I'm Me, We're Us, and I'm Us: Tri-directional Contrastive Learning on Hypergraphs

Dongjin Lee[1] **and Kijung Shin**[1,2]
[1]School of Electrical Engineering, KAIST, South Korea
[2]Kim Jaechul Graduate School of AI, KAIST, South Korea
{dongjin.lee, kijungs}@kaist.ac.kr

Abstract

Although machine learning on hypergraphs has attracted considerable attention, most of the works have focused on (semi-)supervised learning, which may cause heavy labeling costs and poor generalization. Recently, contrastive learning has emerged as a successful unsupervised representation learning method. Despite the prosperous development of contrastive learning in other domains, contrastive learning on hypergraphs remains little explored. In this paper, we propose **TriCL** (Tri-directional Contrastive Learning), a general framework for contrastive learning on hypergraphs. Its main idea is tri-directional contrast, and specifically, it aims to maximize in two augmented views the agreement (a) between the same node, (b) between the same group of nodes, and (c) between each group and its members. Together with simple but surprisingly effective data augmentation and negative sampling schemes, these three forms of contrast enable TriCL to capture both node- and group-level structural information in node embeddings. Our extensive experiments using 14 baseline approaches, 10 datasets, and two tasks demonstrate the effectiveness of TriCL, and most noticeably, TriCL almost consistently outperforms not just unsupervised competitors but also (semi-)supervised competitors mostly by significant margins for node classification. The code and datasets are available at https://github.com/wooner49/TriCL.

1 Introduction

Many real-world interactions are group-wise. Examples include collaborations of researchers, group conversations on messaging apps, co-citations of documents, and co-purchases of items. A *hypergraph*, which is a generalized graph, allows an edge to join an arbitrary number of nodes, and thus each such edge, which is called a *hyperedge*, naturally represents such group-wise interactions (Benson et al. 2018; Do et al. 2020; Lee, Ko, and Shin 2020).

Recently, machine learning on hypergraphs has drawn a lot of attention from a broad range of fields, including social network analysis (Yang et al. 2019), recommender systems (Xia et al. 2021), and bioinformatics (Zheng et al. 2019). Hypergraph-based approaches often outperform graph-based ones on various machine learning tasks, including classification (Feng et al. 2019), clustering (Benson, Gleich, and Leskovec 2016), ranking (Yu et al. 2021), and outlier detection (Lee, Choe, and Shin 2022).

Previous studies have largely focused on developing encoder architectures so-called *hypergraph neural networks* for hypergraph-structured data (Feng et al. 2019; Yadati et al. 2019; Dong, Sawin, and Bengio 2020; Bai, Zhang, and Torr 2021; Arya et al. 2020), and in most cases, such hypergraph neural networks are trained in a (semi-)supervised way. However, data labeling is often time, resource, and labor-intensive, and neural networks trained only in a supervised way can easily overfit and may fail to generalize (Rong et al. 2020), making it difficult to be applied to other tasks.

Thus, self-supervised learning (Liu et al. 2022; Jaiswal et al. 2020; Liu et al. 2021), which does not require labels, has become popular, and especially contrastive learning has achieved great success in computer vision (Chen et al. 2020; Hjelm et al. 2019) and natural language processing (Gao, Yao, and Chen 2021). Contrastive learning has proved effective also for learning on (ordinary) graphs (Veličković et al. 2018; Peng et al. 2020; Hassani and Khasahmadi 2020; Zhu et al. 2020, 2021b; You et al. 2020), and a common approach is to (a) create two augmented views from the input graph and (b) learn machine learning models to maximize the agreement between the two views.

However, contrastive learning on hypergraphs remains largely underexplored with only a handful of previous studies (Xia et al. 2021; Zhang et al. 2021; Yu et al. 2021) (see Section 2 for details). Especially, the following questions remain open: (Q1) what to contrast?, (Q2) how to augment a hypergraph?, and (Q3) how to select negative samples?

For Q1, which is our main focus, we propose *tri-directional contrast*. In addition to *node-level contrast*, which is the only form of contrast employed in the previous studies, we propose the use of *group-level* and *membership-level* contrast. That is, in two augmented views, we aim to maximize agreements (a) between the same node, (b) between the same group of nodes, and (c) between each group and its members. These three forms of contrast are complementary, leading to representations that capture both node- and group-level (i.e., higher-order) relations in hypergraphs.

In addition, for Q2, we demonstrate that combining two simple augmentation strategies (spec., membership corruption and feature corruption) is effective. For Q3, we reveal that uniform random sampling is surprisingly successful, and in our experiments, even an extremely small sample size leads to marginal performance degradation.

Our method TriCL, which is based on the aforementioned observations, is evaluated extensively using 14 baseline approaches, 10 datasets (including three used in Appendix E), and two tasks. The most notable result is that, for node classification, TriCL outperforms not just unsupervised competitors but also all (semi-)supervised competitors on almost all considered datasets, mostly by considerable margins. Moreover, we demonstrate the consistent effectiveness of tri-directional contrast, which is our main contribution.

2 Related Work

Hypergraph learning Due to its enough expressiveness to capture higher-order structural information, learning on hypergraphs has received a lot of attention. Many recent studies have focused on generalizing graph neural networks (GNNs) to hypergraphs (Feng et al. 2019; Bai, Zhang, and Torr 2021; Yadati et al. 2019). Most of them redefine hypergraph message aggregation schemes based on clique expansion (i.e., replacing hyperedges with cliques to obtain a graph) or its variants. While its simplicity is appealing, clique expansion causes structural distortion and leads to undesired information loss (Hein et al. 2013; Li and Milenkovic 2018). On the other hand, HNHN (Dong, Sawin, and Bengio 2020) prevents information loss by extending star expansion using two distinct weight matrices for node- and hyperedge-side message aggregations. Arya et al. (2020) propose HyperSAGE for inductive learning on hypergraphs based on two-stage message aggregation. Several studies attempt to unify hypergraphs and GNNs (Huang and Yang 2021; Zhang et al. 2022); and Chien et al. (2022) generalize message aggregation methods as multiset functions learned by Deep Sets (Zaheer et al. 2017) and Set Transformer (Lee et al. 2019). Most approaches above use (semi-)supervised learning.

Contrastive learning In the image domain, the latest contrastive learning frameworks (e.g., SimCLR (Chen et al. 2020) and MoCo (He et al. 2020)) leverage the unchanging semantics under various image transformations, such as random flip, rotation, color distortion, etc, to learn visual features. They aim to learn distinguishable representations by contrasting positive and negative pairs.

In the graph domain, DGI (Veličković et al. 2018) combines the power of GNNs and contrastive learning, which seeks to maximize the mutual information between node embeddings and graph embeddings. Recently, a number of graph contrastive learning approaches (You et al. 2020; Zhu et al. 2020, 2021b; Hassani and Khasahmadi 2020) that follow a common framework (Chen et al. 2020) have been proposed. Although these methods have achieved state-of-the-art performance on their task of interest, they cannot naturally exploit group-wise interactions, which we focus on in this paper. More recently, gCooL (Li, Jing, and Tong 2022) utilizes community contrast, which is a similar concept to membership-level contrast in TriCL, to maximize community consistency between two augmented views. However, gCooL has an information loss when constructing a community, thus information on subgroups (i.e., a smaller group in a large community) cannot be used. On the other hand, TriCL can preserve and fully utilize such group information.

Hypergraph contrastive learning Contrastive learning on hypergraphs is still in its infancy. Recently, several studies explore contrastive learning on hypergraphs (Zhang et al. 2021; Xia et al. 2021; Yu et al. 2021). For example, Zhang et al. (2021) proposes S^2-HHGR for group recommendation, which applies contrastive learning to remedy a data sparsity issue. In particular, they propose a hypergraph augmentation scheme that uses a coarse- and fine-grained node dropout for each view. However, they do not consider group-wise contrast. Although Xia et al. (2021) employ group-wise contrast for session recommendation, they do not account for node-wise and node-group pair-wise relationships when constructing their contrastive loss. Moreover, these approaches have been considered only in the context of group-based recommendation but not in the context of general representation learning.

3 Proposed Method: TriCL

In this section, we describe TriCL, our proposed framework for hypergraph contrastive learning. First, we introduce some preliminaries on hypergraphs and hypergraph neural networks, and then we elucidate the problem setting and details of the proposed method.

3.1 Preliminaries

Hypergraphs and notation. A *hypergraph*, a set of hyperedges, is a natural extension of a graph, allowing the hyperedge to contain any number of nodes. Formally, let $H = (V, E)$ be a hypergraph, where $V = \{v_1, v_2, \dots, v_{|V|}\}$ is a set of nodes and $E = \{e_1, e_2, \dots, e_{|E|}\}$ is a set of hyperedges, with each hyperedge is a non-empty subset of V. The node feature matrix is represented by $\mathbf{X} \in \mathbb{R}^{|V|\times F}$, where $\boldsymbol{x}_i = \mathbf{X}[i,:]^T \in \mathbb{R}^F$ is the feature of node v_i. In general, a hypergraph can alternatively be represented by its *incidence matrix* $\mathbf{H} \in \{0,1\}^{|V|\times|E|}$, with entries defined as $h_{ij} = 1$ if $v_i \in e_j$, and $h_{ij} = 0$ otherwise. In other words, $h_{ij} = 1$ when node v_i and hyperedge e_j form a *membership*. Each hyperedge $e_j \in E$ is assigned a positive weight w_j, and all the weights formulate a diagonal matrix $\mathbf{W} \in \mathbb{R}^{|E|\times|E|}$. We use the diagonal matrix $\mathbf{D}_V$ to represent the degree of vertices, where its entries $d_i = \sum_j w_j h_{ij}$. Also we use the diagonal matrix $\mathbf{D}_E$ to denote the degree of hyperedges, where its element $\delta_j = \sum_i h_{ij}$ represents the number of nodes connected by the hyperedge e_j.

Hypergraph neural networks. Modern hypergraph neural networks (Feng et al. 2019; Yadati et al. 2019; Bai, Zhang, and Torr 2021; Dong, Sawin, and Bengio 2020; Arya et al. 2020; Chien et al. 2022) follow a two-stage neighborhood aggregation strategy: node-to-hyperedge and hyperedge-to-node aggregation. They iteratively update the representation of a hyperedge by aggregating representations of its incident nodes and the representation of a node by aggregating representations of its incident hyperedges. Let $\mathbf{P}^{(k)} \in \mathbb{R}^{|V|\times F'_k}$ and $\mathbf{Q}^{(k)} \in \mathbb{R}^{|E|\times F''_k}$ be the node

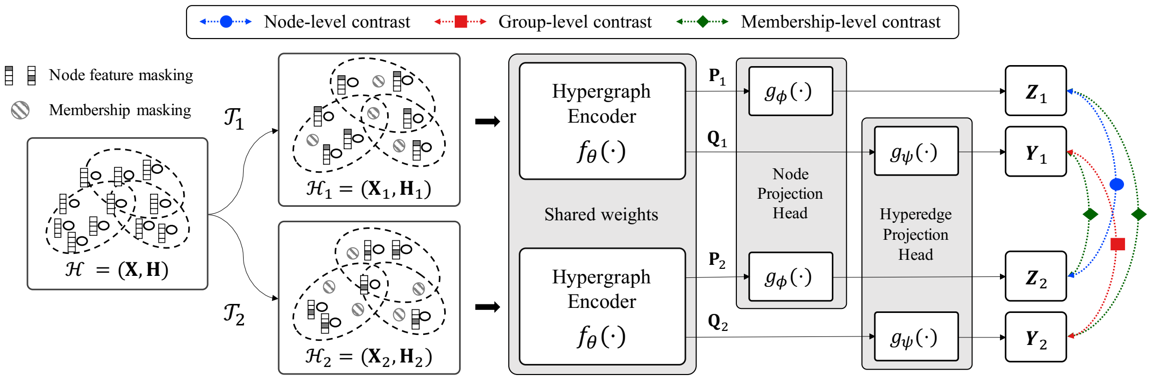

Figure 1: Overview of our proposed TriCL method. First, two different semantically similar views are generated by augmentations $\mathcal{T}_1$ and $\mathcal{T}_2$ from the original hypergraph. From these, we use a shared hypergraph encoder $f_\theta(\cdot)$ to form node and hyperedge representations. After passing node and hyperedge representations to their respective projection heads (i.e., $g_\phi(\cdot)$ and $g_\psi(\cdot)$), we maximize the agreement between two views via our proposed tri-directional contrast, which is a combination of node-, group-, and membership-level contrast.

and hyperedge representations at the k-th layer, respectively. Formally, the k-th layer of a hypergraph neural network is

$$\begin{aligned} \boldsymbol{q}_j^{(k)} &= f_{V\to E}^{(k)}\left(\boldsymbol{q}_j^{(k-1)}, \{\boldsymbol{p}_i^{(k-1)} : v_i \in e_j\}\right), \\ \boldsymbol{p}_i^{(k)} &= f_{E\to V}^{(k)}\left(\boldsymbol{p}_i^{(k-1)}, \{\boldsymbol{q}_j^{(k)} : v_i \in e_j\}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{p}_i^{(0)} = \boldsymbol{x}_i$. The choice of aggregation rules, $f_{V\to E}(\cdot)$ and $f_{E\to V}(\cdot)$, is critical, and a number of models have been proposed.

3.2 Problem Setting: Hypergraph-based Contrastive Learning

Our objective is to train a hypergraph encoder, $f_\theta : \mathbb{R}^{|V|\times F} \times \mathbb{R}^{|V|\times|E|} \to \mathbb{R}^{|V|\times F'} \times \mathbb{R}^{|E|\times F''}$, such that $f_\theta(\mathbf{X}, \mathbf{H}) = (\mathbf{P}, \mathbf{Q})$ produces low-dimensional representations of nodes and hyperedges in a fully unsupervised manner, specifically a contrastive manner. These representations may then be utilized for downstream tasks, such as node classification and clustering.

3.3 TriCL: Tri-directional Contrastive Learning

Basically, TriCL follows the conventional multi-view graph contrastive learning paradigm, where a model aims to maximize the agreement of representations between different views (You et al. 2020; Hassani and Khasahmadi 2020; Zhu et al. 2020). While most existing approaches only use node-level contrast, TriCL applies three forms of contrast for each of the three essential elements constituting hypergraphs: nodes, hyperedges, and node-hyperedge memberships. Figure 1 visually summarizes TriCL's architecture. TriCL is composed of the following four major components:

(1) Hypergraph augmentation. We consider a hypergraph $\mathcal{H} = (\mathbf{X}, \mathbf{H})$. TriCL first generates two alternate views of the hypergraph $\mathcal{H}$: $\mathcal{H}_1 = (\mathbf{X}_1, \mathbf{H}_1)$ and $\mathcal{H}_2 = (\mathbf{X}_2, \mathbf{H}_2)$, by applying stochastic hypergraph augmentation function $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. We use a combination of random *node feature masking* (You et al. 2020; Zhu et al. 2020) and *membership masking* to augment a hypergraph in terms of attributes and structure. Following previous studies (You et al. 2020; Thakoor et al. 2022), node feature masking is not applied to each node independently, and instead, we generate a single random binary mask of size F where each entry is sampled from a Bernoulli distribution $\mathcal{B}(1 - p_f)$, and we use it to mask features of all nodes.Similarly, we use a binary mask of size $K = nnz(\mathbf{H})$ where each element is sampled from a Bernoulli distribution $\mathcal{B}(1 - p_m)$ to mask node-hyperedge memberships. The degree of augmentation is controlled by p_f and p_m, and we can adopt different hyperparameters for each augmented view. More details on hypergraph augmentation are provided in Appendix D.

(2) Hypergraph encoder. A hypergraph encoder $f_\theta(\cdot)$ produces node and hyperedge representations, $\mathbf{P}$ and $\mathbf{Q}$, respectively, for two augmented views: $(\mathbf{P}_1, \mathbf{Q}_1) := f_\theta(\mathbf{X}_1, \mathbf{H}_1)$ and $(\mathbf{P}_2, \mathbf{Q}_2) := f_\theta(\mathbf{X}_2, \mathbf{H}_2)$. TriCL does not constrain the choice of hypergraph encoder architectures if they can be formulated by (1). In our proposed method, we use the element-wise mean pooling layer as a special instance of (1) (see Appendix E.2 for comparison with an alternative). That is, $f_{V\to E}$ and $f_{E\to V}$ are as:

$$\begin{aligned} \boldsymbol{q}_j^{(k)} &= \sigma\left(\sum_{v_i\in e_j} \frac{\boldsymbol{p}_i^{(k-1)}\boldsymbol{\Theta}_E^{(k)}}{\delta_j} + \boldsymbol{b}_E^{(k)}\right), \\ \boldsymbol{p}_i^{(k)} &= \sigma\left(\sum_{e_j : v_i\in e_j} \frac{w_j \boldsymbol{q}_j^{(k)}\boldsymbol{\Theta}_V^{(k)}}{d_i} + \boldsymbol{b}_V^{(k)}\right), \end{aligned} \tag{2}$$

where $\boldsymbol{\Theta}_E^{(k)}$ and $\boldsymbol{\Theta}_V^{(k)}$ are trainable weights and $\boldsymbol{b}_E^{(k)}$ and $\boldsymbol{b}_V^{(k)}$ are trainable biases. We use $w_j = 1$ for simplicity, and (2) can be represented as (3) in matrix form.

$$\begin{aligned} \mathbf{Q}^{(k)} &= \sigma\big(\mathbf{D}_E^{-1}\mathbf{H}^T\mathbf{P}^{(k-1)}\boldsymbol{\Theta}_E^{(k)} + \boldsymbol{b}_E^{(k)}\big), \\ \mathbf{P}^{(k)} &= \sigma\big(\mathbf{D}_V^{-1}\mathbf{H}\mathbf{W}\mathbf{Q}^{(k)}\boldsymbol{\Theta}_V^{(k)} + \boldsymbol{b}_V^{(k)}\big), \end{aligned} \tag{3}$$

where $\mathbf{P}^{(0)} = \mathbf{X}$ and $\mathbf{W}$ is the identity matrix.

(3) Projection head. Chen et al. (2020) empirically demonstrate that including a non-linear transformation called projection head which maps representations to another latent space where contrastive loss is applied helps to improve the quality of representations. We also adopt two projection heads denoted by $g_\phi(\cdot)$ and $g_\psi(\cdot)$ for projecting node and hyperedge representations, respectively. Both projection heads in our method are implemented with a two-layer MLP and ELU activation (Clevert, Unterthiner, and Hochreiter 2016). Formally, $\mathbf{Z}_k := g_\phi(\mathbf{P}_k)$ and $\mathbf{Y}_k := g_\psi(\mathbf{Q}_k)$, where $k = 1, 2$ for two augmented views.

(4) Tri-directional contrastive loss. In TriCL framework, we employ three contrastive objectives: (a) **node-level contrast** aims to discriminate the representations of the same node in the two augmented views from other node representations, (b) **group-level contrast** tries to distinguish the representations of the same hyperedge in the two augmented views from other hyperedge representations, and (c) **membership-level contrast** seeks to differentiate a "real" node-hyperedge membership from a "fake" one across the two augmented views. We utilize the InfoNCE loss (Oord, Li, and Vinyals 2018), one of the popular contrastive losses, as in (Zhu et al. 2020, 2021b; Qiu et al. 2020).

In the rest of this subsection, we first provide a motivating example for the tri-directional contrastive loss. Then, we describe each of its three components in detail.

Motivating example. *How can the three forms of contrast be helpful for node representation learning?* In node classification tasks, for example, information about a group of nodes could help improve performance. Specifically, in co-authorship networks such as Cora-A and DBLP, nodes and hyperedges represent papers and authors, respectively, and papers written by the same author are more likely to belong to the same field and cover similar topics (i.e. homophily exists in hypergraphs (Veldt, Benson, and Kleinberg 2021)). Thus, high-quality author information could be useful in inferring the field of the papers he wrote, especially when information about the paper is insufficient.

Furthermore, leveraging node-hyperedge membership helps enrich the information of each node and hyperedge. For example, the fact that a meteorology paper is written by an author who studies mainly machine learning is a useful clue to suspect that (a) the paper is about application of machine learning techniques to meteorological problems and (b) the author is interested not only in machine learning but also in meteorology. In order to utilize such benefits explicitly, we proposed the tri-directional contrastive loss, which is described below.

Node-level contrast. For any node v_i, its representation from the first view, $\boldsymbol{z}_{1,i}$, is set to the anchor, the representation of it from the second view, $\boldsymbol{z}_{2,i}$, is treated as the positive sample, and the other representations from the second view, $\boldsymbol{z}_{2,k}$, where $k \neq i$, are regarded as negative samples. Let $s(\cdot,\cdot)$ denote the *score* function (a.k.a. *critic* function) that assigns high values to the positive pair, and low values to negative pairs (Tschannen et al. 2019). We use the cosine similarity as the score (i.e. $s(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}/\|\boldsymbol{u}\|\|\boldsymbol{v}\|$). Then the loss function for each positive node pair is defined as:

$$\ell_n(\boldsymbol{z}_{1,i}, \boldsymbol{z}_{2,i}) = -\log \frac{e^{s(\boldsymbol{z}_{1,i},\boldsymbol{z}_{2,i})/\tau_n}}{\sum_{k=1}^{|V|} e^{s(\boldsymbol{z}_{1,i},\boldsymbol{z}_{2,k})/\tau_n}},$$

where τ_n is a temperature parameter. In practice, we symmetrize this loss by setting the node representation of the second view as the anchor. The objective function for node-level contrast is the average over all positive pairs as:

$$\mathcal{L}_n = \frac{1}{2|V|}\sum_{i=1}^{|V|}\big\{\ell_n(\boldsymbol{z}_{1,i}, \boldsymbol{z}_{2,i}) + \ell_n(\boldsymbol{z}_{2,i}, \boldsymbol{z}_{1,i})\big\}. \quad (4)$$

Group-level contrast. For any hyperedge (i.e., a group of nodes) e_j, its representation from the first view, $\boldsymbol{y}_{1,j}$, is set to the anchor, the representation of it from the other view, $\boldsymbol{y}_{2,j}$, is treated as the positive sample, and the other representations from the view where the positive samples lie, $\boldsymbol{y}_{2,k}$, where $k \neq j$, are regarded as negative samples. We also use the cosine similarity as the critic, and then the loss function for each positive hyperedge pair is defined as:

$$\ell_g(\boldsymbol{y}_{1,j}, \boldsymbol{y}_{2,j}) = -\log \frac{e^{s(\boldsymbol{y}_{1,j},\boldsymbol{y}_{2,j})/\tau_g}}{\sum_{k=1}^{|E|} e^{s(\boldsymbol{y}_{1,j},\boldsymbol{y}_{2,k})/\tau_g}},$$

where τ_g is a temperature parameter. The objective function for group-level contrast is defined as:

$$\mathcal{L}_g = \frac{1}{2|E|}\sum_{j=1}^{|E|}\big\{\ell_g(\boldsymbol{y}_{1,j}, \boldsymbol{y}_{2,j}) + \ell_g(\boldsymbol{y}_{2,j}, \boldsymbol{y}_{1,j})\big\}. \quad (5)$$

Membership-level contrast. For any node v_i and hyperedge e_j that form membership (i.e., $v_i \in e_j$) in the original hypergraph, the node representation from the first view, $\boldsymbol{z}_{1,i}$, is set to the anchor, the hyperedge representation from the other view, $\boldsymbol{y}_{2,j}$, is treated as the positive sample. The negative samples are drawn from the representations of the other hyperedges that are not associated with node v_i, denoted by $\boldsymbol{y}_{2,k}$, where $k : i \notin k$. Symmetrically, $\boldsymbol{y}_{2,j}$ can also be the anchor, in which case the negative samples are $\boldsymbol{z}_{1,k}$, where $k : k \notin j$. To differentiate a "real" node-hyperedge membership from a "fake" one, we employ a *discriminator*, $\mathcal{D} : \mathbb{R}^{F'} \times \mathbb{R}^{F''} \to \mathbb{R}$ as the scoring function so that $\mathcal{D}(\boldsymbol{z}, \boldsymbol{y})$ represents the probability scores assigned to this node-hyperedge representation pair (should be higher for "real" pairs) (Hjelm et al. 2019; Veličković et al. 2018). For simplicity, we omit the augmented view number in the equation. Then we use the following objective:

$$\ell_m(\boldsymbol{z}_i, \boldsymbol{y}_j) = \underbrace{-\log \frac{e^{\mathcal{D}(\boldsymbol{z}_i,\boldsymbol{y}_j)/\tau_m}}{e^{\mathcal{D}(\boldsymbol{z}_i,\boldsymbol{y}_j)/\tau_m} + \sum_{k:i\notin k} e^{\mathcal{D}(\boldsymbol{z}_i,\boldsymbol{y}_k)/\tau_m}}}_{\text{when } \boldsymbol{z}_i \text{ is the anchor}} \underbrace{-\log \frac{e^{\mathcal{D}(\boldsymbol{z}_i,\boldsymbol{y}_j)/\tau_m}}{e^{\mathcal{D}(\boldsymbol{z}_i,\boldsymbol{y}_j)/\tau_m} + \sum_{k:k\notin j} e^{\mathcal{D}(\boldsymbol{z}_k,\boldsymbol{y}_j)/\tau_m}}}_{\text{when } \boldsymbol{y}_j \text{ is the anchor}},$$

where τ_m is a temperature parameter. From a practical point of view, considering a large number of negatives poses a prohibitive cost, especially for large graphs (Zhu et al.

2020; Thakoor et al. 2022). We, therefore, decide to randomly select a single negative sample per positive sample for $\ell_m(\boldsymbol{z}_i, \boldsymbol{y}_j)$. Since two views are symmetric, we get two node-hyperedge pairs for a single membership. The objective function for membership-level contrast is defined as:

$$\mathcal{L}_m = \frac{1}{2K} \sum_{i=1}^{|V|} \sum_{j=1}^{|E|} \mathbb{1}_{[h_{ij}=1]} \{\ell_m(\boldsymbol{z}_{1,i}, \boldsymbol{y}_{2,j}) + \ell_m(\boldsymbol{z}_{2,i}, \boldsymbol{y}_{1,j})\}. \tag{6}$$

Finally, by integrating Eq. (4), (5), and (6), our proposed contrastive loss is formulated as:

$$\mathcal{L} = \mathcal{L}_n + \omega_g \mathcal{L}_g + \omega_m \mathcal{L}_m, \tag{7}$$

where ω_g and ω_m are the weights of $\mathcal{L}_g$ and $\mathcal{L}_m$, respectively.

To sum up, TriCL jointly optimizes three contrastive objectives (i.e., node-, group-, and membership-level contrast), which enable the learned embeddings of nodes and hyperedges to preserve both the node- and group-level structural information at the same time.

4 Experiments

In this section, we empirically evaluate the quality of node representations learnt by TriCL on two hypergraph learning tasks: node classification and clustering, which have been commonly used to benchmark hypergraph learning algorithms (Zhou, Huang, and Schölkopf 2006).

4.1 Dataset

We assess the performance of TriCL on 10 commonly used benchmark datasets; these datasets are categorized into (1) co-citation datasets (Cora, Citeseer, and Pubmed) (Sen et al. 2008), (2) co-authorship datasets (Cora and DBLP (Rossi and Ahmed 2015)), (3) computer vision and graphics datasets (NTU2012 (Chen et al. 2003) and ModelNet40 (Wu et al. 2015)), and (4) datasets from the UCI Categorical Machine Learning Repository[1] (Zoo, 20Newsgroups, and Mushroom). Further descriptions and the statistics of datasets are provided in Appendix A. The experiment results for the fourth dataset type can be found in Appendix E.

4.2 Experimental Setup

Evaluation protocol. For the node classification task, we follow the standard linear-evaluation protocol as introduced in Veličković et al. (2018). The encoder is firstly trained in a fully unsupervised manner and computes node representations; then, a simple linear classifier is trained on top of these frozen representations through a ℓ_2-regularized logistic regression loss, without flowing any gradients back to the encoder. For all the datasets, we randomly split them, where 10%, 10%, and 80% of nodes are chosen for the training, validation, and test set, respectively, and evaluate the model with 20 dataset splits.

For the clustering task, we assess the quality of representations using the k-means clustering by operating it on the frozen node representations produced by each model. We employ the local Lloyd algorithm (Lloyd 1982) with the k-means++ seeding (Arthur and Vassilvitskii 2006) approach. For a fair comparison, we train each model with 5 random weight initializations, perform k-means 5 times on each trained encoder, and report the averaged results.

Baselines. We compare TriCL with various representative baseline approaches including 10 (semi-)supervised models and 4 unsupervised models. A detailed description of these baselines is provided in Appendix B. Note that, since the methods working on graphs can not be directly applied to hypergraphs, we use them after transforming hypergraphs to graphs via clique expansion. For all baselines, we report their performance based on their official implementations.

Implementation details. We employ a one-layer mean pooling hypergraph encoder described in (3) and PReLU (He et al. 2015) activation for non-linearlity. A further description of the model, optimizer, and model hyperparameters are provided in Appendix C.

4.3 Performance on Node Classification

Table 1 summarizes the empirical performance of all methods. Overall, our proposed method achieves the strongest performance across all datasets. In most cases, TriCL outperforms its unsupervised baselines by significant margins, and also outperforms the models trained with label supervision. Below, we make three notable observations.

First, applying graph contrastive learning methods, such as Node2vec, DGI, and GRACE, to hypergraph datasets is less effective. They show significantly lower accuracy compared to TriCL. This is because converting hypergraphs to graphs via clique expansion involves a loss of structural information. Especially, the Zoo dataset has large maximum and average hyperedge sizes (see Appendix A). When clique expansion is performed, a nearly complete graph, where most of the nodes are pairwisely connected to each other, is obtained, and thus most of the structural information is lost, resulting in significant performance degradation.

Second, rather than just using node-level contrast, considering the different types of contrast (i.e., group- and membership-level contrast) together can help improve performance. We propose and evaluate two model variants, denoted as TriCL-N and TriCL-NG, which use only node-level contrast and node- and group-level contrast, respectively, to validate the effect of each type of contrast. From Table 1, we note that the more types of contrast we use, the better the performance tends to be. To be more specific, we analyze the effectiveness of each type of contrast (i.e., $\mathcal{L}_n$, $\mathcal{L}_g$, and $\mathcal{L}_m$) on the node classification task in Table 2. We conduct experiments on all combinations of all types of contrast. The results show that using all types of contrast achieves the best performance in most cases as they are complementarily reinforcing each other (see Section 3.3 for motivating examples of how different types of contrast can be helpful for node representation learning). In most cases, using a combination of any two types of contrast is more powerful than using only one. It is noteworthy that while membership-level contrast causes model collapse[2] (especially for the Citeseer,

[1]https://archive.ics.uci.edu/ml/

[2]Model collapse (Zhu et al. 2021a) indicates that the model cannot

	Method	Cora-C	Citeseer	Pubmed	Cora-A	DBLP	NTU2012	ModelNet40	A.R.↓
Supervised	MLP	60.32 ± 1.5	62.06 ± 2.3	76.27 ± 1.1	64.05 ± 1.4	81.18 ± 0.2	65.17 ± 2.3	93.75 ± 0.6	14.4
	GCN*	77.11 ± 1.8	66.07 ± 2.4	82.63 ± 0.6	73.66 ± 1.3	87.58 ± 0.2	71.17 ± 2.4	91.67 ± 0.2	11.1
	GAT*	77.75 ± 2.1	67.62 ± 2.5	81.96 ± 0.7	74.52 ± 1.3	88.59 ± 0.1	70.94 ± 2.6	91.43 ± 0.3	10.1
	HGNN	77.50 ± 1.8	66.16 ± 2.3	83.52 ± 0.7	74.38 ± 1.2	88.32 ± 0.3	72.03 ± 2.4	92.23 ± 0.2	9
	HyperConv	76.19 ± 2.1	64.12 ± 2.6	83.42 ± 0.6	73.52 ± 1.0	88.83 ± 0.2	72.62 ± 2.6	91.84 ± 0.1	10.1
	HNHN	76.21 ± 1.7	67.28 ± 2.2	80.97 ± 0.9	74.88 ± 1.6	86.71 ± 1.2	71.45 ± 3.2	92.96 ± 0.2	10.3
	HyperGCN	64.11 ± 7.4	59.92 ± 9.6	78.40 ± 9.2	60.65 ± 9.2	76.59 ± 7.6	46.05 ± 3.9	69.23 ± 2.8	16.3
	HyperSAGE	64.98 ± 5.3	52.43 ± 9.4	79.49 ± 8.7	64.59 ± 4.3	79.63 ± 8.6	OOT	OOT	16
	UniGCN	77.91 ± 1.9	66.40 ± 1.9	84.08 ± 0.7	77.30 ± 1.4	90.31 ± 0.2	73.27 ± 2.7	94.62 ± 0.2	5.9
	AllSet	76.21 ± 1.7	67.83 ± 1.8	82.85 ± 0.9	76.94 ± 1.3	90.07 ± 0.3	75.09 ± 2.5	96.85 ± 0.2	6.3
Unsupervised	Node2vec*	70.99 ± 1.4	53.85 ± 1.9	78.75 ± 0.9	58.50 ± 2.1	72.09 ± 0.3	67.72 ± 2.1	84.94 ± 0.4	16
	DGI*	78.17 ± 1.4	68.81 ± 1.8	80.83 ± 0.6	76.94 ± 1.1	88.00 ± 0.2	72.01 ± 2.5	92.18 ± 0.2	8.3
	GRACE*	79.11 ± 1.7	68.65 ± 1.7	80.08 ± 0.7	76.59 ± 1.0	OOM	70.51 ± 2.4	90.68 ± 0.3	11
	S^2-HHGR	78.08 ± 1.7	68.21 ± 1.8	82.13 ± 0.6	78.15 ± 1.1	88.69 ± 0.2	73.95 ± 2.4	93.26 ± 0.2	6.6
	Random-Init	63.62 ± 3.1	60.44 ± 2.5	67.49 ± 2.2	66.27 ± 2.2	76.57 ± 0.6	74.39 ± 2.6	96.29 ± 0.3	12.9
	TriCL-N	80.23 ± 1.2	70.28 ± 1.5	83.44 ± 0.6	81.94 ± 1.1	90.88 ± 0.1	75.20 ± 2.6	97.01 ± 0.2	3.3
	TriCL-NG	81.45 ± 1.2	71.38 ± 1.2	83.68 ± 0.7	82.00 ± 1.0	90.94 ± 0.1	**75.25** ± **2.5**	97.02 ± 0.1	2
	TriCL	**81.57** ± **1.1**	**72.02** ± **1.2**	**84.26** ± **0.6**	**82.15** ± **0.9**	**91.12** ± **0.1**	75.23 ± 2.4	**97.08** ± **0.1**	**1.1**

Table 1: Node classification accuracy and standard deviations. Graph methods, marked as $\star$, are applied after converting hypergraphs to graphs via clique expansion. For each dataset, the best and the second-best performances are highlighted in boldface and underlined, respectively. A.R. denotes average rank, OOT denotes cases where results are not obtained within 24 hours, and OOM indicates out of memory on a 24GB GPU. In most cases, TriCL outperforms all others, including the supervised ones.

$\mathcal{L}_n$	$\mathcal{L}_g$	$\mathcal{L}_m$	Cora-C	Citeseer	Pubmed	Cora-A	DBLP	NTU2012	ModelNet40	A.R.↓
✓	-	-	80.23 ± 1.2	70.28 ± 1.5*	83.44 ± 0.6	81.94 ± 1.1	90.88 ± 0.1	75.20 ± 2.6	97.01 ± 0.2	4
-	✓	-	79.69 ± 1.6	71.02 ± 1.3*	80.20 ± 1.3	78.98 ± 1.4	88.60 ± 0.2	74.41 ± 2.6	96.66 ± 0.2	5.6
-	-	✓	76.76 ± 1.8	63.98 ± 2.0	79.86 ± 0.9	76.77 ± 1.1	63.95 ± 7.2	73.11 ± 2.8	96.57 ± 0.2	7
✓	✓	-	81.45 ± 1.2	71.38 ± 1.4	83.68 ± 0.7	82.00 ± 1.0	90.94 ± 0.1	**75.25** ± **2.5**	97.02 ± 0.1	2.3
✓	-	✓	80.49 ± 1.3	70.46 ± 1.5	83.98 ± 0.7	81.62 ± 1.0	90.75 ± 0.1	75.12 ± 2.5	97.03 ± 0.1	3.6
-	✓	✓	80.80 ± 1.1	71.73 ± 1.4	82.81 ± 0.7	80.24 ± 1.0	90.17 ± 0.1	73.76 ± 2.5	96.74 ± 0.1	4.4
✓	✓	✓	**81.57** ± **1.1**	**72.02** ± **1.4**	**84.26** ± **0.6**	**82.15** ± **0.9**	**91.12** ± **0.1**	75.23 ± 2.4	**97.08** ± **0.1**	**1.1**

Table 2: Comparison of node classification accuracy according to whether or not to use each type of contrast (i.e., $\mathcal{L}_n$, $\mathcal{L}_g$, and $\mathcal{L}_m$). Using all types of contrasts (i.e., node-, group-, and membership-level contrast) achieves the best performance in most cases as they are complementarily reinforcing each other.

DBLP, and Mushroom datasets) when used alone, it boosts performance when used with node- or group-level contrast.

Lastly, in Table 2, we note that group-level contrast is more crucial than node-level contrast for the Citeseer dataset (marked with asterisk), even though the downstream task is node-level. This result empirically supports our motivations.

To sum up, the superior performance of TriCL demonstrates that it produces highly generalized representations. More ablation studies and sensitivity analysis on hyperparameters used in TriCL are provided in Appendix E.

Robustness to the number of negatives. To analyze how the number of negative samples influences the node classification performance, we propose an approximation of TriCL's objective called TriCL-Sub. Here, instead of constructing the contrastive loss with all negatives, we randomly subsample k negatives across the hypergraph for node- and group-level contrast, respectively, at every gradient step. Our results in Table 3 show that TriCL is very robust to the number of negatives; even if only two negative samples are used for node- and group-level contrast, the performance degradation is less than 1%, still outperforming the best performing unsupervised baseline method, S^2-HHGR, by great margins. The results indicate that the random negative sampling is sufficiently effective for TriCL, and there is no need to select hard negatives, which incur additional costs.

4.4 Performance on Clustering

To show how well node representations trained with TriCL generalize across various downstream tasks, we evaluate the representations on the clustering task by k-means as described in Section 4.2. We use the node labels as ground truth for the clusters. To evaluate the clusters generated by k-means, we measure the agreement between the true labels and the cluster assignments by two metrics: Normalized Mutual Information (NMI) and pairwise F1 score. Table 4 summarizes the empirical performance. Our results show that TriCL achieves strong clustering performance in terms of all metrics across all datasets (1st place in terms of the aver-

significantly outperform or even underperform Random-Init.

Method	Cora-C	Citeseer	Pubmed	Cora-A	DBLP	NTU2012	ModelNet40	A.P.D.
S^2-HHGR all negatives	78.08 $\pm$ 1.7	68.21 $\pm$ 1.8	82.13 $\pm$ 0.6	78.15 $\pm$ 1.1	88.69 $\pm$ 0.2	73.95 $\pm$ 2.4	93.26 $\pm$ 0.2	-
TriCL-Sub ($k = 2$)	80.62 $\pm$ 1.3	71.95 $\pm$ 1.3	83.22 $\pm$ 0.7	81.25 $\pm$ 1.0	90.66 $\pm$ 0.2	74.95 $\pm$ 2.6	97.02 $\pm$ 0.1	0.65%
TriCL-Sub ($k = 4$)	81.15 $\pm$ 1.2	**72.24 $\pm$ 1.2**	83.91 $\pm$ 0.7	81.85 $\pm$ 0.9	90.83 $\pm$ 0.1	75.02 $\pm$ 2.6	97.05 $\pm$ 0.1	0.23%
TriCL-Sub ($k = 8$)	81.32 $\pm$ 1.2	72.04 $\pm$ 1.3	83.88 $\pm$ 0.7	82.05 $\pm$ 0.9	90.93 $\pm$ 0.1	75.09 $\pm$ 2.5	97.05 $\pm$ 0.1	0.18%
TriCL-Sub ($k = 16$)	81.49 $\pm$ 1.1	72.02 $\pm$ 1.2	84.23 $\pm$ 0.7	82.10 $\pm$ 0.9	90.97 $\pm$ 0.1	75.16 $\pm$ 2.5	97.07 $\pm$ 0.1	0.07%
TriCL all negatives	**81.57 $\pm$ 1.1**	72.02 $\pm$ 1.2	**84.26 $\pm$ 0.6**	**82.15 $\pm$ 0.9**	**91.12 $\pm$ 0.1**	**75.23 $\pm$ 2.4**	**97.08 $\pm$ 0.1**	-

Table 3: TriCL is very robust to the number of negative samples (i.e., k). A.P.D. stands for average performance degradation. Even if only two negative samples are used at every gradient step, the performance degrades by less than 1%.

Method	Cora-C		Citeseer		Pubmed		Cora-A		DBLP		NTU2012		ModelNet40		A.R.$\downarrow$
	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	NMI$\uparrow$	F1$\uparrow$	
features	20.0	28.8	21.5	36.1	19.5	**53.4**	17.2	29.2	37.0	47.3	81.7	69.0	90.6	86.5	4.3
Node2vec*	39.1	44.5	24.5	38.5	23.1	40.1	16.0	34.1	32.4	37.8	78.3	57.7	72.9	53.1	5.0
DGI*	**54.8**	60.1	40.1	51.7	**30.4**	53.0	45.2	52.5	58.0	57.7	79.6	61.7	85.0	73.7	2.9
GRACE*	44.4	45.6	33.3	45.7	16.7	41.9	37.9	43.3	16.7	41.9	74.6	47.5	79.4	59.9	4.7
S^2-HHGR	51.0	56.8	41.1	53.1	27.7	53.2	45.4	52.3	60.3	62.7	82.7	71.2	91.0	90.6	2.5
TriCL	54.5	**60.6**	**44.1**	**57.4**	30.0	51.7	**49.8**	**56.7**	**63.1**	**63.0**	**83.2**	**71.5**	**95.7**	**94.7**	**1.7**

Table 4: Evaluation of the embeddings learned by unsupervised methods using k-means clustering. As a naïve baseline method, expressed by 'features', we only use the node features as an input of k-means. All metrics are normalized by multiplying 100. Larger NMI and F1 indicate better performance, and A.R. denotes average ranking. TriCL ranks first in clustering performance.

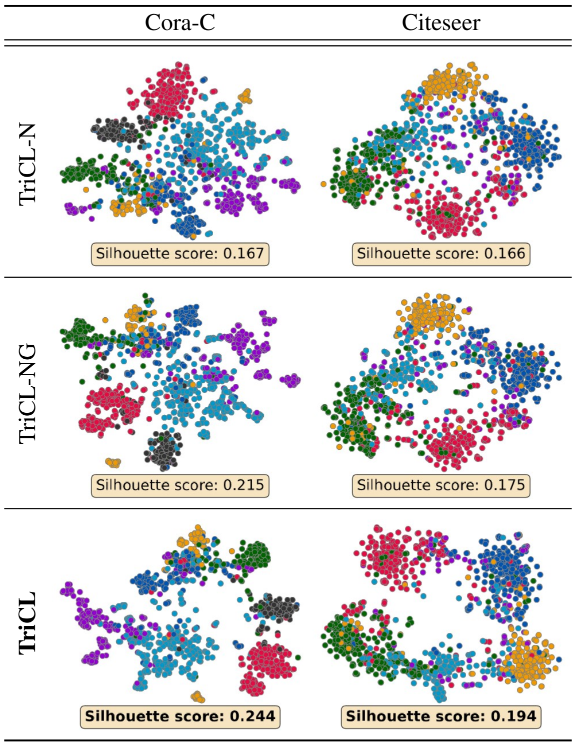

Figure 2: t-SNE plots of the node representations produced by TriCL and its two variants. The node embeddings of TriCL exhibits the most distinct clusters with the help of group and membership contrast, as measured numerically by the Silhouette score (the higher, the better).

age rank). This is because the node embeddings learned by TriCL simultaneously preserve local and community structural information by fully utilizing group-level contrast.

4.5 Qualitative Analysis

To represent and compare the quality of embeddings intuitively, Figure 2 shows t-SNE (Van der Maaten and Hinton 2008) plots of the node embeddings produced by TriCL and its two variants (i.e., TriCL-N and TriCL-NG) on the Citeseer and Cora Co-citation dataset. As expected from the quantitative results, the 2-D projection of embeddings learned by TriCL shows visually and numerically (based on the Silhouette score (Rousseeuw 1987)) more distinguishable clusters than those obtained by its two variants. In Appendix F, we give additional qualitative analysis.

5 Conclusion

In this paper, we proposed TriCL, a novel hypergraph contrastive representation learning approach. We summarize our contributions as follows:

- We proposed the use of tri-directional contrast, which is a combination of node-, group-, and membership-level contrast, that consistently and substantially improves the quality of the learned embeddings.
- We achieved state-of-the-art results in node classification on hypergraphs by using tri-directional contrast together with our data augmentation schemes. Moreover, we verified the surprising effectiveness of uniform negative sampling for our use cases.
- We demonstrated the superiority of TriCL with extensive experiments using 14 baseline approaches, 10 datasets (including three used in Appendix E), and two tasks.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd. and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00157, Robust, Fair, Extensible Data-Centric Continual Learning) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

References

Arthur, D.; and Vassilvitskii, S. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Arya, D.; Gupta, D. K.; Rudinac, S.; and Worring, M. 2020. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv:2010.04558*.

Bai, S.; Zhang, F.; and Torr, P. H. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110: 107637.

Benson, A. R.; Abebe, R.; Schaub, M. T.; Jadbabaie, A.; and Kleinberg, J. 2018. Simplicial closure and higher-order link prediction. *PNAS*, 115(48).

Benson, A. R.; Gleich, D. F.; and Leskovec, J. 2016. Higher-order organization of complex networks. *Science*, 353(6295): 163–166.

Chen, D.-Y.; Tian, X.-P.; Shen, Y.-T.; and Ouhyoung, M. 2003. On visual similarity based 3D model retrieval. *Computer graphics forum*, 22(3): 223–232.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, 1597–1607.

Chien, E.; Pan, C.; Peng, J.; and Milenkovic, O. 2022. You are AllSet: A Multiset Function Framework for Hypergraph Neural Networks. In *ICLR*.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*.

Do, M. T.; Yoon, S.-e.; Hooi, B.; and Shin, K. 2020. Structural patterns and generative models of real-world hypergraphs. In *KDD*, 176–186.

Dong, Y.; Sawin, W.; and Bengio, Y. 2020. HNHN: Hypergraph networks with hyperedge neurons. In *ICML Workshop on Graph Representation Learning and Beyond*.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *AAAI*, volume 33, 3558–3565.

Gao, T.; Yao, X.; and Chen, D. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *ICML*, 4116–4126.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 9729–9738.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 1026–1034.

Hein, M.; Setzer, S.; Jost, L.; and Rangapuram, S. S. 2013. The total variation on hypergraphs-learning on hypergraphs revisited. In *NIPS*.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

Huang, J.; and Yang, J. 2021. Unignn: a unified framework for graph and hypergraph neural networks. In *IJCAI*.

Jaiswal, A.; Babu, A. R.; Zadeh, M. Z.; Banerjee, D.; and Makedon, F. 2020. A survey on contrastive self-supervised learning. *Technologies*, 9(1): 2.

Lee, G.; Choe, M.; and Shin, K. 2022. HashNWalk: Hash and Random Walk Based Anomaly Detection in Hyperedge Streams. In *IJCAI*.

Lee, G.; Ko, J.; and Shin, K. 2020. Hypergraph motifs: concepts, algorithms, and discoveries. In *Proceedings of the VLDB Endowment*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 3744–3753.

Li, B.; Jing, B.; and Tong, H. 2022. Graph Communal Contrastive Learning. In *WWW*, 1203–1213.

Li, P.; and Milenkovic, O. 2018. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. In *ICML*, 3014–3023.

Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; and Tang, J. 2021. Self-supervised learning: Generative or contrastive. *TKDE*.

Liu, Y.; Jin, M.; Pan, S.; Zhou, C.; Zheng, Y.; Xia, F.; and Yu, P. 2022. Graph self-supervised learning: A survey. *TKDE*.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv:1807.03748*.

Peng, Z.; Huang, W.; Luo, M.; Zheng, Q.; Rong, Y.; Xu, T.; and Huang, J. 2020. Graph representation learning via graphical mutual information maximization. In *WWW*, 259–270.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*, 1150–1160.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*.

Rossi, R.; and Ahmed, N. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI*.

Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20: 53–65.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Thakoor, S.; Tallec, C.; Azar, M. G.; Azabou, M.; Dyer, E. L.; Munos, R.; Veličković, P.; and Valko, M. 2022. Large-scale representation learning on graphs via bootstrapping. In *ICLR*.

Tschannen, M.; Djolonga, J.; Rubenstein, P. K.; Gelly, S.; and Lucic, M. 2019. On Mutual Information Maximization for Representation Learning. In *ICLR*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11).

Veldt, N.; Benson, A. R.; and Kleinberg, J. 2021. Higher-order homophily is combinatorially impossible. *arXiv:2103.11818*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep Graph Infomax. In *ICLR*.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 1912–1920.

Xia, X.; Yin, H.; Yu, J.; Wang, Q.; Cui, L.; and Zhang, X. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI*, volume 35, 4503–4511.

Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; and Talukdar, P. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *NeurIPS*.

Yang, D.; Qu, B.; Yang, J.; and Cudre-Mauroux, P. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *WWW*, 2147–2157.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. In *NeurIPS*.

Yu, J.; Yin, H.; Li, J.; Wang, Q.; Hung, N. Q. V.; and Zhang, X. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *WWW*, 413–424.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. In *NIPS*.

Zhang, J.; Gao, M.; Yu, J.; Guo, L.; Li, J.; and Yin, H. 2021. Double-Scale Self-Supervised Hypergraph Learning for Group Recommendation. In *CIKM*, 2557–2567.

Zhang, J.; Li, F.; Xiao, X.; Xu, T.; Rong, Y.; Huang, J.; and Bian, Y. 2022. Hypergraph Convolutional Networks via Equivalency between Hypergraphs and Undirected Graphs. *arXiv:2203.16939*.

Zheng, X.; Zhu, W.; Tang, C.; and Wang, M. 2019. Gene selection for microarray data classification via adaptive hypergraph embedded dictionary learning. *Gene*, 706: 188–200.

Zhou, D.; Huang, J.; and Schölkopf, B. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*.

Zhu, R.; Zhao, B.; Liu, J.; Sun, Z.; and Chen, C. W. 2021a. Improving contrastive learning by visualizing feature transformation. In *ICCV*, 10306–10315.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020. Deep graph contrastive representation learning. In *ICML Workshop on Graph Representation Learning and Beyond*.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021b. Graph contrastive learning with adaptive augmentation. In *WWW*, 2069–2080.