?**

Self-supervised Learning (SSL)

Instead of relying on human-annotated labels, self-supervised learning acquires “labels” from data itself by using an “automatic” process.

Reduces the dependence on manual labels!

Self-supervised Learning (SSL)

"pretext task": use the data itself to generate labels and use supervised methods to solve unsupervised problems.

The representations learned by performing this task can be used as a starting point for our downstream supervised tasks.

Critical problem: how to design the pretext task?
Self-supervised Learning: Computer Vision

Contrastive learning:

Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g., rotation degree, noise level). Note that we only test these operators in ablation, the augmentation policy used to train our models only includes random crop (with flip and resize), color distortion, and Gaussian blur. (Original image cc-by: Von.grzanka)

https://simclr.github.io
Self-supervised Learning: NLP

Large-scale pre-trained language model: BERT

2 self-supervised pre-training schemes of BERT:
• Masked Language Modeling (MLM)
• Next Sentence Prediction (NSP)
Self-supervised Learning on graphs

How to **design pretext tasks** in graph domain?

Can we transfer the pretext tasks designed for CV/NLP to graph domain? - Not trivial!

**Data space**
- CV/NLP: 2D/1D regular-grid Euclidean space
- Graph: Non-Euclidean space

**Reliance between samples**
- CV/NLP: Independent samples (image/text)
- Graph: data examples (nodes) in graph data are correlated by the topological structure

Cannot easily transfer!
Need: exclusive definitions and taxonomies
Self-supervised Learning on graphs

Early studies:

- **Node2vec:**

- **Graph autoencoder (GAE)**
Self-supervised Learning on graphs

A pioneer work of graph SSL:

**Deep Graph Infomax**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, R Devon Hjelm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publication date</td>
<td>2019/5</td>
</tr>
<tr>
<td>Journal</td>
<td>7th International Conference on Learning Representations (ICLR 2019)</td>
</tr>
<tr>
<td>Total citations</td>
<td>Cited by 1535</td>
</tr>
</tbody>
</table>

Growing trend!
Self-supervised Learning on graphs

After DGI...

Multi-view contrastive learning[1]

Subgraph contrastive learning[3]

Following questions:
- Which are the representative works?
- How to categorize them?
- How to formulate them with a unified framework?
- What is the research frontiers?
- Where can GSSL be applied?
- What are the potential future directions?

Part 2: Taxonomy of graph self-supervised learning

- Uniform framework
- Categories of GSSL
- Representative methods
Graph Self-Supervised Learning: A Survey

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, Philip S. Yu, Life Fellow, IEEE

Abstract—Deep learning on graphs has attracted significant interests recently. However, most of the works have focused on (semi-) supervised learning, resulting in shortcomings including heavy label reliance, poor generalization, and weak robustness. To address these issues, self-supervised learning (SSL), which extracts informative knowledge through well-designed pretext tasks without relying on manual labels, has become a promising and trending learning paradigm for graph data. Different from SSL on other domains like computer vision and natural language processing, SSL on graphs has an exclusive background, design ideas, and taxonomies. Under the umbrella of graph self-supervised learning, we present a timely and comprehensive review of the existing approaches which employ SSL techniques for graph data. We construct a unified framework that mathematically formalizes the paradigm of graph SSL. According to the objectives of pretext tasks, we divide these approaches into four categories: generation-based, auxiliary properly-based, contrast-based, and hybrid approaches. We further describe the applications of graph SSL across various research fields and summarize the community used datasets, evaluation benchmarks, performance comparison, and open-source codes of graph SSL. Finally, we discuss the remaining challenges and potential future directions in this research field.

Index Terms—Self-supervised learning, graph analytics, deep learning, graph representation learning, graph neural networks.

Graph self-supervised learning: A survey

Y Liu, M Jin, S Pan, C Zhou, Y Zheng... - ... on Knowledge and ... 2022 - ieeexplore.ieee.org

Deep learning on graphs has attracted significant interests recently. However, most of the works have focused on (semi-) supervised learning, resulting in shortcomings including heavy label reliance, poor generalization, and weak robustness. To address these issues, self-supervised learning (SSL), which extracts informative knowledge through well-designed pretext tasks without relying on manual labels, has become a promising and trending learning paradigm for graph data. Different from SSL on other domains like computer vision...
Overview

• **Unified framework and systematic taxonomy**
  We propose a unified framework that mathematically formalizes graph SSL approaches. Based on our framework, we systematically categorize the existing works into four categories.

• **Comprehensive and up-to-date review**
  We conduct a comprehensive and timely review for classical and latest graph SSL approaches.

• **Abundant resources and applications.**
  We collect abundant resources on graph SSL, including datasets, evaluation benchmark, performance comparison, and open-source codes. We also summarize the practical applications of graph SSL in various research fields.

• **Outlook on future directions**
  We point out the technical limitations of current research. We further suggest six promising directions for future works from different perspectives.
Encoder-Decoder Framework

Graph encoder: GNNs, Transformers, DNN…

Pretext task: For encoder training

Downstream task: To solve real-world problems
Encoder-Decoder Framework

**Graph encoder:**
GNNs, Transformers, DNN...

**Pretext task:**
For encoder training

**Downstream task:**
To solve real-world problems

**Q1:** How to share the encoder between two tasks?

**Q2:** Which types of pretext tasks do we have?

**Q3:** What kind of downstream tasks can be solved?
3 SSL schemes

(i) Pre-training and Fine-tuning (PF)

(ii) Joint Learning (JL)

(iii) Unsupervised Representation Learning (URL)

Q1: How to share the encoder between two tasks?
4 Categories of Graph SSL

(i) Generation-based

(ii) Auxiliary Property-based

(iii) Contrast-based

(iv) Hybrid

Q2: Which types of pretext tasks do we have?
3 Types of Downstream Tasks

Q3: What kind of downstream tasks can be solved?

(i) **Node-level tasks:**
Node classification, node regression…

(ii) **Edge-level tasks:**
Link prediction, edge classification…

(iii) **Graph-level tasks:**
graph classification, graph regression, …
Outline of Graph SSL

Hybrid

Generation-based

Self-supervised Learning

Auxiliary Property-based

Graph

Feature Generation

Structure Generation

Contrast-based

Same-Scale Contrast

Node-Level

Graph-Level

Cross-Scale Contrast

Patch-Global

Context-Global

Clustering-based

Pair Relation-based

Auxiliary Property Classification

Auxiliary Property Regression
Generation-based methods aim to reconstruct the input data and use the input data as the supervision signals.

**Origin:** Autoencoder

- **Feature Generation:** reconstruct the feature information
- **Structure Generation:** reconstruct the topological structure information

**Generation-based Methods**
Feature Generation

- **Pretext Decoder**: Fully connected layers that regresses the features

- **SSL Loss**: Regression loss (MSE)
**Intuition:** Use the neighboring information to reconstruct the masked features (similar to MLM in BERT)
Feature Generation: Representative Method

- Self-Supervised Masked Graph Autoencoder (GraphMAE)

GraphMAE

Structure Generation

- **Pretext Decoder**: Adjacency matrix reconstruction network
- **SSL Loss**: Binary cross-entropy
Structure Generation: Representative Method

- Graph Autoencoder (GAE)

\[ A \xrightarrow{X} G\text{conv} \xrightarrow{\ldots} \xrightarrow{G\text{conv}} \text{Encoder} \xrightarrow{Z} \varphi(\begin{pmatrix} Z & Z^T \end{pmatrix} \cdot ) \xrightarrow{\hat{A}} \]

Dot production decoder

Structure Generation: Representative Method

- Pre-Training GNNs for Generic Structural Feature Extraction

A multi-layer GNN is pre-trained on three structure-guided tasks

Part of GNN layers are fine-tuned on the given downstream tasks

Other Representative Generation-Based Methods

- **GPT-GNN**

## Generation-based Methods: Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pretext Task Category</th>
<th>Downstream Task Level</th>
<th>Training Scheme</th>
<th>Data Type of Graph</th>
<th>Input Data Perturbation</th>
<th>Generation Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Completion [17]</td>
<td>FG</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Feature Masking</td>
<td>Node Feature</td>
</tr>
<tr>
<td>AttributeMask [40]</td>
<td>FG</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Feature Masking</td>
<td>PCA Node Feature</td>
</tr>
<tr>
<td>AttrMasking [16]</td>
<td>FG</td>
<td>Node</td>
<td>PF</td>
<td>Attributed</td>
<td>Feature Masking</td>
<td>Node/Edge Feature</td>
</tr>
<tr>
<td>MGAE [41]</td>
<td>FG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>Feature Noising</td>
<td>Node Feature</td>
</tr>
<tr>
<td>Corrupted Features Reconstruction [42]</td>
<td>FG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>Feature Noising</td>
<td>Node Feature</td>
</tr>
<tr>
<td>GALA [43]</td>
<td>FG</td>
<td>Node/Link</td>
<td>JL</td>
<td>Attributed</td>
<td>-</td>
<td>Node Feature</td>
</tr>
<tr>
<td>Autoencoding [42]</td>
<td>FG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>-</td>
<td>Node Feature</td>
</tr>
<tr>
<td>GAE/VGAE [32]</td>
<td>SG</td>
<td>Link</td>
<td>URL</td>
<td>Attributed</td>
<td>-</td>
<td>Adjacency Matrix</td>
</tr>
<tr>
<td>SIG-VAE [44]</td>
<td>SG</td>
<td>Node/Link</td>
<td>URL</td>
<td>Plain/Attributed</td>
<td>-</td>
<td>Adjacency Matrix</td>
</tr>
<tr>
<td>ARGA/ARVGA [45]</td>
<td>SG</td>
<td>Node/Link</td>
<td>URL</td>
<td>Attributed</td>
<td>-</td>
<td>Adjacency Matrix</td>
</tr>
<tr>
<td>SuperGAT [46]</td>
<td>SG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>-</td>
<td>Partial Edge</td>
</tr>
<tr>
<td>Denoising Link Reconstruction [47]</td>
<td>SG</td>
<td>Node/Link/Graph</td>
<td>PF</td>
<td>Attributed</td>
<td>Edge Masking</td>
<td>Masked Edge</td>
</tr>
<tr>
<td>EdgeMask [40]</td>
<td>SG</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Edge Masking</td>
<td>Masked Edge</td>
</tr>
<tr>
<td>Zhu et al. [48]</td>
<td>SG</td>
<td>Node</td>
<td>PF</td>
<td>Attributed</td>
<td>Feature Masking/Edge Masking</td>
<td>Partial Edge</td>
</tr>
</tbody>
</table>
Auxiliary Property-based Methods: Origin

**Generation-based methods** aim to predict node-, link- and graph- level properties which can be obtained from the graph data freely.

**Origin:** Supervised learning ⇒ Learn with “sample-label” pairs

**Difference:**
⇒ Supervised learning uses **manual labels** to train models
⇒ Auxiliary property-based methods uses **pseudo labels** to train models

Taxonomy: follows supervised learning
Auxiliary Property Classification

I. Clustering-based

II. Pair Relation-based

How to acquire properties?
- Clustering
- Pair Relation

• **Pretext Decoder**: Classifier head

• **SSL Loss**: Classification Loss (Cross-entropy)
Clustering-based Auxiliary Property Classification: Representative Methods

- **Node Feature Clustering**
  - Feature-based clustering (e.g., k-means)

- **Graph Topology Partitioning**
  - Structure-based clustering (e.g., Metis)

Pair Relation-based Auxiliary Property Classification: Representative Method

Objective: predict hop counts

1-hop context
\[ h(<z_i, z_j>, y=0) \]

2-hop context
\[ h(<z_i, z_j>, y=1) \]

3-hop context
\[ h(<z_i, z_j>, y=2) \]

\[ \cdots \]

k-hop context
\[ h(<z_i, z_j>, y=k-1) \]

Auxiliary Property Regression: Representative Method

- **NodeProperty**

  - **Pretext Decoder**: Regression head
  - **SSL Loss**: Regression Loss (MSE)

  E.g., target property ⇒ the degree of nodes

## Auxiliary Property-based Methods: Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pretext Task Category</th>
<th>Downstream Task Level</th>
<th>Training Scheme</th>
<th>Data Type of Graph</th>
<th>Property Level</th>
<th>Mapping Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Clustering [17]</td>
<td>CAPC</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node</td>
<td>Feature-based Clustering</td>
</tr>
<tr>
<td>M3S [54]</td>
<td>CAPC</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>Node</td>
<td>Feature-based Clustering</td>
</tr>
<tr>
<td>Graph Partitioning [17]</td>
<td>CAPC</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node</td>
<td>Structure-based Clustering</td>
</tr>
<tr>
<td>Cluster Preserving [48]</td>
<td>CAPC</td>
<td>Node/Link/Graph</td>
<td>PF</td>
<td>Attributed</td>
<td>Node</td>
<td>Structure-based Clustering</td>
</tr>
<tr>
<td>CAGNN [55]</td>
<td>CAPC</td>
<td>Node</td>
<td>URL</td>
<td>Attributed</td>
<td>Node</td>
<td>Feature-based Clustering with Structural Refinement</td>
</tr>
<tr>
<td>S^2GRL [56]</td>
<td>PAPC</td>
<td>Node/Link</td>
<td>URL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Shortest Distance Function</td>
</tr>
<tr>
<td>PairwiseDistance [41]</td>
<td>PAPC</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Shortest Distance Function</td>
</tr>
<tr>
<td>Centrality Score Ranking [48]</td>
<td>PAPC</td>
<td>Node/Link/Graph</td>
<td>PF</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Centrality Scores Comparison</td>
</tr>
<tr>
<td>TopoTER [57]</td>
<td>PAPC</td>
<td>Node/Graph</td>
<td>URL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Topological Transformation Indicator</td>
</tr>
<tr>
<td>NodeProperty [41]</td>
<td>APR</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node</td>
<td>Degree Calculation</td>
</tr>
<tr>
<td>Distance2Cluster [41]</td>
<td>APR</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Distance to Cluster Center</td>
</tr>
<tr>
<td>PairwiseAttrSim [41]</td>
<td>APR</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Cosine Similarity of Feature</td>
</tr>
<tr>
<td>SimP-GCN [58]</td>
<td>APR</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>Node Pair</td>
<td>Cosine Similarity of Feature</td>
</tr>
</tbody>
</table>
Contrast-based Methods: Origin

Contrast-based methods learn by maximizing the agreement between two augmented instances.

**Origin:** Visual Contrastive Learning ⇒ Mutual Information (MI) Maximization

**Key components:**
- Data augmentation
- Contrastive model <main taxonomy>
- Contrastive objective
Data Augmentation on Graphs

- **Attributive augmentations**
  - Node feature masking (NFM)
  - Node feature shuffle (NFS)

- **Topological augmentations**
  - Edge modification (EM)
  - Graph diffusion (GD)

- **Hybrid augmentations**
  - Subgraph sampling (SS)
Graph Contrastive Learning: Taxonomy

- **Same-Scale Contrast**
  - Node-Level
  - Graph-Level

- **Contrast-based**

- **Cross-Scale Contrast**
  - Patch-Global
  - Context-Global
Node-Level Same-Scale Contrast: Representative Method

- **GRACE**

\[ \mathcal{G} = (X, A) \]

- **SimCLR Contrastive Learning Framework**
  - Intra + Inter view contrast
  - Augmentation: Remove edges (EM) + mask features (NFM)

Two graph views are first generated via graph augmentations.

Then, online and target networks are employed to generate node representations for each view.

A multi-scale graph contrastive schema with the self-knowledge distillation is proposed to train the online graph encoder.
Graph-Level Same-Scale Contrast: Representative Method

- **GraphCL**

  - **SimCLR Contrastive Learning Framework**
  - Augmentation: EM+SS

Patch-Global Cross-Scale Contrast: Representative Method

- **DGI**

- Maximize the MI between node and full graph

Patch-Global Cross-Scale Contrast: Representative Method

- **G-Zoom**

![Graph augmentation and encoding diagram](a) Graph Augmentation and Encoding

![Contrastive learning with adjusted zooming diagram](b) Contrastive Learning with Adjusted Zooming

- **Node vs. Node**
- **Node vs. Context**
- **Node vs. Graph**

Context-Global Cross-Scale Contrast: Representative Method

- **MICRO-Graph**

  ![Diagram showing MICRO-Graph process]

  - Motif vs. Full graph

MI Estimation - Contrastive Loss

- **Jensen-Shannon Estimator**
  \[ \mathcal{MI}_{JSD}(h_i, h_j) = \mathbb{E}_P \left[ \log \left( \mathcal{D}(h_i, h_j) \right) \right] \\
  - \mathbb{E}_{P \times \tilde{P}} \left[ \log \left( 1 - \mathcal{D}(h_i, h'_j) \right) \right]. \]

- **Noise-Contrastive Estimator**
  \[ \mathcal{MI}_{NCE}(h_i, h_j) = \mathbb{E}_{P \times \tilde{P}_N} \left[ \log \frac{e^{\mathcal{D}(h_i, h_j)}}{e^{\mathcal{D}(h_i, h_j)} + \sum_{n \in N} e^{\mathcal{D}(h_i, h'_n)}} \right]. \]

- **Triplet loss**
  \[ \mathcal{L}_{triplet} = \mathbb{E}_{P \times \tilde{P}} \left[ \max \left( \mathcal{D}(h_i, h_j) - \mathcal{D}(h_i, h'_j) + \epsilon, 0 \right) \right]. \]

- **BYOL loss**
  \[ \mathcal{L}_{byol} = \mathbb{E}_{P^N} \left[ -\frac{2}{N} \sum_{i, j \in N} \frac{[p_\psi(h_i)]^T h_j}{\|p_\psi(h_i)\| \|h_j\|} \right]. \]

- **Barlow Twins loss**
  \[ \mathcal{L}_{bt} = \mathbb{E}_{B \sim P^N} \left[ \sum_a (1 - \frac{\sum_{i \in B} H_{ia}^{(1)} H_{ia}^{(2)}}{\|H_{ia}^{(1)}\| \|H_{ia}^{(2)}\|})^2 \right. \]
  \[ + \lambda \sum_a \sum_{b \neq a} \left( \frac{\sum_{i \in B} H_{ia}^{(1)} H_{ib}^{(2)}}{\|H_{ia}^{(1)}\| \|H_{ib}^{(2)}\|} \right)^2 \right]. \]
## Contrast-based Methods: Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pretext Task Category</th>
<th>Downstream Task Level</th>
<th>Training Scheme</th>
<th>Data Type of Graph</th>
<th>Graph Augmentation</th>
<th>Objective Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>node2vec [31]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Plain</td>
<td>SS</td>
<td>SkipGram</td>
</tr>
<tr>
<td>GraphSAGE [79]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>SS</td>
<td>JSD</td>
</tr>
<tr>
<td>SILAR [79]</td>
<td>NSC</td>
<td>Node</td>
<td>JL</td>
<td>Heterogeneous</td>
<td>Meta-path sampling</td>
<td>JSD</td>
</tr>
<tr>
<td>LINE [79]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Plain</td>
<td>SS</td>
<td>JSD</td>
</tr>
<tr>
<td>GRACE [33]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>NFM+EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>GROK [18]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>NFM+Adversarial EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>GCA [67]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>Adaptive NFM+Adaptive EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>GraphCL(N) [81]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>SS+NFS+EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>GCC [13]</td>
<td>NSC</td>
<td>Node/Graph</td>
<td>ULR</td>
<td>Plain</td>
<td>SS</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>HeCo [82]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Heterogeneous</td>
<td>NFM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>Contrast-Reg [71]</td>
<td>NSC</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>Arbitrary</td>
<td>JSD</td>
</tr>
<tr>
<td>BORL [63]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>NFM+EM</td>
<td>BYOL</td>
</tr>
<tr>
<td>SelfGNN [84]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>GD+Node attribution transformation</td>
<td>BYOL</td>
</tr>
<tr>
<td>G-BT [86]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>NFM+EM</td>
<td>Barlow Twins</td>
</tr>
<tr>
<td>MERIT [66]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>SS+GD-NFM+EM</td>
<td>BYOL+InfoNCE</td>
</tr>
<tr>
<td>DGraphCL [65]</td>
<td>NSC</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>Adaptive NFM+Adaptive EM</td>
<td>KL-Divergence</td>
</tr>
<tr>
<td>GraphCL(G) [65]</td>
<td>GSC</td>
<td>Graph</td>
<td>PF/URL</td>
<td>Attributed</td>
<td>Ss+NFM+EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>DAFL [88]</td>
<td>GSC</td>
<td>Graph</td>
<td>ULR</td>
<td>Attributed</td>
<td>Noise Mixing</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>AD-GCL [73]</td>
<td>GSC</td>
<td>Graph</td>
<td>PF/URL</td>
<td>Attributed</td>
<td>Adversarial EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>JOAC [69]</td>
<td>GSC</td>
<td>Graph</td>
<td>PF/URL</td>
<td>Attributed</td>
<td>Automated</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>CSSL [74]</td>
<td>GSC</td>
<td>Graph</td>
<td>PF/URL</td>
<td>Attributed</td>
<td>Ss+Node insertion/deletion+EM</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>LCGAN [89]</td>
<td>GSC</td>
<td>Graph</td>
<td>JL</td>
<td>Attributed</td>
<td>Arbitrary</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>RCSD [74]</td>
<td>GSC</td>
<td>Graph</td>
<td>JL/URL</td>
<td>Attributed</td>
<td>GD+EM</td>
<td>BYOL+InfoNCE</td>
</tr>
<tr>
<td>DCI [13]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>GIC [90]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>Arbitrary</td>
<td>JSD</td>
</tr>
<tr>
<td>HDG [91]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Heterogeneous</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>ConCH [92]</td>
<td>PCCG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>DMGI [93]</td>
<td>PCCG</td>
<td>Node</td>
<td>JL/URL</td>
<td>Heterogeneous</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>EGI [94]</td>
<td>PCCG</td>
<td>Node</td>
<td>PF/JL</td>
<td>Attributed</td>
<td>SS</td>
<td>JSD</td>
</tr>
<tr>
<td>STDG [70]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Dynamic</td>
<td>Node feature shuffling</td>
<td>JSD</td>
</tr>
<tr>
<td>KSRL [95]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>None</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>MVRUL [14]</td>
<td>PCCG</td>
<td>Node/Graph</td>
<td>ULR</td>
<td>Attributed</td>
<td>GD+SS</td>
<td>JSD</td>
</tr>
<tr>
<td>SUBG-CN [77]</td>
<td>PCCG</td>
<td>Node</td>
<td>ULR</td>
<td>Attributed</td>
<td>SS+Node representation shuffling</td>
<td>Triplet</td>
</tr>
<tr>
<td>SLICE [96]</td>
<td>PCCG</td>
<td>Edge</td>
<td>JL</td>
<td>Heterogeneous</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>InfoGraph [97]</td>
<td>PCCG</td>
<td>Graph</td>
<td>JL/URL</td>
<td>Attributed</td>
<td>None</td>
<td>JSD</td>
</tr>
<tr>
<td>Robinson et al. [88]</td>
<td>PCCG</td>
<td>Graph</td>
<td>ULR</td>
<td>Attributed</td>
<td>Arbitrary</td>
<td>JSD</td>
</tr>
<tr>
<td>BIGI [99]</td>
<td>GCGC</td>
<td>Graph</td>
<td>ULR</td>
<td>Heterogeneous</td>
<td>SS</td>
<td>JSD</td>
</tr>
<tr>
<td>HTC [102]</td>
<td>CGGC</td>
<td>Graph</td>
<td>JL</td>
<td>Attributed</td>
<td>NPS</td>
<td>JSD</td>
</tr>
<tr>
<td>MICRO-Graph [103]</td>
<td>CGGC</td>
<td>Graph</td>
<td>ULR</td>
<td>Attributed</td>
<td>SS</td>
<td>InfoNCE</td>
</tr>
<tr>
<td>SUGAR [102]</td>
<td>CGGC</td>
<td>Graph</td>
<td>JL</td>
<td>Attributed</td>
<td>SS</td>
<td>JSD</td>
</tr>
</tbody>
</table>
Hybrid Methods: Motivation

**Hybrid methods** integrate various pretext tasks together in a multi-task learning fashion.

**Motivation:**
- A single pretext task cannot provide sufficient guidance.
- Using multiple pretext tasks can better leverage the advantages of various types of supervision signals.
Hybrid Methods: Representative Methods

- **GMI**

  - Edge MI: Structure generation
  - Node MI: Same-scale contrast

Hybrid Methods: Representative Methods

- **GROVER**

- Node- and edge-level reconstruction

- Context- and graph-level auxiliary properties prediction

- Backbone model: Node and edge GNN transformers

# Hybrid Methods: Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pretext Task Categoriess</th>
<th>Downstream Task Level</th>
<th>Training Scheme</th>
<th>Data Type of Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-Bert [104]</td>
<td>FG/SG</td>
<td>Node</td>
<td>PF</td>
<td>Attributed</td>
</tr>
<tr>
<td>PT-DGNN [105]</td>
<td>FG/SG</td>
<td>Link</td>
<td>PF</td>
<td>Dynamic</td>
</tr>
<tr>
<td>M. et al. [43]</td>
<td>FG/FG/FG</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
</tr>
<tr>
<td>GMI [106]</td>
<td>SG/NSC</td>
<td>Node/Link</td>
<td>URL</td>
<td>Attributed</td>
</tr>
<tr>
<td>CG³ [107]</td>
<td>SG/NSC</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
</tr>
<tr>
<td>MVMI-FT [108]</td>
<td>SG/PGCC</td>
<td>Node</td>
<td>URL</td>
<td>Attributed</td>
</tr>
<tr>
<td>GraphLoG [109]</td>
<td>NSC/GSC/CGCC</td>
<td>Graph</td>
<td>PF</td>
<td>Attributed</td>
</tr>
<tr>
<td>HDMI [110]</td>
<td>NSC/PGCC</td>
<td>Node</td>
<td>URL</td>
<td>Multiplex</td>
</tr>
<tr>
<td>LnL-GNN [111]</td>
<td>NSC/NSC</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
</tr>
<tr>
<td>Hu et al. [48]</td>
<td>SG/APC/APC</td>
<td>Node/Link/Graph</td>
<td>PF</td>
<td>Attributed</td>
</tr>
<tr>
<td>GROVER [10]</td>
<td>APC/APC</td>
<td>Node/Link/Graph</td>
<td>PF</td>
<td>Attributed</td>
</tr>
<tr>
<td>Kou et al. [112]</td>
<td>FG/SG/APC</td>
<td>Node</td>
<td>JL</td>
<td>Attributed</td>
</tr>
</tbody>
</table>
Part 3: Frontiers of graph self-supervised learning

- Efficient graph self-supervised learning: A new paradigm
- Heterophilic graph self-supervised learning
- Heterogeneous graph self-supervised learning
Efficient graph self-supervised learning: A new paradigm
Existing Problems - Slow Computation with Node Comparison

These contrastive-learning approaches rely on node-to-node comparison.

Existing Problems - Slow Computation with Node Comparison

Node-to-node comparison require heavy gradient computation. For example, for the two representative contrastive losses:

**InfoNCE Loss**
\[
\mathcal{L}_{\text{NCE}}(i) = -\log \frac{e^{z_i \cdot c_i / \tau}}{\sum_{k=1}^{N} e^{z_i \cdot z_k / \tau}},
\]

Gradient Computation require all negative samples

**JSD-estimator**
\[
\mathcal{L}_{\text{JSD}}(i) = -\log \mathcal{D}(z_i, \bar{s}) + \log (1 - \mathcal{D}(\tilde{z}_i, \bar{s})) , \quad \bar{s} = \sigma \left( \frac{1}{N} \sum_{i=1}^{N} z_i \right)
\]

Gradient Computation require all positive samples

Introduction to Group Discrimination (GD)

Positive Group: Summarised Node representations (\( \mathcal{R}^{1 \times D} \)) generated with original or augmented graph.

Summarisation (e.g., sum):

\[
\begin{align*}
\mathcal{R}^{1 \times D} & \quad \rightarrow \quad \mathcal{R}^{1 \times 1} \\
\text{Positive Group:} & \quad \text{Summarised Node representation (generated with original or augmented graph)}
\end{align*}
\]

Negative Group: Summarised Node representation (\( \mathcal{R}^{1 \times 1} \)) generated with corrupted graph.

Introduction to Group Discrimination (GD)

Use a very simple BCE loss to conduct discrimination

$$\mathcal{L}_{BCE} = -\frac{1}{2N} \left( \sum_{i=1}^{2N} y_i \log h_i + (1 - y_i) \log(1 - h_i) \right)$$

A very simple binary classification task: discriminating positive/negative samples

If positive $\rightarrow y = 1$, else $\rightarrow y = 0$

$b$ Group Discrimination

$$h_i \in \mathcal{R}^{1 \times 1}$$ is the summarised node embedding/binary prediction for a node $i$

Rethinking DGI

Original Thought of DGI → MI maximization between nodes and summary vector.

\[
\mathcal{L}_{\text{DGI}} = \frac{1}{2N} \left( \sum_{i=1}^{N} \log \mathcal{D}(z_i, \bar{s}) + \log (1 - \mathcal{D}(\bar{z}_i, \bar{s})) \right),
\]

However, due to inappropriate usage of Sigmoid function....

$$\mathcal{L}_{DGI} = \frac{1}{2N} \left( \sum_{i=1}^{N} \log \mathcal{D}(z_i, \overrightarrow{s}) + \log(1 - \mathcal{D}(\tilde{z}_i, \overrightarrow{s})) \right),$$

$$\overrightarrow{s} = \sigma \left( \frac{1}{N} \sum_{i=1}^{N} z_i \right)$$

Sigmoid Function

Node Embeddings
Rethinking DGI

Value in summary vector $\bar{s}$ almost becomes constant vector $s = \epsilon I = I$ with no variance.

The assumption of learning via MI interaction between nodes and summary vector

<table>
<thead>
<tr>
<th>Dataset</th>
<th>0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>70.3±0.7</td>
<td>82.4±0.2</td>
<td>82.3±0.3</td>
<td>82.5±0.4</td>
<td>82.3±0.3</td>
<td>82.5±0.1</td>
</tr>
<tr>
<td>CiteSeer</td>
<td>61.8±0.8</td>
<td>71.7±0.6</td>
<td>71.9±0.7</td>
<td>71.6±0.9</td>
<td>71.7±1.0</td>
<td>71.6±0.8</td>
</tr>
<tr>
<td>PubMed</td>
<td>68.3±1.5</td>
<td>77.8±0.5</td>
<td>77.9±0.8</td>
<td>77.7±0.9</td>
<td>77.4±1.1</td>
<td>77.2±0.9</td>
</tr>
</tbody>
</table>

Changing $\epsilon$ has trivial effect on model performance.

Rethinking DGI

Simplifying DGI

Set $\epsilon$ to 1 for $s = \epsilon I = I$, and remove $w$ in $\mathcal{D}(z_i, \bar{s}) = z_i \cdot w \cdot \bar{s}$.

$$\mathcal{L}_{\text{DGI}} = \frac{1}{2N} \left( \sum_{i=1}^{N} \log \mathcal{D}(z_i, \bar{s}) + \log(1 - \mathcal{D}(\tilde{z}_i, \bar{s})) \right),$$

$$= \frac{1}{2N} \left( \sum_{i=1}^{N} \log(z_i \cdot \bar{s}) + \log(1 - \tilde{z}_i \cdot \bar{s}) \right),$$

$$= \frac{1}{2N} \left( \sum_{i=1}^{N} \log(\text{sum}(z_i)) + \log(1 - \text{sum}(\tilde{z}_i)) \right),$$
Rethinking DGI

\[ \text{Considering summarised embedding as } h_i \rightarrow \text{become BCE loss} \]

\[ \mathcal{L}_{\text{BCE}} = \frac{1}{2N} \left( \sum_{i=1}^{2N} y_i \log h_i + (1 - y_i) \log(1 - h_i) \right), \]
Rethinking DGI

With the new loss ➔ **Dramatic improvement in memory and time**

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>DGI</td>
<td>81.7±0.6</td>
<td>71.5±0.7</td>
<td>77.3±0.6</td>
</tr>
<tr>
<td></td>
<td>$\text{DGI}_{\text{BCE}}$</td>
<td>82.5±0.3</td>
<td>71.7±0.6</td>
<td>77.7±0.5</td>
</tr>
<tr>
<td>Memory</td>
<td>DGI</td>
<td>4189MB</td>
<td>8199MB</td>
<td>11471MB</td>
</tr>
<tr>
<td></td>
<td>$\text{DGI}_{\text{BCE}}$</td>
<td>1475MB</td>
<td>64.8%</td>
<td>1587MB</td>
</tr>
<tr>
<td>Time</td>
<td>DGI</td>
<td>0.085s</td>
<td>0.134s</td>
<td>0.158s</td>
</tr>
<tr>
<td></td>
<td>$\text{DGI}_{\text{BCE}}$</td>
<td>0.010s</td>
<td>8.5×</td>
<td>0.021s</td>
</tr>
</tbody>
</table>

Rethinking DGI

\[
= \frac{1}{2N} \left( \sum_{i=1}^{N} \log(\text{sum}(z_i)) + \log(1 - \text{sum}(\tilde{z}_i)) \right),
\]

Replacing the summation with other aggregation function

Table 11: The experiment result on three datasets with different aggregation function on node embeddings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>82.5 ±0.2</td>
<td>71.7 ±0.6</td>
<td>77.7 ±0.5</td>
</tr>
<tr>
<td>Mean</td>
<td>81.8 ±0.5</td>
<td>71.8 ±1.1</td>
<td>76.5 ±1.2</td>
</tr>
<tr>
<td>Min</td>
<td>80.4 ±1.3</td>
<td>61.7 ±1.8</td>
<td>70.1 ±1.9</td>
</tr>
<tr>
<td>Max</td>
<td>71.4 ±1.2</td>
<td>65.3 ±1.4</td>
<td>70.2 ±2.8</td>
</tr>
<tr>
<td>linear</td>
<td>82.2 ±0.4</td>
<td>72.1 ±0.7</td>
<td>77.9 ±0.5</td>
</tr>
</tbody>
</table>
Rethinking DGI

\[ \mathcal{L}_{BCE} = \frac{1}{2N} \left( \sum_{i=1}^{2N} y_i \log h_i + (1 - y_i) \log(1 - h_i) \right), \]

With this loss, we can see instead of contrastive learning, DGI is a Group Discrimination method.
Proposed Framework: Graph Group Discrimination (GGD)

## Experiment (Small-to-Medium scale Dataset)

### Overall Performance Comparison

<table>
<thead>
<tr>
<th>Data</th>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
<th>Comp</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, A, Y</td>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
<td>76.3±0.5</td>
<td>87.3±1.0</td>
</tr>
<tr>
<td>X, A, Y</td>
<td>GAT</td>
<td>83.0±0.7</td>
<td>72.5±0.7</td>
<td>79.0±0.3</td>
<td>79.3±1.1</td>
<td>86.2±1.5</td>
</tr>
<tr>
<td>X, A, Y</td>
<td>SGC</td>
<td>81.0±0.0</td>
<td>71.9±0.1</td>
<td>78.9±0.0</td>
<td>74.4±0.1</td>
<td>86.4±0.0</td>
</tr>
<tr>
<td>X, A, Y</td>
<td>CG3</td>
<td>83.4±0.7</td>
<td>73.6±0.8</td>
<td>80.2±0.8</td>
<td>79.9±0.6</td>
<td>89.4±0.5</td>
</tr>
<tr>
<td>X, A</td>
<td>DGI</td>
<td>81.7±0.6</td>
<td>71.5±0.7</td>
<td>77.3±0.6</td>
<td>75.9±0.6</td>
<td>83.1±0.5</td>
</tr>
<tr>
<td>X, A</td>
<td>GMI</td>
<td>82.7±0.2</td>
<td>73.0±0.3</td>
<td>80.1±0.2</td>
<td>76.8±0.1</td>
<td>85.1±0.1</td>
</tr>
<tr>
<td>X, A</td>
<td>MVGRL</td>
<td>82.9±0.7</td>
<td>72.6±0.7</td>
<td>79.4±0.3</td>
<td>79.0±0.6</td>
<td>87.3±0.3</td>
</tr>
<tr>
<td>X, A</td>
<td>GRACE</td>
<td>80.0±0.4</td>
<td>71.7±0.6</td>
<td>79.5±1.1</td>
<td>71.8±0.4</td>
<td>81.8±1.0</td>
</tr>
<tr>
<td>X, A</td>
<td>BGRL</td>
<td>80.5±1.0</td>
<td>71.0±1.2</td>
<td>79.5±0.6</td>
<td>89.2±0.9</td>
<td>91.2±0.8</td>
</tr>
<tr>
<td>X, A</td>
<td>GBT</td>
<td>81.0±0.5</td>
<td>70.8±0.2</td>
<td>79.0±0.1</td>
<td>88.5±1.0</td>
<td>91.1±0.7</td>
</tr>
<tr>
<td>X, A</td>
<td>GGD</td>
<td>84.1±0.4</td>
<td>73.0±0.6</td>
<td>81.3±0.8</td>
<td>90.1±0.9</td>
<td>92.5±0.6</td>
</tr>
</tbody>
</table>

### Time Consumption Improvement (epoch per second)

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
<th>Comp</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGI</td>
<td>0.085</td>
<td>0.134</td>
<td>0.158</td>
<td>0.171</td>
<td>0.059</td>
</tr>
<tr>
<td>GMI</td>
<td>0.394</td>
<td>0.497</td>
<td>2.285</td>
<td>1.297</td>
<td>0.637</td>
</tr>
<tr>
<td>MVGRL</td>
<td>0.123</td>
<td>0.171</td>
<td>0.488</td>
<td>0.663</td>
<td>0.468</td>
</tr>
<tr>
<td>GRACE</td>
<td>0.056</td>
<td>0.092</td>
<td>0.893</td>
<td>0.546</td>
<td>0.203</td>
</tr>
<tr>
<td>BGRL</td>
<td>0.085</td>
<td>0.094</td>
<td>0.147</td>
<td>0.337</td>
<td>0.273</td>
</tr>
<tr>
<td>GBT</td>
<td>0.073</td>
<td>0.072</td>
<td>0.103</td>
<td>0.492</td>
<td>0.173</td>
</tr>
<tr>
<td>GGD</td>
<td>0.010</td>
<td>0.021</td>
<td>0.015</td>
<td>0.016</td>
<td>0.009</td>
</tr>
</tbody>
</table>

**Improve** 7.3-39.4x 3.4-23.7x 6.9-152.3x 10.7-15.3x 19.2-70.8x

### Memory Consumption Improvement (MB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
<th>Comp</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGI</td>
<td>4,189</td>
<td>8,199</td>
<td>11,471</td>
<td>7,991</td>
<td>4,946</td>
</tr>
<tr>
<td>GMI</td>
<td>4,527</td>
<td>5,467</td>
<td>14,697</td>
<td>10,655</td>
<td>5,219</td>
</tr>
<tr>
<td>MVGRL</td>
<td>5,381</td>
<td>5,429</td>
<td>6,619</td>
<td>6,645</td>
<td>6,645</td>
</tr>
<tr>
<td>GRACE</td>
<td>1,913</td>
<td>2,043</td>
<td>12,597</td>
<td>8,129</td>
<td>4,881</td>
</tr>
<tr>
<td>BGRL</td>
<td>1,627</td>
<td>1,749</td>
<td>2,299</td>
<td>5,069</td>
<td>3,303</td>
</tr>
<tr>
<td>GBT</td>
<td>1,651</td>
<td>1,799</td>
<td>2,461</td>
<td>5,037</td>
<td>2,641</td>
</tr>
<tr>
<td>GGD</td>
<td>1,475</td>
<td>1,587</td>
<td>1,629</td>
<td>1,787</td>
<td>1,637</td>
</tr>
</tbody>
</table>

**Improve** 10.7-72.6% 11.8-80.6% 27.2-85.8% 64.5-83.2% 38.0-75.4%
Experiment (Large scale Dataset - Ogbn-arxiv)

Using only **0.18** seconds and **69.8%** less memory to reach SOTA.

10783 faster than existing methods.

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Valid</th>
<th>Test</th>
<th>Memory</th>
<th>Time</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised GCN</td>
<td>73.0±0.2</td>
<td>71.7±0.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MLP</td>
<td>57.7±0.4</td>
<td>55.5±0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Node2vec</td>
<td>71.3±0.1</td>
<td>70.1±0.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DGI</td>
<td>71.3±0.1</td>
<td>70.3±0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRACE(10k epos)</td>
<td>72.6±0.2</td>
<td>71.5±0.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BGRL(10k epos)</td>
<td>72.5±0.1</td>
<td>71.6±0.1</td>
<td>OOM (Full-graph)</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>GBT(300 epos)</td>
<td>71.0±0.1</td>
<td>70.1±0.2</td>
<td>14,959MB</td>
<td>6.47</td>
<td>1,941.00</td>
</tr>
<tr>
<td>GGD(1 epo)</td>
<td>72.7±0.3</td>
<td>71.6±0.5</td>
<td>4,513MB</td>
<td>69.8%</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Fast convergence → converge with only 1 epoch

---

Heterophilic Graph Self-supervised Learning
Homophily assumption

Most UGRL methods are designed based on the homophily assumption:

Linked nodes tend to share similar attributes with each other.

- Low-pass filter-like GNNs\textsuperscript{[1]} (e.g., GCN) as encoders:

Limitation

Do real-world graphs always obey the homophily assumption?
No!

- Pure homophilic graph is ideal, real-world graphs often contain heterophilic edges.
- Real-world homophilic graphs can also include heterophilic edges.
- In heterophilic graphs, heterophilic edges are much more than homophilic edges.
- Adversarial attack tends to reduce the homophily of graphs [5].

The behind homophily assumption hinders the generalization ability to heterophilic graphs and robustness against adversarial attack of most UGRL methods.

Observation

Most UGRL methods are designed based on the homophily assumption:

Linked nodes tend to share similar attributes with each other.

Visualization of the cosine similarity of:

- Raw Feat.
- Rep. by GAE
- Rep. by DGI
- Rep. by GRACE

CNP: connected node pairs
RNP: randomly sampled node pairs

(a) Cosine Sim.  (b) Cosine Sim.  (c) Cosine Sim.  (d) Cosine Sim.

Cora dataset

(a) Cosine Sim.  (b) Cosine Sim.

Texas dataset

All the connected nodes are pushed to be closer in the representation space, even if some of them have moderate feature similarities that are comparable to randomly sampled node pairs.
Contribution

To address the aforementioned limitation...

(Q1) Is it possible to distinguish between two types of edges in an unsupervised manner?

(A1) trainable edge discriminator with a pivot-anchored ranking loss function.

(Q2) How to effectively couple edge discriminating with representation learning into an integrated UGRL model?

(A2) dual-channel graph encoding module with robust cross-channel contrasting. Training with a closed-loop interplay.

Proposed method - GREET

To discriminate the homophilic and heterophilic edges without accessing node labels.

Edge discriminating

- Edge discriminator – a two-layer MLP:

\[
\begin{align*}
    h'_i &= \text{MLP}_1([x_i||s_i]), \\
    h'_j &= \text{MLP}_1([x_j||s_j]), \\
    \theta_{i,j} &= \frac{\text{MLP}_2([h'_i||h'_j]) + \text{MLP}_2([h'_j||h'_i])}{2},
\end{align*}
\]

- Input of edge discriminator:
  Raw feature + Structural encoding (SE)

Random walk diffusion process-based SE\[6\]:

\[
s_i = \left[ T_{ii}, T_{i1}^2, \ldots, T_{i1}^d \right] \in \mathbb{R}^{d_s}, \text{ where } T = AD^{-1}
\]
View generalization

- Gumbel-Max reparametrization trick [7]:

\[ \hat{w}_{i,j} = \text{Sigmoid} \left( (\theta_{i,j} + \log \delta - \log(1 - \delta)) / \tau_g \right) \]

- View generation:

\[
\begin{align*}
\mathcal{G} &= (\mathbf{A}, \mathbf{X}) \\
\mathcal{G}^{(hm)} &= (\mathbf{A}^{(hm)}, \mathbf{X}) \\
\mathcal{G}^{(ht)} &= (\mathbf{A}^{(ht)}, \mathbf{X})
\end{align*}
\]

where

\[ A_{i,j}^{(hm)} = \hat{w}_{i,j}, \quad A_{i,j}^{(ht)} = 1 - \hat{w}_{i,j}, \quad \text{for } e_{i,j} \in \mathcal{E} \]

Dual-channel encoding

- **Homo. View encoder – low-pass filter:**
  \[ H_0^{(hm)} = \text{MLP}^{(hm)}(X), \quad H_l^{(hm)} = \tilde{A}^{(hm)} H_{l-1}^{(hm)} \]

- **Hetero. View encoder – high-pass filter:**
  \[ H_0^{(ht)} = \text{MLP}^{(ht)}(X), \quad H_l^{(ht)} = \tilde{L}^{(ht)} H_{l-1}^{(ht)}, \]
  where  \[ \tilde{L}^{(ht)} = I - \alpha \tilde{A}^{(ht)} \]
Dual-channel encoding

- Homo. View encoder – low-pass filter:
  \[ \mathbf{H}_0^{(hm)} = \text{MLP}^{(hm)}(\mathbf{X}), \quad \mathbf{H}_t^{(hm)} = \tilde{\mathbf{A}}^{(hm)} \mathbf{H}_{t-1}^{(hm)} \]

- Hetero. View encoder – low-pass filter:
  \[ \mathbf{H}_0^{(ht)} = \text{MLP}^{(ht)}(\mathbf{X}), \quad \mathbf{H}_t^{(ht)} = \tilde{\mathbf{L}}^{(ht)} \mathbf{H}_{t-1}^{(ht)} \]

Node representations:
\[ \mathbf{H} = [\mathbf{H}^{(hm)} \| \mathbf{H}^{(ht)}] \in \mathbb{R}^{n \times d_r} \]
Pivot-anchored ranking loss

\[ R^{(hm)}(e_{i,j}) = s_{\ell_i,\ell_j} - s_{e_{i,j}} + \gamma^{(hm)}_+ , \]

\[ R^{(ht)}(e_{i,j}) = [s_{e_{i,j}} - s_{\ell_i,\ell_j} + \gamma^{(ht)}_+]_+ , \]

Where \( s_{e_{i,j}} = \cos(h_i, h_j) \)

Rep. sim. of connected nodes \( i, j \) where \( e_{i,j} \) is an existing edge

Rep. sim. of two randomly sampled nodes \( \ell_i, \ell_j \)

\( \gamma^{(hm)}, \gamma^{(ht)} \): margins (hyper-params)

Pivot-anchored ranking loss

\[
\mathcal{R}^{(hm)}(e_{i,j}) = [s_{v_{i}'v_{j}'} - s_{e_{i,j}} + \gamma^{(hm)}]_+,
\]

\[
\mathcal{R}^{(ht)}(e_{i,j}) = [s_{e_{i,j}} - s_{v_{i}'v_{j}'} + \gamma^{(ht)}]_+,
\]

Where \( s_{e_{i,j}} = \cos(h_{i}, h_{j}) \)

\[
\mathcal{L}_r = \mathcal{L}_r^{(hm)} + \mathcal{L}_r^{(ht)}
\]

\[
\mathcal{L}_r^{(hm)} = \frac{1}{W(hm)} \sum_{e_{i,j} \in E} \mathcal{R}^{(hm)}(e_{i,j}),
\]

\[
\mathcal{L}_r^{(ht)} = \frac{1}{W(ht)} \sum_{e_{i,j} \in E} (1 - \hat{w}_{i,j}) \mathcal{R}^{(ht)}(e_{i,j}),
\]

Dual-channel contrastive loss

\[ \mathcal{L}_c = -\frac{1}{n} \sum_{v_i \in \mathcal{V}} \left[ \frac{1}{|\mathcal{N}_i|} \sum_{v_j \in \mathcal{N}_i} \left( \log \frac{e^\frac{\cos(\mathbf{z}_i^{(hm)}, \mathbf{z}_j^{(h)})}{\tau_c}}{\sum_{v_k \in \mathcal{V} \setminus \mathcal{V}_i} e^\frac{\cos(\mathbf{z}_k^{(hm)}, \mathbf{z}_j^{(h)})}{\tau_c}} + \log \frac{e^\frac{\cos(\mathbf{z}_i^{(h)}, \mathbf{z}_j^{(hm)})}{\tau_c}}{\sum_{v_k \in \mathcal{V} \setminus \mathcal{V}_i} e^\frac{\cos(\mathbf{z}_k^{(h)}, \mathbf{z}_j^{(hm)})}{\tau_c}} \right) \right] \]

(kNN extends positive samples:
\[ \mathcal{N}_i = \text{kNN}(v_i, k) \]

Alternative training scheme

## Performance comparison

- Node classification @ homophilic graphs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN*</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
<td>76.89±0.37</td>
<td>86.34±0.48</td>
<td>92.35±0.25</td>
<td>93.10±0.17</td>
<td>95.54±0.19</td>
</tr>
<tr>
<td>GAT*</td>
<td>83.0</td>
<td>72.5</td>
<td>79.0</td>
<td>77.42±0.19</td>
<td>87.06±0.35</td>
<td>92.64±0.42</td>
<td>92.41±0.27</td>
<td>95.45±0.17</td>
</tr>
<tr>
<td>MLP</td>
<td>56.11±0.34</td>
<td>56.91±0.42</td>
<td>71.35±0.05</td>
<td>72.02±0.21</td>
<td>73.88±0.10</td>
<td>78.54±0.05</td>
<td>90.42±0.08</td>
<td>93.54±0.05</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>69.47±0.55</td>
<td>58.82±0.61</td>
<td>69.87±1.25</td>
<td>74.35±0.06</td>
<td>85.68±0.06</td>
<td>89.44±0.11</td>
<td>84.61±0.22</td>
<td>91.77±0.15</td>
</tr>
<tr>
<td>node2vec</td>
<td>71.24±0.89</td>
<td>47.64±0.77</td>
<td>66.47±1.00</td>
<td>71.79±0.05</td>
<td>84.39±0.08</td>
<td>89.67±0.12</td>
<td>85.08±0.03</td>
<td>91.19±0.04</td>
</tr>
<tr>
<td>GAE</td>
<td>71.07±0.39</td>
<td>65.22±0.43</td>
<td>71.73±0.92</td>
<td>70.15±0.01</td>
<td>85.27±0.19</td>
<td>91.62±0.13</td>
<td>90.01±0.71</td>
<td>94.92±0.07</td>
</tr>
<tr>
<td>VGAE</td>
<td>79.81±0.87</td>
<td>66.75±0.37</td>
<td>77.16±0.31</td>
<td>75.63±0.19</td>
<td>86.37±0.21</td>
<td>92.20±0.11</td>
<td>92.11±0.09</td>
<td>94.52±0.00</td>
</tr>
<tr>
<td>DGI</td>
<td>82.29±0.56</td>
<td>71.49±0.14</td>
<td>77.43±0.84</td>
<td>75.73±0.13</td>
<td>84.09±0.39</td>
<td>91.49±0.25</td>
<td>91.95±0.40</td>
<td>94.57±0.38</td>
</tr>
<tr>
<td>GMI</td>
<td>82.51±1.47</td>
<td>71.56±0.56</td>
<td>79.83±0.90</td>
<td>75.06±0.13</td>
<td>81.76±0.52</td>
<td>90.72±0.33</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>MVGRL</td>
<td>83.03±0.27</td>
<td>72.75±0.46</td>
<td>79.63±0.38</td>
<td>77.97±0.18</td>
<td>87.09±0.27</td>
<td>92.01±0.13</td>
<td>91.97±0.19</td>
<td>95.53±0.10</td>
</tr>
<tr>
<td>GRACE</td>
<td>80.08±0.53</td>
<td>71.41±0.38</td>
<td>80.15±0.34</td>
<td>79.16±0.36</td>
<td>87.21±0.44</td>
<td>92.65±0.32</td>
<td>92.78±0.23</td>
<td>95.39±0.32</td>
</tr>
<tr>
<td>GCA</td>
<td>80.39±0.42</td>
<td>71.21±0.24</td>
<td><strong>80.37±0.75</strong></td>
<td>79.35±0.12</td>
<td>87.84±0.27</td>
<td>92.78±0.17</td>
<td>93.32±0.12</td>
<td>95.87±0.15</td>
</tr>
<tr>
<td>BGRL</td>
<td>81.08±0.17</td>
<td>71.59±0.42</td>
<td>79.97±0.36</td>
<td>78.74±0.22</td>
<td><strong>88.92±0.33</strong></td>
<td><strong>93.24±0.29</strong></td>
<td>93.26±0.36</td>
<td>95.76±0.38</td>
</tr>
<tr>
<td>GREET</td>
<td><strong>83.81±0.87</strong></td>
<td><strong>73.08±0.84</strong></td>
<td><strong>80.29±1.00</strong></td>
<td><strong>80.68±0.31</strong></td>
<td><strong>87.94±0.35</strong></td>
<td><strong>92.85±0.31</strong></td>
<td><strong>94.65±0.18</strong></td>
<td><strong>96.13±0.12</strong></td>
</tr>
</tbody>
</table>


97
## Performance comparison

- **Node classification @ heterophilic graphs**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Chameleon</th>
<th>Squirrel</th>
<th>Actor</th>
<th>Cornell</th>
<th>Texas</th>
<th>Wisconsin</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>59.63±2.32</td>
<td>36.28±1.52</td>
<td>30.83±0.77</td>
<td>57.03±3.30</td>
<td>60.00±4.80</td>
<td>56.47±6.55</td>
</tr>
<tr>
<td>GAT</td>
<td>56.38±2.19</td>
<td>32.09±3.27</td>
<td>28.06±1.48</td>
<td>59.46±3.63</td>
<td>61.62±3.78</td>
<td>54.71±6.87</td>
</tr>
<tr>
<td>MLP</td>
<td>46.91±2.15</td>
<td>29.28±1.33</td>
<td>35.66±0.94</td>
<td>81.08±7.93</td>
<td>81.62±5.51</td>
<td>84.31±3.40</td>
</tr>
<tr>
<td>Geom-GCN*</td>
<td>60.90</td>
<td>38.14</td>
<td>31.63</td>
<td>60.81</td>
<td>67.57</td>
<td>64.12</td>
</tr>
<tr>
<td>H2GCN*</td>
<td>59.39±1.98</td>
<td>37.90±2.02</td>
<td>35.86±1.03</td>
<td>82.16±4.80</td>
<td>84.86±6.77</td>
<td>86.67±4.69</td>
</tr>
<tr>
<td>FAGCN</td>
<td>63.44±2.05</td>
<td>41.17±1.94</td>
<td>35.74±0.62</td>
<td>81.35±5.05</td>
<td>84.32±6.02</td>
<td>83.33±2.01</td>
</tr>
<tr>
<td>GPR-GNN</td>
<td>61.58±2.24</td>
<td>39.65±2.61</td>
<td>35.27±1.04</td>
<td>81.89±5.93</td>
<td>83.24±4.95</td>
<td>84.12±3.45</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>47.74±2.05</td>
<td>32.93±1.58</td>
<td>22.78±0.64</td>
<td>39.18±5.57</td>
<td>46.49±6.49</td>
<td>33.53±4.92</td>
</tr>
<tr>
<td>node2vec</td>
<td>41.93±3.29</td>
<td>22.84±0.72</td>
<td>28.28±1.27</td>
<td>42.94±7.46</td>
<td>41.92±7.76</td>
<td>37.45±7.09</td>
</tr>
<tr>
<td>GAE</td>
<td>33.84±2.77</td>
<td>28.03±1.61</td>
<td>28.03±1.18</td>
<td>58.85±3.21</td>
<td>58.64±4.53</td>
<td>52.55±3.80</td>
</tr>
<tr>
<td>VGAE</td>
<td>35.22±2.71</td>
<td>29.48±1.48</td>
<td>26.99±1.56</td>
<td>59.19±4.09</td>
<td>59.20±4.26</td>
<td>56.67±5.51</td>
</tr>
<tr>
<td>DGI</td>
<td>39.95±1.75</td>
<td>31.80±0.77</td>
<td>29.82±0.69</td>
<td>63.35±4.61</td>
<td>60.59±7.56</td>
<td>55.41±5.96</td>
</tr>
<tr>
<td>GMI</td>
<td>46.97±3.43</td>
<td>30.11±1.92</td>
<td>27.82±0.90</td>
<td>54.76±5.06</td>
<td>50.49±2.21</td>
<td>45.98±2.76</td>
</tr>
<tr>
<td>MVGRL</td>
<td>51.07±2.68</td>
<td>35.47±1.29</td>
<td>30.02±0.70</td>
<td>64.30±5.43</td>
<td>62.38±5.61</td>
<td>62.37±4.32</td>
</tr>
<tr>
<td>GRACE</td>
<td>48.05±1.81</td>
<td>31.33±1.22</td>
<td>29.01±0.78</td>
<td>54.86±6.95</td>
<td>57.57±5.68</td>
<td>50.00±5.83</td>
</tr>
<tr>
<td>GRACE-FA</td>
<td>52.68±2.14</td>
<td>35.97±1.20</td>
<td>32.55±1.28</td>
<td>67.57±4.98</td>
<td>64.05±7.46</td>
<td>63.73±6.81</td>
</tr>
<tr>
<td>GCA</td>
<td>49.80±1.81</td>
<td>35.50±0.91</td>
<td>29.65±1.47</td>
<td>55.41±4.56</td>
<td>59.46±6.16</td>
<td>50.78±4.06</td>
</tr>
<tr>
<td>BORL</td>
<td>47.46±2.74</td>
<td>32.64±0.78</td>
<td>29.86±0.75</td>
<td>57.30±5.51</td>
<td>59.19±5.85</td>
<td>52.35±4.12</td>
</tr>
<tr>
<td>GREET</td>
<td>63.64±1.26</td>
<td>42.29±1.43</td>
<td>36.55±1.01</td>
<td>85.14±4.87</td>
<td>87.03±2.36</td>
<td>84.90±4.48</td>
</tr>
</tbody>
</table>
Heterogenous Graph Self-supervised Learning
Heterogeneous Graphs

(a) Heterogeneous Graph
(b) Network Schema
(c) Meta-path

Figure 1: A toy example of HIN (ACM) and relative illustrations of meta-path and network schema.

HeCo Framework – (View Generation)

Figure 2: The overall architecture of our proposed HeCo.

Meta-path View

\[ w_{\phi_m} = \frac{1}{|V|} \sum_{i \in V} a^{\top}_{sc} \cdot \tanh \left( W_{sc} h_i^{\Phi_m} + b_{sc} \right), \]

\[ \beta_{\phi_m} = \frac{\exp \left( w_{\phi_m} \right)}{\sum_{i=1}^S \exp \left( w_{\phi_i} \right)}, \]

\[ z^{sc}_i = \sum_{m=1}^S \beta_{\phi_m} \cdot h_i^{\Phi_m}. \]
HeCo Framework – (Contrastive Learning)

Figure 3: A schematic diagram of view mask mechanism.

Masked Node in Network Schema View

Masked People/Subjects in Meta-path View

\[
\begin{align*}
    z_i^{sc \_proj} &= W^{(2)} \sigma \left( W^{(1)} z_i^{sc} + b^{(1)} \right) + b^{(2)}, \\
    z_i^{mp \_proj} &= W^{(2)} \sigma \left( W^{(1)} z_i^{mp} + b^{(1)} \right) + b^{(2)},
\end{align*}
\]

\[
L_i^{sc} = - \log \frac{\sum_{j \in \mathcal{P}_i} \exp \left( \frac{\text{sim} \left( z_i^{sc \_proj}, z_j^{mp \_proj} \right)}{\tau} \right)}{\sum_{k \in \{\mathcal{P}_i \cup \mathcal{N}_i\}} \exp \left( \frac{\text{sim} \left( z_i^{sc \_proj}, z_k^{mp \_proj} \right)}{\tau} \right)},
\]

\[
J = \frac{1}{|V|} \sum_{i \in V} \left[ \lambda \cdot L_i^{sc} + (1 - \lambda) \cdot L_i^{mp} \right],
\]

HeCo Framework – (Contrastive Learning)

We can obtain $L^{\text{mp}}$ similarly

$$
L_i^{sc} = -\log \frac{\sum_{j \in P_i} \exp \left( \frac{\text{sim}(z_{i \text{sc profund}, j \text{mp profund})}{\tau}}{\sum_{k \in (P_i \cup N_i)} \exp \left( \frac{\text{sim}(z_{i \text{sc profund}, z_{k \text{mp profund})}{\tau}}}{\tau} \right)}
$$

$$
z_i^{sc \text{ profund}} = W^{(2)} \sigma \left( W^{(1)} z_i^{sc} + b^{(1)} \right) + b^{(2)}
$$

$$
z_i^{mp \text{ profund}} = W^{(2)} \sigma \left( W^{(1)} z_i^{mp} + b^{(1)} \right) + b^{(2)}
$$

Figure 3: A schematic diagram of view mask mechanism.

$$
J = \frac{1}{|V|} \sum_{i \in V} \left[ \lambda \cdot L_i^{sc} + (1 - \lambda) \cdot L_i^{mp} \right]
$$

Table 2: Quantitative results (%±e) on node classification.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Metric</th>
<th>Split</th>
<th>GraphSAGE</th>
<th>GAE</th>
<th>MP2vec</th>
<th>HERec</th>
<th>HetGNN</th>
<th>HAN</th>
<th>DGI</th>
<th>DMGI</th>
<th>HeCo</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ma-F1</td>
<td>20</td>
<td>47.13±4.7</td>
<td>62.72±3.1</td>
<td>51.91±0.9</td>
<td>55.13±1.5</td>
<td>72.11±0.9</td>
<td>85.66±2.1</td>
<td>79.27±3.8</td>
<td>87.86±0.2</td>
<td>88.56±0.8</td>
<td>87.61±0.5</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>55.96±6.8</td>
<td>61.61±3.2</td>
<td>62.41±0.6</td>
<td>62.11±0.8</td>
<td>72.02±0.4</td>
<td>87.47±1.1</td>
<td>80.23±3.3</td>
<td>86.23±0.8</td>
<td>85.04±0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>56.59±5.7</td>
<td>61.67±2.9</td>
<td>61.13±0.4</td>
<td>64.35±0.8</td>
<td>74.33±0.6</td>
<td>88.41±1.1</td>
<td>80.03±3.3</td>
<td>87.94±0.7</td>
<td>86.87±0.8</td>
<td></td>
</tr>
<tr>
<td>Mi-F1</td>
<td>20</td>
<td>49.72±3.5</td>
<td>68.62±1.9</td>
<td>53.13±0.9</td>
<td>57.47±1.5</td>
<td>71.89±1.1</td>
<td>85.11±2.2</td>
<td>79.63±3.5</td>
<td>87.69±0.8</td>
<td>88.13±0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>50.94±5.6</td>
<td>69.38±1.9</td>
<td>64.35±0.6</td>
<td>64.48±1.2</td>
<td>74.68±1.2</td>
<td>88.24±1.2</td>
<td>80.41±3.0</td>
<td>87.92±0.9</td>
<td>87.42±0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>60.22±4.3</td>
<td>75.61±2.2</td>
<td>72.62±0.3</td>
<td>65.15±0.9</td>
<td>76.08±0.7</td>
<td>88.10±1.2</td>
<td>80.15±3.2</td>
<td>87.82±0.5</td>
<td>88.71±0.5</td>
<td></td>
</tr>
<tr>
<td>AUC</td>
<td>20</td>
<td>65.88±3.7</td>
<td>79.50±2.4</td>
<td>71.66±0.7</td>
<td>74.51±1.3</td>
<td>84.36±1.0</td>
<td>93.47±1.3</td>
<td>91.47±2.3</td>
<td>96.72±0.3</td>
<td>96.92±0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>71.06±5.2</td>
<td>79.14±2.5</td>
<td>80.48±0.4</td>
<td>79.34±0.5</td>
<td>85.01±0.6</td>
<td>94.84±0.9</td>
<td>91.52±2.3</td>
<td>96.35±0.3</td>
<td>96.40±0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>79.45±6.2</td>
<td>77.99±2.8</td>
<td>79.33±0.4</td>
<td>84.11±1.4</td>
<td>86.48±1.3</td>
<td>94.11±1.9</td>
<td>94.79±0.7</td>
<td>96.79±0.2</td>
<td>95.55±0.4</td>
<td></td>
</tr>
<tr>
<td>Ma-F1</td>
<td>20</td>
<td>71.79±8.4</td>
<td>90.90±0.1</td>
<td>88.98±0.2</td>
<td>89.57±0.4</td>
<td>89.51±1.1</td>
<td>89.31±0.9</td>
<td>87.93±2.4</td>
<td>89.24±0.9</td>
<td>91.28±0.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>73.69±8.4</td>
<td>92.69±0.3</td>
<td>88.68±0.2</td>
<td>89.73±0.4</td>
<td>88.61±0.8</td>
<td>88.87±1.0</td>
<td>88.62±2.6</td>
<td>89.23±0.4</td>
<td>90.34±0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>73.86±8.1</td>
<td>90.88±0.2</td>
<td>90.25±1.1</td>
<td>90.18±0.3</td>
<td>89.56±0.5</td>
<td>89.20±0.8</td>
<td>89.19±0.9</td>
<td>89.40±0.6</td>
<td>90.64±0.3</td>
<td></td>
</tr>
<tr>
<td>DBLP</td>
<td>20</td>
<td>71.44±8.7</td>
<td>91.55±0.1</td>
<td>89.67±0.1</td>
<td>90.24±0.4</td>
<td>90.11±1.0</td>
<td>90.16±0.9</td>
<td>89.72±2.6</td>
<td>90.78±0.3</td>
<td>91.97±0.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>73.61±8.6</td>
<td>90.69±0.3</td>
<td>89.16±0.2</td>
<td>90.15±0.4</td>
<td>89.03±0.7</td>
<td>89.47±0.9</td>
<td>89.22±0.5</td>
<td>89.92±0.4</td>
<td>90.76±0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>74.65±8.3</td>
<td>90.95±0.2</td>
<td>91.17±1.0</td>
<td>90.10±0.3</td>
<td>90.43±0.6</td>
<td>90.34±0.8</td>
<td>90.35±0.8</td>
<td>90.66±0.5</td>
<td>91.59±0.2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>29.20</td>
<td>27.72</td>
</tr>
<tr>
<td>GAE</td>
<td>72.59</td>
<td>73.71</td>
</tr>
<tr>
<td>MP2vec</td>
<td>30.80</td>
<td>25.28</td>
</tr>
<tr>
<td>HERec</td>
<td>47.54</td>
<td>35.67</td>
</tr>
<tr>
<td>HetGNN</td>
<td>61.19</td>
<td>59.12</td>
</tr>
<tr>
<td>DGI</td>
<td>51.73</td>
<td>41.63</td>
</tr>
<tr>
<td>DMGI</td>
<td>51.66</td>
<td>46.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMiner</td>
<td>36.87</td>
<td>36.94</td>
</tr>
</tbody>
</table>

Table 3: Quantitative results (%±e) on node clustering.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ACM</th>
<th>DBLP</th>
<th>Freebase</th>
<th>AMiner</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMI</td>
<td>ARI</td>
<td>NMI</td>
<td>ARI</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>29.20</td>
<td>27.72</td>
<td>51.50</td>
<td>36.40</td>
</tr>
<tr>
<td>GAE</td>
<td>27.42</td>
<td>24.49</td>
<td>72.59</td>
<td>73.71</td>
</tr>
<tr>
<td>MP2vec</td>
<td>48.43</td>
<td>34.65</td>
<td>73.55</td>
<td>77.70</td>
</tr>
<tr>
<td>HERec</td>
<td>47.54</td>
<td>35.67</td>
<td>70.21</td>
<td>73.99</td>
</tr>
<tr>
<td>HetGNN</td>
<td>41.33</td>
<td>34.81</td>
<td>69.79</td>
<td>75.34</td>
</tr>
<tr>
<td>DGI</td>
<td>51.73</td>
<td>41.63</td>
<td>59.23</td>
<td>61.85</td>
</tr>
<tr>
<td>DMGI</td>
<td>51.66</td>
<td>46.64</td>
<td>70.06</td>
<td>75.46</td>
</tr>
</tbody>
</table>

Part 4: Applications of graph self-supervised learning

• Recommender system
• Outlier detection
• More applications: Chemistry, graph structure learning…
Graphs in recommender system

Users → Items

Interaction → User-item graph

Item transmission/similarity → Item-item graph (for sequential/session-based recommendation)

Social relation → User-user graph (for social recommendation)
GNNs for recommender system

LightGCN for collaborative filtering

SR-GNN for session-based recommendation

GraphRec for social recommendation


GSSL for recommendation: Motivations

Learning scheme: observed interactions $\rightarrow$ ranking loss (e.g. BPR)

$$\mathcal{L}_{\text{main}} = \sum_{(u, i, j) \in \mathcal{O}} -\log \sigma(\hat{y}_{ui} - \hat{y}_{uj}),$$

- Problem 1: Sparse Supervision Signal
The observed interactions can be extremely sparse compared to the whole interaction space

**GSSL:**
**provide extra supervision signals from data itself!**
GSSL for recommendation: Motivations

Learning scheme: observed interactions $\rightarrow$ ranking loss (e.g. BPR)

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in O} - \log \sigma(\hat{y}_{ui} - \hat{y}_{uj}),$$

- Problem 1: Sparse Supervision Signal
- Problem 2: Noisy interaction

Observed interactions usually contain noises, e.g., a user is misled to click an item and finds it uninteresting after consuming it

**GSSL:**
- **Regularize the model to prevent it from over-fitting the noisy interaction**
- **Data augmentations to reduce the impact by noise**

Contrast-based method

Scenario: collaborative filtering recommendation

Augmentations: Node Dropout (ND), Edge Dropout (ED), and Random Walk (RW)

Contrast-based method

Following works of SGL: Scenario: collaborative filtering recommendation

SimGCL

Original Graph → Graph Encoder → Representation → Joint Learning → Recommendation

Original

Perturbed

$e' = e_i + \Delta'_i, \ e'' = e_i + \Delta''_i$

representation-level augmentation!

LightGCL

Original Graph → ApproxSVD → Reconstruct → GCNs → Pairwise Loss for Recommendation

SVD-based augmentation

$\mathcal{L}_r$

$\mathcal{L}_s^{(u)}, \mathcal{L}_s^{(v)}$


Contrast-based method

SEPT

Scenario: social recommendation

Contrast-based method

COTREC

Scenario: session-based recommendation

Generation-based method

PMGT

Scenario: Multimodal Side Information-based Recommendation

(a) Framework of PMGT

(b) Node Embedding Initialization

Loss:

\[ \mathcal{L}_{edge} = \frac{1}{|V|} \sum_{h \in V} \frac{1}{|N_h|} \sum_{t \in N_h} \left[ -\log \left( \sigma \left( \frac{h^T_t}{||t||_2} \right) \right) \right] \]

\[ -Q \cdot \mathbb{E}_{t \sim P_t} \log \left( \sigma \left( -\frac{h^T_t}{||t||_2} \frac{h^T_t}{||t||_2} \right) \right) \]

Feature reconstruction:

\[ \mathcal{L}_{feature} = \frac{1}{|V|} \sum_{h \in V} \frac{1}{|M_h|} \sum_{t \in M_h} \sum_{i} ||h^T_t w^i_t - x^i_t||_2^2 \]

Generation-based method

MAERec

Scenario: Sequential Recommendation

\[ L_{mask} = - \sum_{v \in V} y'(v) \]

"Learning to mask" loss

\[ L_{con} = - \sum_{(a, a') \in \mathcal{E} \setminus \mathcal{P}^k} \log \frac{\exp(s_{a, a'})}{\sum_{a'' \in \mathcal{Y}} \exp(s_{a, a''})} \]

Reconstruction loss: recovering the masked global item transition paths
Hybrid method

Scenario: Sequential Recommendation

Three tasks:
- Masked Node Prediction (MNP)
- Masked Edge Prediction (MEP)
- Meta-path Type Prediction (MTP)

Summary: GSSL for recommender systems

• **Scenarios**
  - Collaborative filtering-based recommendation
  - Social recommendation
  - Session-based recommendation
  - Sequential recommendation
  - …

• **Pretext tasks**
  - Mainstream solution: Contrast-based GSSL
  - Promising directions: Generation-based and hybrid GSSL

• **Representative methods**

SGL

MAERec
Graph-based outlier detection

- Abnormal Nodes
- Abnormal Edges
- Abnormal Communities

Anomaly detection

Fraud detection

Out-of-distribution detection

https://neo4j.com/blog/enterprise-fraud-detection/
Graph-based outlier detection

Social Network Graph

Anomaly Detection

Social bots

Rumors

Fake news

......
Graph-based outlier detection

https://neo4j.com/blog/enterprise-fraud-detection/
The lack of annotated labels for outliers:

**Challenge:** It’s difficult to annotate the anomalies/out-of-distribution samples from numerous normal sample!
GSSL for outlier detection: motivation

It’s difficult to annotate the anomalies/out-of-distribution samples from numerous normal sample!

Self-supervised methods:
capture the latent patterns of normal data without any label
→ the model can find the outlier according to its normality

Capture the normal patterns from itself!
Generation-based method

DOMINANT

Scenario: node-level anomaly detection

Generation-based method

AnomalyDAE

Scenario: node-level anomaly detection

GUIDE

Scenario: node-level anomaly detection

Consider various motifs in structure-based auto-encoder

Contrast-based method

CoLA

Scenario: node-level anomaly detection

Contrast-based method

ANEMONE

Scenario: node-level anomaly detection

Multi-scale contrastive learning!
Contrast-based method

GOOD-D

Scenario: graph-level out-of-distribution/anomaly detection

Auxiliary property-based method

**Sub-CR**

Scenario: node-level anomaly detection

Hybrid method

SL-GAD

Scenario: node-level anomaly detection

Contrast-based + generation-based

Hybrid method

GLADC

Scenario: graph-level anomaly detection

Contrast-based + generation-based

Summary: GSSL for outlier detection

• **Scenarios**
  - Node-level
  - Graph-level

• **Pretext tasks**
  - Early methods: generation-based: autoencoder
  - Mainstream methods: contrast-based: from single scale to multi-scale
  - A new perspective: auxiliary property – predict the hop
  - Advanced solutions: hybrid GSSL

• **Representative methods**

![Diagram]

DOMINANT

CoLA
More applications: chemistry

GROVER for molecular pre-train model

Contextual property prediction (node/edge level task)

Graph-level motif prediction

MIRACLE for drug-drug interaction prediction


More applications: graph structure learning

SLAPS

SUBLIME


Summary

- **Recommender Systems**
- **Outlier Detection**
- **Chemistry**
- **Graph Structure Learning**
- **Boarder Applications**
  - Expert finding
  - Program repairing
  - Open world modeling
  - Medical
  - Federated Learning
  ...
Part 5: Future directions and conclusion

• Potential directions of graph self-supervised learning
• Conclusion
Future Directions

• Theoretical Foundation

The existing methods are mostly designed with intuition and their performance gain is evaluated by empirical experiments, but don’t have a solid theoretical foundation.

Potential theoretical basis:

Information theory

Spectral graph theory

Future Directions

• Interpretability and Robustness

Most of the current works lack these properties.
Interpretability: Explainable GSSL model
Robustness: adversarial attack/defense of GSSL model

Interpretability


Adversarial Attack
Future Directions

• Pretext Tasks for Complex Types of Graphs

Most of the existing works: Plain graph, Attributed graph

Potential targets:

Hypergraph
Dynamic graph
Spatial-temporal graph
Heterogeneous graph
Future Directions

- Augmentation for Graph Contrastive Learning

Existing augmentations: Feature and/or structure perturbing.

Can we develop more effective augmentation strategy for graphs?

Knowledge-based augmentation

- Atom masking
- Bond deletion
- Subgraph removal

Spectral-based augmentation

---

Future Directions

• Learning with Multiple Pretext Tasks
  How to effectively leverage different pretext tasks? Can we select pretext tasks automatically?

• Broader Scope of Applications
  Can we apply GSSL to more graph-related scenarios?
Conclusion

• Background

Graph neural networks

Self-supervised learning on graph: acquires supervision signals from data itself for graph-based deep learning models.
Conclusion

• Graph self-supervised learning: Taxonomy
Conclusion

• Graph self-supervised learning: Frontiers

Efficient GSSL paradigm: Group Discrimination

GSSL for Heterophilic graph

GSSL for Heterogeneous graph

https://ai.googleblog.com/2023/03/teaching-old-labels-new-tricks-in.html
Conclusion

• Graph self-supervised learning: Applications
  • Recommender Systems
  • Outlier Detection
  • Chemistry
  • Graph Structure Learning

Boarder Applications…
Thanks for listening!
Q&A