Self-supervised Hypergraph Representation Learning

1st Boxin Du
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
boxindu2@illinois.edu

2nd Changhe Yuan
Amazon
USA
ychanghe@amazon.com

3rd Robert Barton
Immunai
USA
rab375@cornell.edu

4th Tal Neiman
Meta
USA
tal.neiman2@gmail.com

5th Hanghang Tong
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA
htong@illinois.edu

***Abstract*— Despite the prevalence of hypergraphs in a variety of high-impact applications, there are relatively few works on hypergraph representation learning, most of which primarily focus on hyperlink prediction, and are often restricted to the transductive learning setting. Among others, a major hurdle for effective hypergraph representation learning lies in the label scarcity of nodes and/or hyperedges. To address this issue, this paper presents an end-to-end, bi-level pre-training strategy with Graph Neural Networks for hypergraphs. The proposed framework named HyperGRL bears three distinctive advantages. First, it is mainly designed in the self-supervised fashion which has broad applicability, and meanwhile it is also capable of ingesting the labeling information when available. Second, at the heart of the proposed HyperGRL are two carefully designed pretexts, one on the node level and the other on the hyperedge level, which enable us to encode both the local and the global context in a mutually complementary way. Third, the proposed framework can work in both transductive and inductive settings. When applying the two proposed pretexts in tandem, it can accelerate the adaptation of the knowledge from the pre-trained model to downstream applications in the transductive setting, thanks to the bi-level nature of the proposed method. Extensive experiments demonstrate that: (1) HyperGRL achieves up to 5.69% improvements in hyperedge classification, and (2) improves pre-training efficiency by up to 42.80% on average[1].**

I. Introduction

Hypergraph, as a generalization of the traditional graph data, is ubiquitous in various domains, and has drawn increasing attention recently [1]–[3]. Different from traditional graphs, which consist of nodes and edges to represent *pairwise* relations between nodes, each hyperedge contains a collection of nodes, which represents a *high-order* relation. For example, in the clinical studies of the pharmacological mechanism [1], [4], the effects of medical treatment is often the result of the combined interactions of a set of drugs. Here the combination of drugs for one disease forms one hyperedge. In the bioinformatics research, protein/multi-protein complexes, which consist of different collections of protein molecules, display different functions [5]. Here a collection of protein molecules consists of one hyperedge. In the social network domain, a group of users who participate in the same event could form a hyperedge of that event [6], [7]. In the academic domain, authors of the same paper could be a hyperedge of the paper they jointly publish. Compared with the traditional high-order analysis on simple graphs, and the meta-path/meta-structure analysis on heterogeneous graphs, hypergraph provides a more suitable graph model for mining both the directed or undirected high-order relations, on which traditional methods usually fall short.

Representation learning on hypergraph offers a promising way to streamline various hypergraph applications. However, the traditional graph representation learning methods are not directly applicable in capturing the *high-order* relations of the hypergraphs. To date, relatively few works on hypergraph representation learning exist, most of which focus on hyperlink prediction [1], [2], [7], [8]. The major difficulties of representation learning on hypergraph are two-fold. First, the labels of diverse downstream tasks are usually very scarce, which makes it difficult for training the downstream neural models. Second, most of the recent hypergraph representation learning methods only work in the transductive learning setting [1], [3], [7], [9]–[11]. Specifically, these methods require all data to be seen during the training for feature generation or representation learning in the model, which renders the inability of these methods to handle unseen data. For example, in the hyperlink prediction problem, many existing methods require that all candidate hyperlinks to be seen during training.

In this paper, inspired by the recent advances of pre-training strategies developed in Natural Language Processing (NLP) community [12] and Graph Neural Networks (GNNs) research [13]–[15], we propose HyperGRL, a self-supervised pre-training based hypergraph representation learning framework, with the target of both inductive and transductive hyperedge classification. Compared with existing methods, the proposed HyperGRL enjoys the following three distinctive advantages. First, the proposed pre-training framework is capable of lever-

[1]The source code link: https://github.com/boxindu/HyperGRL

aging labeled data (with supervised pre-training) as well as unlabeled data (with self-supervised pre-training), to learn transferable knowledge for diverse downstream tasks without the help of extra domain-specific hypergraph datasets [13], [16]. Second, our method explores bi-level (i.e. node-level and hyperedge-level) self-supervised pretext tasks, which aim at capturing the intrinsic *high-order* relationships of nodes and hyperedges respectively. Third, the proposed HyperGRL can work in both transductive and inductive settings. The pre-training strategy proposed for the transductive setting is adaption-aware, in the sense that the pre-trained model could be more adaptive to the downstream tasks compared to traditional pre-training methods, and meanwhile be more computationally efficient.

The main contributions of the paper are as follows.

- **Novel Pre-training Framework**. We propose a bi-level pre-training framework for hypergraph representation learning named HyperGRL, equipped with two mutually complementary self-supervised pretext tasks. The proposed framework can be applied to both transductive and inductive settings. For the tranductive setting, the proposed HyperGRL further embraces an adaptation-aware pre-training strategy to accelerate the knowledge transfer.
- **Extensive Empirical Evaluations.** We perform extensive experiments to demonstrate the efficacy of HyperGRL. In particular, the proposed HyperGRL (1) outperforms *all* baselines across all datasets for inductive hyperedge classification, with an up to 5.69% improvement over the best competitor, and (2) improves pre-training efficiency by up to 42.8% on average. In addition, the proposed HyperGRL has been successfully applied to a real-world e-commerce application, namely inconsistent variation family (IVF) classification, outperforming the current models therein.

The rest of the paper is organized as follows. Section II formally describes the problem studied by the paper. Section III presents the proposed model. Section IV shows the experiments on public datasets and the real-world case study is shown in Section V. Related work is introduced in Section VI. The paper is concluded in Section VII.

II. Problem Definition

The main notations used in this paper are summarized in Table I. We first define the hypergraphs as follows.

TABLE I: Symbols and Definition

Symbols	Definition
$G = (\mathcal{V}, \mathcal{E}, \mathbf{F}^{(n)})$	a hypergraph of node set $\mathcal{V}$, edge set $\mathcal{E}$, and feature $\mathbf{F}^{(n)}$
$\mathbf{M}$	the hypergraph incidence matrix
$\mathbf{A}$	the adjacency matrix of nodes inferred from $\mathbf{M}$
Θ, Ω_i	parameters of GNN module and adjustment modules
$f_\Theta(\cdot)$	GNN module with parameter Θ
$g_{\Omega_i}(\cdot)$	neural adjustment module with parameter Ω_i
$f_{\Theta:\Theta_0=\Theta'}(\cdot)$	a pre-trained GNN module with initialization Θ'

Definition 1: **Hypergraph:** A hypergraph is represented by $G = (\mathcal{V}, \mathcal{E}, \mathbf{F}^{(n)})$, in which $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of n nodes and $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ is a set of m hyperedges. $e_i = \{v_j^{(i)}\}, 1 \leq j \leq n$ represents the i-th hyperedge in which the nodes $v_j^{(i)} \in \mathcal{V}$. We say that node v_j is *inside* hyperedge e_i. $\mathbf{F}^{(n)}$ is the feature matrix[2] for nodes.

A hypergraph incidence matrix [11] $\mathbf{M} \in \mathbb{R}^{n \times m}$ is defined such that $\mathbf{M}(i, j) = 1$ if node i appears in hyperedge j, and $\mathbf{M}(i, j) = 0$ otherwise. From $\mathbf{M}$, we can build an adjacency matrix $\mathbf{A} = \mathbf{M}^\intercal \mathbf{M}$, in which $\mathbf{A}(i, j)$ indicates the number of nodes that appear in both hyperedge i and hyperedge j.

Before formally defining the inductive hyperedge classification problem, we provide a brief review of Graph Neural Networks.

Preliminaries on Graph Neural Networks. GNNs are powerful deep learning models on graphs. Representative models include Graph Convolutional Networks (GCN) [17], Graph Isomorphism Networks (GIN) [18], Graph Attention Networks (GAT) [19], etc. The intuition behind many existing GNN models is to learn the node representation by convolutionally aggregating both the node/edge features and the features of the node's local neighbors through neural networks. Message passing is often adopted as a popular choice to design various GNN models [20]. There are two main steps in the message passing process, including message passing and message updating. In the message passing step, the node features are passed to its neighbors. In the message updating step, the received features are passed through an aggregation function (e.g., a neural network) for node representations. Typical message passing can be summarized as:

$$\mathbf{h}_{m_v}^{(t+1)} = P_t(\{\mathbf{h}_v^{(t)}, \mathbf{h}_w^{(t)}, \mathbf{e}_{vw}\}), \forall w \in \mathcal{N}(v) \tag{1}$$

$$\mathbf{h}_v^{(t+1)} = U_t(\mathbf{h}_v^{(t)}, \mathbf{h}_{m_v}^{(t+1)}) \tag{2}$$

where P_t and U_t are the message passing function and node representation updating function of the $t-$th iteration respectively. $\mathbf{h}_v, \mathbf{h}_w$ are node representations of neighboring nodes (v, w), and are initialized as node features. $\mathbf{e}_{vw}$ is the feature of the edge between node v and node w. Different GNN models differ in the functions $P_t()$ and/or $U_t()$. For example, GCN [17] takes the summation of the neighboring nodes in the message passing step and attaches a neural network module on the passed message and the node feature itself for feature aggregation. By using GNN as a neural function $f_\Theta(\cdot)$ for node representations, the inductive hyperedge classification problem is defined as follows.

Definition 2: **Inductive Hyperedge classification:** Given a set of hyperedges $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ which are not seen in the training stage, the goal of GNN model $f_\Theta(\cdot)$ is to learn embeddings for the downstream classifier[3] $g_\Omega(\cdot)$ to classify them into t categories. $g_\Omega(f_\Theta(e_i)) = \mathbf{p}_i, i \in \{1, 2, ..., m\}$, where $\mathbf{p}_i$ is the prediction vector for e_i with a non-zero entry indicating e_i's predicted category.

By adopting pre-training strategy, model $f_\Theta(\cdot)$ is first trained on pretext task(s). Note that there could be more than one pretext task. The fine-tuning module can be represented as

[2]Optionally, there might be a feature matrix for hyperedges $\mathbf{F}^{(h)}$.

[3]$g_\Omega()$ is also known as a neural adjustment module, which is an MLP specified for pretext tasks with parameter Ω for mapping node representations to the predicted labels/values.

$f_{\Theta:\Theta_0=\Theta'}(\cdot)$ given the pre-trained GNN module $f_{\Theta'}(\cdot)$. Generally speaking, pre-training $f_{\Theta'}(\cdot)$ could be either supervised if the labels for the pretext task are available, or unsupervised, such as self-supervised methods.

III. Proposed Pre-training Framework

We first present the challenges and key ideas in Section III-A. The pre-training stage, including the proposed bi-level self-supervised pretext tasks as well as an adaptation-aware pre-training strategy, is presented in Section III-B, Section III-C and Section III-D. The overall HyperGRL framework architecture is elaborated in Section III-E.

A. Challenges and Key Ideas

The first challenge for pre-training hypergraphs is how to design the self-supervised pretext tasks, since the *high-order* node relations of hypergraphs are significantly different from traditional graphs structurally. Our idea is to incorporate both node-level and hyperedge-level pretext tasks, which aim at capturing both local and global contextual patterns of hypergraphs. Locally, the node inside one specific hyperedge should be distinguished from nodes outside this hyperedge given the context of node. For one specific hyperedge and a given inside node, we define the context of the node as all other nodes inside the hyperedge as shown in Figure 1. Globally, the similarities between hyperedges ought to be preserved. However, calculating pairwise hyperedge similarities itself is challenging and costly, with at least $O(m^2)$ time complexity for calculating every pair of hyperedges if the number of hyperedges is m. As an approximation for learning pairwise hyperedge similarities, our idea is to first cluster the hyperedges, based on the features of nodes inside hyperedges or the hyperedge adjacency matrix when available, and then to preserve the membership characteristic of the hyperedge clusters.

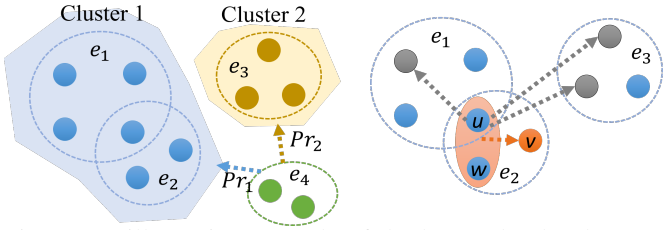

Fig. 1: An illustrative example of the hyperedge-level pretext task (left), and the node-level pretext task (right). Pr_1, Pr_2 are probabilities for assigning e_4 to cluster 1 and cluster 2. The red area on the right subfigure shows the context of node v, and the gray nodes are the sampled negative examples of node v and its context. Best viewed in color.

The second challenge is how to mitigate the divergence between the self-supervised pretext tasks and the downstream tasks. Even with two mutually complementary pretext tasks, such divergence might still exist, in the sense that a well-trained pre-trained model might be overly fit on the pretext tasks, and could not optimally generalize to the downstream tasks. In the transductive setting, our idea is to design an adaptation-aware pre-training strategy, which targets at learning a well-adaptive pre-trained model for downstream tasks. In this strategy, only one self-supervised pretext task (node-level) is fully trained until convergence, and the other self-supervised pretext task (hyperedge-level) is applied on the unlabeled data for fast adaptation.

B. Node-level Self-supervised Pretext Task

Task Description. In this pretext task, we aim at predicting the relationship between a given node and its hyperedge context (i.e., other nodes inside the hyperedge). Intuitively, we expect the node and the context share similar representation if they belong to the same hyperedge.

Specifically, in order to obtain node-level self-supervised training labels, we first uniformly sample a seed node (e.g. $v_i^{(s)}$) inside each hyperedge, and obtain its corresponding context (e.g., C_i). The combined node-context pair $(v_i^{(s)}, C_i)$ is a positive example. Next, for each pair $(v_i^{(s)}, C_i)$, we adopt a negative sampling method for negative examples ($v_{ij}^{(n)} \sim Pr_{ij}$, the negative sampling probability) of the selected context. We utilize a GNN module inside the *clique-expansion* of hyperedges for the hidden representation of nodes. The representation of nodes corresponding to contexts are aggregated by a pooling layer for the context representations. The node-context relationship is learned via a binary cross-entropy objective. For notation simplicity, let $\mathbf{h}$ be the representation learned by GNN module, and $\hat{\mathbf{h}} := g_{\Omega_n}^{(n)}(\mathbf{h})$ be the node representation after applying the neural adjustment function $g_{\Omega_n}^{(n)}(\cdot)$ of the node-level pretext task.

$$\mathcal{L}^{(n)} = \sum_i \sum_j log[1-\sigma((\hat{\mathbf{h}}_i^{(s)})^\intercal \hat{\mathbf{h}}_i^{(C)})] + log[\sigma((\hat{\mathbf{h}}_{ij}^{(n)})^\intercal \hat{\mathbf{h}}_i^{(C)})] \tag{3}$$

where $\hat{\mathbf{h}}_i^{(s)}, \hat{\mathbf{h}}_i^{(C)}, \hat{\mathbf{h}}_{ij}^{(n)}$ are the hidden representation of seed node i, context of node i, and the negative sample j of node i after using the adjustment function, respectively. $\sigma(\cdot)$ is a sigmoid function. $(\hat{\mathbf{h}}_i^{(s)})^\intercal \hat{\mathbf{h}}_i^{(C)}$ is expected to be larger than $(\hat{\mathbf{h}}_{ij}^{(n)})^\intercal \hat{\mathbf{h}}_i^{(C)}$ since the positive seed node-context pair share similar feature distributions.

Negative Sampling Method. For a given pair of node and its corresponding context in one hyperedge, one naive negative sampling method is to uniformly sample nodes outside this hyperedge. This is applicable even if all hyperedges of a hypergraph do not share nodes, which is often the case in real-world applications (see the Case Study in Section V). However, this method does not distinguish between structurally close-by and distant nodes/hyperedges, especially when the relations of nodes from different hyperedges can be obtained via the adjacency matrix $\mathbf{A}$ from the incidence matrix $\mathbf{M}$. Structurally close nodes tend to have very similar features or even the same labels, and should be avoided as negative samples. We use the following negative sampling strategy such that the structurally close nodes would have exponentially lower probabilities to be sampled for negative nodes selection. Let $\tilde{\mathbf{A}} = \mathbf{A}^k$, where the entries of $\tilde{\mathbf{A}}$ give the number of paths of length k. We normalized $\tilde{\mathbf{A}}$ as $\hat{\mathbf{A}} = \mathbf{D}^{-1}\tilde{\mathbf{A}}$, where $\mathbf{D}$ is a diagonal degree matrix of $\tilde{\mathbf{A}}$, to prevent extremely low probabilities. For a

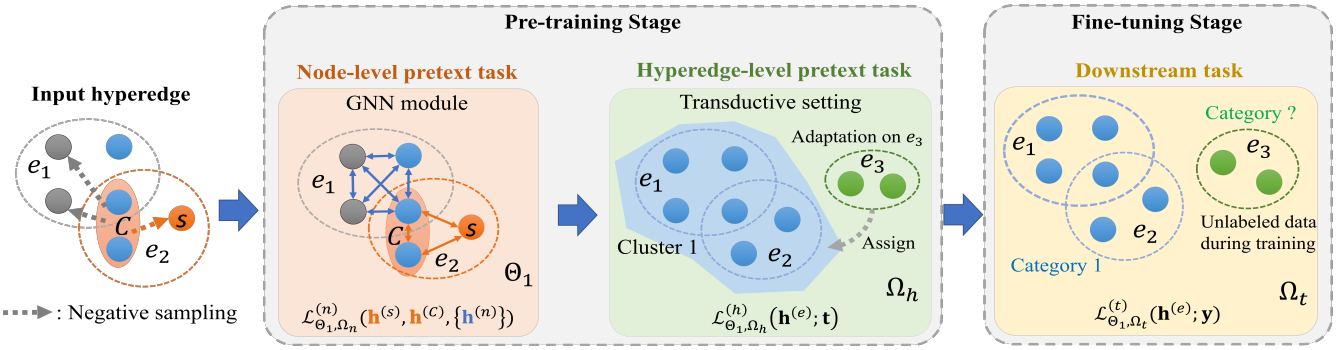

Fig. 2: The pre-training and fine-tuning framework with node-level and hyperedge-level self-supervised pre-training/adaptation in HyperGRL for hyperedge classification. Best viewed in color.

given node i, the probability of sampling node $j \neq i$ is given as follows.

$$Pr_{ij} = \frac{exp(-\gamma \cdot \hat{\mathbf{A}}_{ij})}{\sum_j exp(-\gamma \cdot \hat{\mathbf{A}}_{ij})} \tag{4}$$

in which $\gamma > 0$ is a scaling scalar for further tuning the sampling probabilities. We consider the nodes which are not reachable by paths of length k having high and equal probability of being sampled. In practice, we find that a small k is often sufficient for good performance, which also helps with the efficiency for calculating and storing $\mathbf{A}^k$. This negative sampling method is referred to as the exponential sampling method. The positive/negative hyperedges represent the hyperedges from which positive/negative nodes are extracted in the rest of the paper.

C. Hyperedge-level Self-supervised Pretext Task

In this pretext task, different from the local node-level pretext task, we strive to capture the more global patterns of hypergraphs. As discussed in Section III-A, we aim at predicting hyperedges' cluster membership information. The clustering is conducted on the graph of hyperedges as follows. First, the graph of hyperedges is constructed with adjacency matrix $\mathbf{A} = \mathbf{M}^\top\mathbf{M}$, such that $\mathbf{A}(i,j)$ is the number of nodes that exist in both hyperedges i and j. Next, the METIS algorithm [21] is applied to partition the graph of hyperedges into q clusters. q is empirically selected, and is set to be larger than or equal to the number of categories of hyperedges in the empirical experiments (see Section IV). The GNN module, which shares the parameters with node-level pretext task, is applied inside the *clique expansion* of the hyperedges, and the representations for hyperedges are obtained by a graph pooling layer. Then, by adopting the categorical cross-entropy loss, the hyperedge-level self-supervised task is written as:

$$\mathcal{L}^{(h)} = -\sum_i [\log(\text{softmax}(g^{(h)}_{\Omega_h}(\mathbf{h}^{(e)}_i))) \circ \mathbf{y}^{(h)}_i]^\top \mathbf{1} \tag{5}$$

where $\mathbf{h}^{(e)}_i$ is the hidden representation of the hyperedge i, $g^{(h)}_{\Omega_h}(\cdot)$ is a neural adjustment function to map the hyperedge representations to q-dimensional vector. $\mathbf{y}^{(h)}_i$ is the multi-class label vector of hyperedge i indicating the cluster membership, and $\mathbf{1}$ is an all-one vector, for taking the summation of the $\log(\text{softmax}(\cdot))$ scores of the correct categories.

D. Adaptation-aware Pre-training Strategy

Traditional pre-training methods use a two-stage procedure, in which the first stage trains the pretext task until convergence with self-labels, and the second stage trains the downstream task with the task labels. Specifically in our scenario with two self-supervised pretext tasks, the two-stage training can be realized in a serial procedure as follows.

$$\Theta' = \text{argmin}_\Theta \mathcal{L}^{(n)}(g_{\Omega_n}(f_\Theta(\mathcal{E}_{train}, \mathbf{F}^{(n)})); \mathbf{y}^{(n)}) \tag{6a}$$

$$\Theta^{(pre)} = \text{argmin}_\Theta \mathcal{L}^{(h)}(g_{\Omega_h}(f_{\Theta:\Theta_0=\Theta'}(\mathcal{E}_{train}, \mathbf{F}^{(n)})); \mathbf{y}^{(h)}) \tag{6b}$$

$$\hat{\Theta} = \text{argmin}_\Theta \mathcal{L}^{(t)}(g_{\Omega_t}(f_{\Theta:\Theta_0=\Theta^{(pre)}}(\mathcal{E}_{train}, \mathbf{F}^{(n)})); \mathbf{y}^{(t)}) \tag{6c}$$

where $\mathcal{L}^{(t)}$ and $\mathbf{y}^{(t)}$ are the loss function and labels for the downstream task respectively. g_{Ω_t} is the neural adjustment module for downstream task. The parameters of GNN module is first obtained by training the node-level pretext task and then by training the hyperedge-level pretext task.

As discussed in Section III-A, the above strategy in the transductive setting might bring non-negligible divergence between pre-training and downstream task training, which would lead to a sub-optimal model. Moreover, since the hyperedge-level pretext task approximately preserves the pairwise hyperedge distances, pre-training this task till convergence might result in an even larger divergence. To address this issue, we propose an adaptation-aware pre-training strategy. The key idea is only performing node-level pre-training on the hyperedges with downstream task labels, and utilizing the hyperedge-level pretext task as adaptation on the hyperedges whose labels are to be predicted in the transductive setting.

Specifically, we maintain Eq. (6a). But instead of training the hyperedge-level pretext task until convergence (Eq. (6b)), we use it as an adaptation task, which can be represented as s steps of gradient descent on test hyperedges $\mathcal{E}_{test}$, and Eq. (6b) is replaced by s steps of:

$$\Theta := \Theta - \epsilon \cdot \frac{\partial \mathcal{L}^{(h)}(g_{\Omega_h}(f_{\Theta:\Theta_0=\Theta'}(\mathcal{E}_{test}, \mathbf{F}^{(n)})); \mathbf{y}^{(h)})}{\partial \Theta} \tag{7}$$

where ϵ is the learning rate. Compared with the traditional pre-training method, the advantages of the adaptation-aware training strategy are two-fold. First, the pre-trained model for transferring general data knowledge is more adaptive to the

downstream tasks. Second, it is a more efficient pre-training strategy, because only one pretext task needs to be fully trained, and the adaptation, which only requires a few steps, can be conducted whenever the test data[4] for the downstream task is available (e.g., in an online system).

E. Proposed HyperGRL Framework

Overall Framework. The end-to-end framework architecture is illustrated in Figure 2. Negative sampling is first conducted on the input hyperedge set for node and context pairs. In the core of the framework is the bi-level pre-training stage.

In the node-level pretext task, a GNN module is adopted for the positive and negative hyperedges from which seed node/context representations are generated. It is parameterized as Θ_1 in Figure 2. The framework could flexibly support various types of GNN models, such as GCN [17], GraphSAGE [22], GIN [18], etc. GIN is adopted in Section IV-B due to its superior empirical performance. After adopting the GNN module, inspired by HGNN [23], we gather messages of nodes for obtaining the context/hyperedge representation. Here, the messages are flowed from nodes to hyperedges, with the aim of: (1) generating hyperedge representations for classification; (2) supporting a more general setting where hyperedges do not share nodes (see Section V). A pooling layer is used on the aggregated features, such as Set2set [24][5].

Next, the hyperedge-level pretext task is adopted in two learning settings. First, in the transductive learning setting, the hyperedge-level pretext task is used as an adaptation stage (Eq. (7)). Second, in the inductive learning setting, the hyperedge-level pretext task is trained as an additional pre-training stage. The hyperedge representations are constructed as the outputs of the pooling layers in the GNN module from the node-level pretext task. The fine-tuning for the downstream task follows the pre-training, with the initialization of the pre-trained GNN module.

Complexity Analysis. With the uniform negative sampling method, the major computation of the model is applying GNN module on all hyperedges for training. Taking GIN as an example, the computational complexity is $O(d^2 n' L m \cdot iter)$ where d is the feature dimension, n' is the number of nodes for each hyperedge, L is the number of layers of GNN module, m is the number of hyperedges, and $iter$ is the number of iterations. Here for notation simplicity, we assume all hyperedges share equal number of nodes n', which is often much smaller than the number of hyperedges m in a hypergraph. For HyperGRL with the exponential negative sampling method, the major computation is calculating $\tilde{\mathbf{A}}^k$ in Eq. (4), which takes $O(kmn')$, where m is the number of non-zero entries in $\mathbf{A}$. The time complexity of METIS algorithm for hyperedge-level pretext task is $O(m + n' + q \cdot log(q))$, where q is the number of clusters [21].

[4]Note that the test data could be observed in the transductive setting, and by self-supervised design, the labels of test data are naturally avoided during training.

[5]In order to achieve permutation equivariance, set module could be adopted, such as Deep Sets [25].

IV. Experiments

In this section, we present the evaluation of the effectiveness and efficiency of the proposed framework on public datasets. The statistics of the datasets are summarized in Table II.

TABLE II: The summary of datasets

Name	# of nodes	# of hyper-edges	Min # of nodes in hyperedge	Max # of nodes in hyperedge
Cora	2,708	2,427	2	169
Pubmed	19,717	3,887	3	20
Corum	6,132	4,736	2	131
Disgenet	8,352	8,386	3	487

A. Experimental Setup

Here, we present the experimental results on four public datasets (*Cora*, *Pubmed*, *Corum*, *Disgenet*) to evaluate the proposed model.

Dataset Processing. In particular, we pre-process two versions of *Cora* and *Pubmed* datasets as *Cora/Pubmed-clean/noisy* as follows. For the first version, we take the ego-network (subgraph of the center node with 1-hop neighbors) of each node as hyperedges, and assign the hyperedge label as the label of the majority in the ego-network. We name this version as the noisy version since the hyperedge might contain nodes with different categories. For the second version, we also take the ego-network of each node as hyperedges, but only keep those whose nodes share the same category. We name this version as the clean version. For *Corum* and *Disgenet*, they are originally hypergraph datasets, and do not need processing.

Model Training Details. First, for HyperGRL with uniform negative sampling, we adopt a local sampling method for the negative hyperedges in order to sample negative nodes. Specifically, the uniform sampling is conducted inside each batch of hyperedges of the stochastic gradient descent algorithm (e.g. Adam optimizer). To this end, the node-level self-supervised pretext task captures the local node-context relationship within a batch at each parameter updating. Since the batch is uniformly sampled, this local sampling method eventually equals to the global uniform sampling on all hyperedges for pre-training. Compared with exponential sampling, which calculates $\tilde{\mathbf{A}}^k$ (Eq. (4)) for globally sampling negative nodes, this uniform negative sampling is more efficient (see Figure 5).

Second, in order to obtain the hyperedge embeddings from the node embeddings as the outputs of GNN module, mean pooling is adpoted on public datasets and Set2set [24] pooling with 1 recurrent layer is adopted on case study because of optimal practical performance. Other sophisticated pooling methods for hypergraphs could be one future direction.

Third, all the adaptation functions which take the node/hyperedge embeddings as inputs for adapting with the self-supervised pretext/downstream tasks are set as MLPs with 0.5 dropout rate. All non-linear activation functions are set as ReLU function.

Hyper-parameter Setting. For the model optimization, we adopt Adaptive Moment Estimation (Adam). The hyper-parameters are set such that the downstream task perform well

TABLE III: Accuracy of hyperedge classification (mean $\pm$ std in %). Bold and underline values indicate the best and the 2-nd best performance respectively. We also conduct a significance test. For all the tables, $\bullet/*$ indicates the result is significantly better/worse than the 2-nd best/the best model with p-value < 0.01, and $\circ$ indicates no significant difference.

Models	Cora-noisy	Cora-clean	Pubmed-noisy	Pubmed-clean	Corum	Disgenet
DHNE	69.85±1.01	72.48±0.52	78.65±1.59	83.23±0.74	53.11±1.39	30.35±0.83
Hyper-SAGNN	66.94±2.12	67.81±1.25	83.20±2.00	83.82±0.62	79.64±0.29	31.74±0.06
SAGE	72.51±1.78	76.01±1.48	83.37±2.32	78.84±1.88	56.22±2.43	18.31±1.37
Joint Training	70.84±0.67	81.65±1.16	85.39±1.48	86.45±0.98	76.85±0.77	34.02±1.28
DW	73.39±1.64	81.66±1.41	82.73±1.70	88.54±0.62	67.35±2.18	29.07±1.45
Deep-Hyperedge	74.35±0.64	72.66±0.93	64.33±0.71	81.58±1.01	82.74±1.64	**35.46±1.08**
HyperGRL (Uni., no Ada.)	77.98±1.81 $\bullet$	**86.41±1.62** $\bullet$	85.18±0.68 $\circ$	88.21±1.51 $\circ$	84.07±0.70 $\bullet$	33.23±1.45 $*$
HyperGRL (Exp., no Ada.)	77.68±2.67 $\bullet$	81.74±1.14 $\circ$	86.29±1.50 $\bullet$	87.81±0.49 $*$	84.49±0.47 $\bullet$	34.10±0.83 $*$
HyperGRL (Uni.)	74.53±1.31 $\bullet$	83.86±3.55 $\bullet$	85.52±0.95 $\circ$	87.23±0.96 $*$	84.31±0.99 $\bullet$	35.05±0.49 $\circ$
HyperGRL (Exp.)	**78.78±1.28** $\bullet$	83.77±1.62 $\bullet$	**86.94±0.73** $\bullet$	**90.84±0.29** $\bullet$	**85.33±1.09** $\bullet$	33.90±2.26 $*$

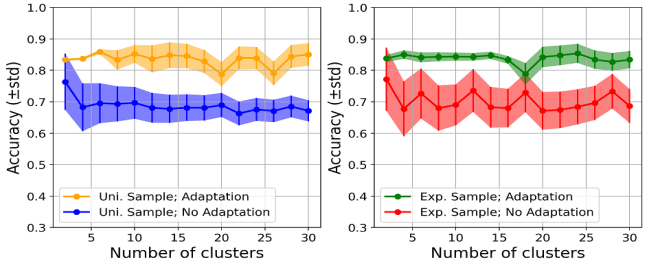

Fig. 3: Accuracy ($\pm$std) vs. # of clusters in Hyperedge-level pretext task on *Corum* dataset

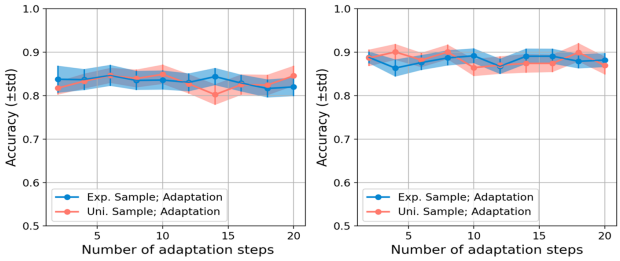

Fig. 4: Accuracy ($\pm$std) vs. # of adaptation steps on *Cora-clean* (left) and *Pubmed-clean* (right) dataset

on validation set. Specifically, for the optimizer, we select the batch size as 64, and learning rate as 0.001 with learning rate scheduler that reduces learning rate on plateau. The number of epochs for node-level pretext task and hyperedge-level pretext task (for traditional pre-training) are set to 50. For the effectiveness evaluations on public datasets, the number of clusters for hyperedge-level pretext task is set as $10, 10, 3, 3, 21, 20$ for *Cora-noisy*, *Cora-clean*, *Pubmed-noisy*, *Pubmed-clean*, *Corum*, *Disgenet* datasets, respectively. The number of GNN layers is set as 1. The number of adaptation steps are set to 5 for all datasets. The dimension of node/hyperedge embeddings are set as 64.

Hardware and Software Details. All experiments are performed on a machine with a Intel(R) Core(TM) i7-9800X CPU (64.0 GB RAM) and GeForce GTX 1080 GPU. The model is implemented with Python 3.6 and Pytorch 1.5.0 [26].

B. Effectiveness Results on Public Datasets

The results for hyperedge classification are presented in Table III. The metric is the multi-class classification accuracy, and the results are mean and standard deviation values over ten runs. All supervised methods share the same training, validation, and testing ratio of 6:2:2. The best results are shown in bold fonts, and the second best results are shown with underlines. We adopt the two-GNN module for the inductive setting (no Ada. in Table III), and one-GNN module for the transductive setting (Ada. in Table III). The GNN module here is GIN. Both exponential and uniform sampling methods are evaluated (shorted as Exp. and Uni. in Table III). For baselines, *Hyper-SAGNN*, *SAGE*, *DHNE* and *Joint Training* use the inductive setting. *Joint Training* is a proposed baseline which trains the two pretext tasks together with the downstream task in a joint loss. *DW* and *Deep-Hyperedge* use the transductive setting.

From the table, we make the following observations. First, for the inductive setting, HyperGRL significantly outperforms all baselines on all datasets (by up to 5.69%), and for the transductive setting, HyperGRL significantly outperforms all baselines on five out of six datasets by up to 4.75%. The framework with adaptation-aware pre-training outperforms the traditional pre-training method in five out of six datasets. The exponential sampling method is competitive with the uniform negative sampling method when no adaptation stage is used, but it shows improvements when adaptation-aware pre-training is applied. Second, GraphSage and DeepWalk are applied on the simple graph transformed from the hypergraph to learn the hyperedge embedding, via incidence matrix $\mathbf{M}$. We can see that these methods do not perform as well as the proposed model for two reasons: (1) by converting to simple graph, the high-order information of hyperedges might be lost; (2) the proposed GNN-based model with pre-training learns generic embeddings which are more suitable for downstream tasks. Third, the 'Joint Training' model is our proposed baseline, which trains the downstream task along with the two self-supervised pretexts in a multi-task fashion. The performance of the joint training model is competitive compared with other baselines, but not as good as the HyperGRL framework. This further indicates the effectiveness of the proposed pre-training methods. Fourth, among the baselines, *Deep Hyperedge*, *Hyper-SAGNN* and *DeepWalk* have relatively better

performance over the rest of the baseline methods, because *DHNE* is originally designed for hyperlink prediction. Note that *Deep Hyperedge* requires hyperedge association as input. HyperGRL does not use such information in downstream tasks, but still outperforms *Deep Hyperedge* in five datasets, and is also competitive on *Disgenet*. For *Disgenet*, the relatively low accuracy is due to the highly imbalanced data with 21 hyperedge categories.

C. Ablation Study

First, we compare the hyperedge classification performance by using different GNN modules in our framework. For all the different versions of GNN modules, we apply the adaptation-aware pre-training strategy, and the exact same hyper-parameters for the rest of the framework. The results are shown in Table IV. The results are the mean and standard deviation values of ten runs. We can see that the GIN module overall shows the best performance over the rest of the GNN modules in our framework for the hyperedge classification task. Also, for the same GNN module, we can see that generally the exponential negative sampling method outperforms the uniform negative sampling method.

Second, we conduct the ablation study on the transductive hyperedge classification performance for bi-level self-supervised pretext tasks (Table V). We apply HyperGRL without any pre-training stage (the first row), with only node-level self-supervised pretext task (the second row), and with only hyperedge-level self-supervised pretext task (the third row). We can observe that the HyperGRL framework with bi-level self-supervised pretext tasks shows the best performance, which demonstrates the effectiveness of both levels of self-supervised pretext tasks.

D. Parameter Sensitivity Results

Here, we study the hyper-parameter sensitivity of our proposed framework. First, we show the results of the sensitivity of the number of clusters in the hyperedge-level pretext task in Figure 3. Note that we use the same model architecture with one-GNN module for all experiments in this subsection. The experiments are conducted in both inductive and transductive settings, with both uniform and exponential negative sampling methods. We observe that: (1) the framework shows relatively stable performance on a large range of the number of clusters; and (2) the proposed framework that uses the adaptation-aware pre-training strategy overall shows more stable performance compared with the framework that does not use the adaptation-aware pre-training. Exponential sampling with adaptation-aware training generally has the best stability. This indicates that the bi-level adaptation-aware pre-training framework has a better adaptation ability for the downstream task. Second, the results of hyperedge classification accuracy vs. the number of adaptation steps are shown in Figure 4. The results demonstrate slight performance drop but relatively stable over the tested adaptation steps in the range of $[1, 20]$ for both exponential and uniform negative sampling methods.

E. Efficiency Results

We compare the efficiency of the adaptation-aware pre-training strategy with the traditional pre-training method as we describe in Subsection III-D. The running time comparison of the pre-training stage is presented in Figure 5. For both methods, the number of node-level pre-training is set equal utill convergence. The number of adaptation steps for adaptation-aware pre-training method is set to 5, which is the same as the setting in effectiveness evaluations. As we can see, adaptation-aware pre-training significantly reduces the pre-training time by 38.2% with Exp. sampling and 42.8% with Uni. sampling. Besides, as we adopt local sampling in the uniform negative sampling method, the running time for uniform negative sampling achieves $\sim$7 times reduction compared with exponential negative sampling.

Combining the results of effectiveness (Table. III) and efficiency (Figure. 5), we can see that the adaptation-aware pre-training strategy not only boosts the performance of downstream task, but also improves the pre-training efficiency compared to traditional pre-training method. The important point here is that it is highly non-trivial to improve the pre-training efficiency without sacrificing accuracy. Furthermore, as discussed in Section III-D, pre-training on one pretext task could be conducted offline and the adaptation-aware pre-training could be conducted whenever the new data for the downstream task is available in an online environment. However, traditional pre-training methods need to be re-trained whenever new data is available.

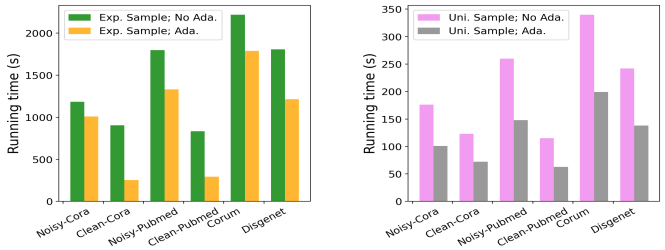

Fig. 5: Running time comparison of the pre-training stage for traditional and adaptation-aware pre-training stragety.

V. A Case Study

In this section, we introduce a real-world application of the proposed model on the inconsistent variation family detection problem in a large online store, and show that how the proposed pre-training strategy can help improve a base classification model (without pre-training) as well as other baselines which do not use pre-training, even when direct hyperedge associations are not available (i.e. hyperedges do not share nodes).

A. Preliminaries and Experimental Setup

Preliminaries. Large E-commerce services such as Amazon, Etsy, and eBay often utilize merchandise relationships such as variations and substitutes to improve their catalog quality. These relationships between merchandise items are the cornerstone in the field of relationship science.

TABLE IV: Accuracy of hyperedge classification for different GNN module (mean ± std in %)

Models	Cora-noisy	Cora-clean	Pubmed-noisy	Pubmed-clean	Corum	Disgenet
GCN (Uni.)	75.58±0.98	82.09±1.26	83.59±0.63	85.96±2.27	65.23±0.48	32.36±0.44
GCN (Exp.)	76.88±2.08	**84.65±3.32**	84.14±1.62	86.06±1.94	83.34±0.25	32.84±0.18
SAGE (Uni.)	75.28±0.69	80.68±0.94	83.89±1.27	86.15±2.71	64.98±0.45	33.33±0.79
SAGE (Exp.)	75.52±1.01	81.04±2.09	85.18±0.86	86.94±0.32	66.03±0.28	31.69±0.21
GAT (Uni.)	73.12±2.56	81.31±0.24	84.15±1.49	86.74±2.67	82.70±2.47	31.89±2.38
GAT (Exp.)	71.21±0.72	80.15±1.68	84.79±0.72	88.49±0.72	**85.75±1.29**	29.89±1.48
GIN (Uni.)	74.53±1.81 ∗	83.86±3.55 ∗	85.52±0.95 •	87.23±0.96 ∗	84.31±0.99 ∗	**35.05±0.49** •
GIN (Exp.)	**78.78±1.28** •	83.77±1.62 ∗	**86.94±0.73** •	**90.84±0.29** •	85.33±1.09 ∗	33.90±2.26 •

TABLE V: Ablation study on bi-level self-supervised pretext tasks (mean ± std in %)

Models	Cora-noisy	Cora-clean	Pubmed-noisy	Pubmed-clean	Corum	Disgenet
No Pre-train	69.12±0.48	81.48±1.24	84.40±1.53	84.60±2.49	78.25±0.13	32.76±1.54
Only Node	74.23±0.84	82.27±1.25	85.04±0.37	86.84±1.94	82.52±2.14	33.75±1.70
Only Hyperedge	73.24±1.53	82.78±1.30	86.29±1.36	85.67±0.38	84.17±1.56	33.40±0.65
HyperGRL (Exp.)	**78.78±1.28** •	**83.77±1.62** •	**86.94±0.73** ○	**90.84±0.29** •	**85.33±1.09** •	**33.90±2.26** ○

Variation family refers to a family of product items which are functionally the same but differ in specific attributes. Such variation families are present in the same detail page. For example, in Figure 6 the correctly grouped 'Nike Air Max 270 React' shoe family is shown in one detail page. All items inside this family are this particular model but have different sizes and colors. The grouping attributes of a variation family are shared by all items inside the variation family (e.g., brand 'Nike'), and the variation theme attributes are attributes which differ from item to item inside the variation family (e.g., size and color). When variation families contain inconsistent items (e.g., an Adidas shoe is grouped into the variation family in Figure 6), they are called inconsistent variation family (IVF). One of the most important tasks in the catalog system is the IVF detection.

In order to solve this problem, each variation family can be regarded as a hyperedge, in which every item should belong to the same merchandise if it is a consistent family. Then, this specific problem is transformed into hyperedge classification problem. As for the pretext tasks, we adopt a task called variation theme learning, which is defined as:

Definition 3: **Variation Theme Learning:** Given a set of variation families (hypergraph $G = (\mathcal{V}, \mathcal{E}, \mathbf{F}^{(n)})$ with $\mathcal{E} = \{e_1, ..., e_m\}$), the Variation Theme Learning aims at learning the variation theme attributes and grouping attributes of the families (i.e. learn which columns of $\mathbf{F}^{(n)}$ correspond to grouping/variation theme attributes).

Note that variation families are independent, which makes the direct hyperedge associations unavailable. Most baselines from Section IV become inapplicable, such as *SAGE*, *DW*, *Deep-Hyperedge*, and *DHNE*. Our idea is to transform this problem as a hyperedge multi-label classification problem since variation families (hyperedges) might have multiple grouping/variation theme attributes. Due to the fact that the labels of grouping/variation theme attributes are available in our catalog system for variation family datasets, this pretext task in the pre-training stage could be supervised. During training, the hyperedge representation produced by HyperGRL is first used for the multi-label supervised pretext task (variation theme learning), and then the model is fine-tuned by the IVF detection task[6]. IVF detection task has binary labels indicating consistent and inconsistent families (binary classification).

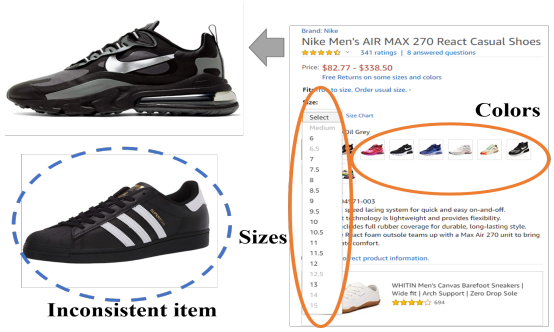

Fig. 6: An example of correctly grouped variation family with variation theme attributes, and an inconsistent item.

Experimental Setup. We sample subsets of three product lines from the catalog system, named Category 1, Category 2, and Category 3. Details of these datasets are show in Table VII.

The proposed method we select for this experiments is HyperGRL with supervised pre-training task, and HyperGRL with supervised pre-training task plus self-supervised node-level adaptation. Here the supervised pre-training task denotes variation theme learning. For comparison, we use three strong baselines, including the joint training method, which jointly trains the pretext task (variation theme learning task) with the targeted IVF classification (denoted as 'Joint training' in Table VI), HyperGRL model without pre-training stage, and the *Hyper-SAGNN* model with and without the help of our proposed pre-training strategy. Note that the original *Hyper-SAGNN* is not a pre-training model. We modify the model to adapt it in our pre-training framework as a strong baseline. Besides the above adaptation, we apply the pre-trained *Glove* embeddings [27] for text features of items.

[6]As suggested by [13], these two tasks are highly likely to be positively correlated. Briefly, inconsistent variation families contain inconsistent items with different distributions of grouping attributes compared with other items, and vice versa.

TABLE VI: Results of IVF classification for case study (mean ± std in %)

Dataset	Category 1		Category 2		Category 3	
Metric	AUC-I	AUC-C	AUC-I	AUC-C	AUC-I	AUC-C
Hyper-SAGNN	56.12±1.19	88.87±0.93	58.72±1.94	89.81±1.64	70.26±1.12	88.47±0.78
Hyper-SAGNN, pre-train	60.26±1.42	90.12±0.38	59.82±0.55	89.57±0.21	72.17±0.34	90.17±1.42
HyperGRL, no pre-train	60.51±0.89	90.32±0.88	58.09±0.57	89.64±0.79	71.50±1.49	89.80±1.72
Joint Training	62.80 ±1.33	**90.99±0.78**	56.73±1.41	89.55±0.82	71.50±1.52	89.80±1.39
HyperGRL, pre-train	**64.06±1.27** •	90.74±1.40 ○	59.42±1.78 ○	89.84±0.70 ○	74.32±0.81 •	**90.62±1.53** •
HyperGRL, pre-train + self-ada.	61.86±1.32 ○	90.31±1.80 ○	**61.73±1.24** •	**90.42±1.74** •	**75.42±2.01** •	90.32±1.85 ○

TABLE VII: Statistics of Datasets for Case Study

Name	# of families	# of items	# of unique items
Category 1	1,186	64,752	9,540
Category 2	2,169	48,860	8,474
Category 3	4,247	138,662	21,131

The metric for this experiment is the PR-AUC for predicting inconsistent variation family (IVF) and consistent variation family (CVF), denoted as AUC-I and AUC-C respectively in Table VI. The results are reported based on the average of five runs.

B. Results on Datasets of Product Lines

As shown in Table VI, first we can see that by using the supervised pretext task (the last three rows), the model performances are consistently better than those without pre-trained pretext task in all product lines. Second, for both methods using pretext task, the pre-training strategy outperforms the joint training strategy by AUC-I metric and they are very close by AUC-C metric. The *Hyper-SAGNN* with the adapted pre-training stage also has relatively competitive performance compared with the proposed HyperGRL model. This indicates that in this real-world application of hyperedge classification, pre-training can indeed improve the base model as well as baseline method, and pre-training is generally a better training strategy than joint training. Third, by utilizing the adaptation stage with the node-level self-supervised pretext task (the last row), the model is able to further improve the AUC-I performance on Categoty-2 and Category-3 product lines, which demonstrates the effectiveness of the adaptation-aware pre-training of HyperGRL in this application scenario.

VI. Related work

GNN-related Research. GNN-related works have been partially covered in Section II. It becomes popular since Bronstein et al. introduce geometric deep learning [28] to bring the convolutional network to the graph learning field, and Kipf et al. significantly simplify the graph convolutional process and develop GCN [17]. To name a few representative works, Velivckovic et al. [19] improve GCN by introducing attentions of neighborhoods, which is calculated dynamically at the message passing step. [18] study the expressiveness of GNN model compared with the closely related Weisfeiler-Lehman graph isomorphism testing algorithm. HyperGCN by Yadati et al. [11] extends the convolutional operation on hypergraphs. A recent work by Zhang et al. [29] adopted high-order subgraphs in GNN for bundle recommendation. The graph neural model on hypergraphs still needs further study for handling its unique high-order relations. Traditional high-order graph mining problem focuses on the similarities between high-order sub-structures and graphs, with applications of subgraph matching [30], [31] and query answering in Knowledge Graphs [32]. [33] provides a complete summary for recent advances in multi-network graph mining and high-order techniques. However, the self-supervised strategy on hypergraphs have not yet been sufficiently studied.

Pre-training Strategy. After the pre-training strategy is first used in NLP communities [12], researches begin to appear in the graph learning field. [13] develop a new strategy and self-supervised methods for pre-training Graph Neural Networks. The key to the success of this method is to pre-train an expressive GNN model at the level of individual nodes as well as entire graphs so that the GNN model can learn meaningful local and global representations. More recently, [15] propose a generative pre-training strategy for GNN models in which a self-supervised attributed graph generation task is introduced. [34] propose a graph contrastive coding method for the pre-training stage in the GNN model. [14] summarize and compare commonly used self-supervised training strategies on graphs and point out a few future directions.

Hypergraph Learning. Among the earliest work, Li et al. [9] transform the hyperlink prediction problem as a ranking problem for hyperedges, and adopt a SimRank-like [35] method. [3] use a coordinated matrix minimization method for nonnegative matrix factorization in the adjacency space of the hypergraphs for hyperedge prediction. More recently, representation learning and GNN-based methods are developed for this direction. Tu et al. propose DHNE [8] which applies auto-encoder and deep neural networks for the representation learning of hyperedges. Hyper-SAGNN by Zhang et al. [2] combines the attention-based dynamic representations and the MLP-based static representations for hyperedge prediction. HGNN by Feng et al. [23] generalizes the convolution operation to the hypergraph by hypergraph Laplacian. Deep Hyperedges [10] by Payne jointly uses contextual and permutation-invariant node memberships of hyperedges for hyperedge classification. DHGNN by Jiang et al. [36] adopts vertex convolution and hyperedge convolution in the dynamic hypergraph representation learning. Yu et al. [37] propose a multi-channel hypergraph convolutional network for social recommendation. Other recent approaches on hypergraph tasks such as hyperlink prediction, stock trend prediction and hypergraph recommendation include [6], [7], [11], [38]–[40].

VII. Conclusion

In this paper, we study the problem of hypergraph representation learning, and propose an end-to-end hypergraph pre-training framework HyperGRL. It incorporates bi-level self-supervised pretext tasks, and enables both transductive and inductive learning. HyperGRL does not require extra domain-specific datasets, and is adaptation-aware, which makes the pre-trained model more adaptive to downstream tasks compared to traditional pre-training methods. Extensive experiments demonstrate that: (1) HyperGRL shows significant improvements of downstream task performance and stability over baselines; (2) HyperGRL is efficient compared with traditional pre-training methods; (3) the proposed framework shows great applicability in real-world applications of online stores. Future directions include further exploration of the divergence between pretext and downstream tasks, and designing hard negative sampling strategies for hypergraphs.

VIII. Acknowledgement

The paper is partially supported by NSF (1947135, 2134079 and 1939725), NIFA (2020-67021-32799) and ARO (W911NF2110088).

References

[1] M. Vaida and K. Purcell, "Hypergraph link prediction: Learning drug interaction networks embeddings," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1860–1865.

[2] R. Zhang, Y. Zou, and J. Ma, "Hyper-sagnn: a self-attention based graph neural network for hypergraphs," *arXiv preprint arXiv:1911.02613*.

[3] M. Zhang, Z. Cui, S. Jiang, and Y. Chen, "Beyond link prediction: Predicting hyperlinks in adjacency space." in *AAAI*, vol. 1, no. 3, 2018.

[4] S. H. Sindrup and T. S. Jensen, "Efficacy of pharmacological treatments of neuropathic pain: an update and effect related to mechanism of drug action," *PAIN®*, vol. 83, no. 3, pp. 389–400, 1999.

[5] J. Piñero, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz, and L. I. Furlong, "The disgenet knowledge platform for disease genomics: 2019 update," *Nucleic acids research*, vol. 48, no. D1, pp. D845–D855, 2020.

[6] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach," in *The World Wide Web Conference*, 2019, pp. 2147–2157.

[7] N. Yadati, V. Nitin, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, "Link prediction in hypergraphs using graph convolutional networks."

[8] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," *arXiv preprint arXiv:1711.10146*, 2017.

[9] D. Li, Z. Xu, S. Li, and X. Sun, "Link prediction in social networks based on hypergraph," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 41–42.

[10] J. Payne, "Deep hyperedges: a framework for transductive and inductive learning on hypergraphs," *arXiv preprint arXiv:1910.02633*, 2019.

[11] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," in *Advances in Neural Information Processing Systems*, 2019, pp. 1511–1522.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[14] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang, "Self-supervised learning on graphs: Deep insights and new direction," *arXiv preprint arXiv:2006.10141*, 2020.

[15] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.

[16] Y. Lu, X. Jiang, Y. Fang, and C. Shi, "Learning to pre-train graph neural networks," 2021.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[20] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[21] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[22] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[23] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3558–3565.

[24] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015.

[25] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in neural information processing systems*, 2017, pp. 3391–3401.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[27] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[28] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[29] Z. Zhang, B. Du, and H. Tong, "Suger: A subgraph-based graph convolutional network method for bundle recommendation," *arXiv preprint arXiv:2205.11231*, 2022.

[30] B. Du, S. Zhang, N. Cao, and H. Tong, "First: Fast interactive attributed subgraph matching," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1447–1456.

[31] L. Liu, B. Du, H. Tong *et al.*, "G-finder: Approximate attributed subgraph matching," in *2019 IEEE international conference on big data (big data)*. IEEE, 2019, pp. 513–522.

[32] L. Liu, B. Du, H. Ji, C. Zhai, and H. Tong, "Neural-answering logical queries on knowledge graphs," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[33] B. Du, S. Zhang, Y. Yan, and H. Tong, "New frontiers of multi-network mining: Recent developments and future trend," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4038–4039.

[34] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[35] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[36] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks." in *IJCAI*, 2019, pp. 2635–2641.

[37] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," *arXiv preprint arXiv:2101.06448*, 2021.

[38] S.-e. Yoon, H. Song, K. Shin, and Y. Yi, "How much and when do we need higher-order informationin hypergraphs? a case study on hyperedge prediction," in *Proceedings of The Web Conference 2020*, 2020.

[39] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," in *EMNLP*, 2020.

[40] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *SIGIR*, 2020.