Semi-supervised Hypergraph Node Classification on Hypergraph Line Expansion

Chaoqi Yang
University of Illinois Urbana-Champaign
chaoqiy2@illinois.edu

Ruijie Wang
University of Illinois Urbana-Champaign
ruijiew2@illinois.edu

Shuochao Yao
George Mason University
shuochao@gmu.edu

Tarek Abdelzaher
University of Illinois Urbana-Champaign
zaher@illinois.edu

ABSTRACT

Previous hypergraph expansions are solely carried out on either vertex level or hyperedge level, thereby missing the symmetric nature of data co-occurrence, and resulting in information loss. To address the problem, this paper treats vertices and hyperedges equally and proposes a new hypergraph expansion named the *line expansion* (LE) for hypergraphs learning. The new expansion bijectively induces a homogeneous structure from the hypergraph by modeling vertex-hyperedge pairs. Our proposal essentially reduces the hypergraph to a simple graph, which enables the existing graph learning algorithms to work seamlessly with the higher-order structure. We further prove that our line expansion is a unifying framework over various hypergraph expansions. We evaluate the proposed LE on five hypergraph datasets in terms of the hypergraph node classification task. The results show that our method could achieve at least 2% accuracy improvement over the best baseline consistently.

CCS CONCEPTS

• **Computing methodologies → Knowledge representation and reasoning, Neural networks**.

KEYWORDS

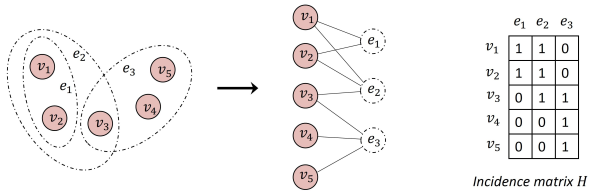
Hypergraph Learning; Hypergraph Expansion; Node Classification

ACM Reference Format:
Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. 2022. Semi-supervised Hypergraph Node Classification on Hypergraph Line Expansion. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557447

1 INTRODUCTION

This paper proposes a new hypergraph structure transformation, namely *line expansion* (LE), for the problem of hypergraph learning. Specifically, this paper focuses on hypergraph node classification.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557447



	e_1	e_2	e_3
v_1	1	1	0
v_2	1	1	0
v_3	0	1	1
v_4	0	0	1
v_5	0	0	1

Figure 1: Bipartite Relation in Hypergraphs. Here, pink circles v denotes vertices (or hypernodes) and dashed circles e denotes hyperedges. They are treated equally in this paper.

The proposed LE is a topological mapping, transforming the hypergraph into a homogeneous structure, while preserving all the higher-order relations. LE allows all the existing graph learning algorithms to work elegantly on hypergraphs.

The problem of hypergraph learning is important. Graph structured data are ubiquitous in practical machine/deep learning applications, such as social networks [10], protein networks [27], and co-author networks [49]. Intuitive pairwise connections among nodes are usually insufficient for capturing real-world higher-order relations. For example, in co-author networks, the edges between authors are created by whether they have co-authored a paper or not. A simple graph structure cannot separate the co-author groups for each paper. For another example, in biology, proteins are bound by poly-peptide chains, thus their relations are naturally higher-order. Hypergraphs allow modeling such multi-way relations, where (hyper)edges can connect to more than two nodes.

However, the research on spectral graph theory for hypergraphs is far less been developed [10]. Hypergraph learning was first introduced in [49] as a propagation process on hypergraph structure, however, [1] indicated that their Laplacian matrix is equivalent to pairwise operation. Since then, researchers explored non-pairwise relationships by developing nonlinear Laplacian operators [7, 28], utilizing random walks [5, 10] and learning the optimal weights [28, 29] of hyperedges. Essentially, most of these algorithms focus on vertices, viewing hyperedges as connectors, and they explicitly break the bipartite property of hypergraphs (shown in Figure 1).

Two types of deep learning models have been designed on hypergraphs. [17] develops Chebyshev formula for hypergraph Laplacians and proposed HGNN. Using a similar hypergraph Laplacian, [46] proposes HyperGCN while [3] generalizes [26, 42] and defines two neural hypergraph operators. However, this first line of works all construct a simple weighted graph and lose high-order information. A recent work HyperSage [2] generalizes the message passing neural networks (MPNN) [19] and uses two-stage message passing functions [15] on hypergraphs. Based on the two-stage procedure, UniGNN [25] generalizes GCN [26], GAT [42], GIN [45] models to hypergraphs and AllSet [9] further unifies a whole class of two-stage models with multiset functions. This type of models are empirically more powerful than the first type, however, they essentially treat the hypergraph as a bipartite graph and build two message passing functions on the heterogeneous structure.

Different from previous works, we propose a novel hypergraph learning model from a new perspective. We propose *line expansion* (LE) for hypergraphs, which is a powerful bijective mapping from a hypergraph structure to a homogeneous graph structure. Relying on our LE mapping, all existing graph representation learning methods [14, 24, 26, 42] can seamlessly and effortlessly work on hypergraphs.

Specifically, from the complex and heterogeneous hypergraph, our LE can induce a simple graph structure (examples in Figure 2), where the "node" is a vertex-hyperedge pair, and "edge" between two "node"s are constructed if two "node"s share the same vertex or hyperedge (w.l.o.g., we use concept "node" in simple graphs and concept "vertex" in hypergraphs). It is interesting that the new induced structure is isomorphic to the line graph of the star expansion of the original hypergraph, which is (i) homogeneous (i.e., a graph where nodes have the same semantics) and (ii) symmetrical with respect to the vertices and hyperedges. We further prove that LE is also (iii) bijective, which means all high-order information is preserved during the transformation, i.e., the hypergraph can be recovered uniquely from the induced line expansion graph.

Therefore, to conduct hypergraph representation learning, we **first** transform the hypergraph to the induced simple graph. **Then**, features from hypergraph vertices are projected to node features in the induced graph. **Next**, we apply graph learning algorithms (i.e., graph convolutional network [26]) to obtain node representations. **Finally**, the node representations from the induced graph is aggregated and back-projected to the original hypergraph vertices for classification. The LE transformation is differentiable and the overall learning process is end-to-end.

The proposed line expansion of hypergraphs is novel and informative. In the traditional formulations, the hyperedges are usually transformed into cliques of edges (e.g., clique/star expansions [1]) or hypergraph cuts [49], or the learning solely depends on edge connectivity (e.g., hyperedge expansions [37]). Differently, LE treats vertices and hyperedges equally, thus preserving the nature of hypergraphs. Note that, LE is also significantly different from other hypergraph formulations, such as tensor based representation [34], line graphs of hypergraphs [6], intersection graphs of hypergraphs [32], or middle graphs of hypergraphs [12]. These formulations either require strong constraints (e.g., uniform hypergraphs) or result in heterogeneous topologies as well as other structures that complicate practical usage. Previous formulations may restrict applicability of graph algorithms due to their special structures.

Further, this paper revisits the formulation of the standard star or clique expansion and simple graph learning algorithms. We conclude that they can be unified as special cases of LE. Empirically, this paper demonstrates the effectiveness of LE on five real-world hypergraphs. We apply the popular graph convolutional networks (GCNs) [26] on LE as our method, and it consistently outperforms several strong hypergraph learning baselines.

The organization of the paper is as follows. In Section 2, we introduce the general notations of hypergraphs and formulate our problem. In Section 3, we propose *line expansion* of hypergraphs and show some interesting properties. In Section 4, we generalize GCNs to hypergraphs by *line expansion*. We evaluate *line expansion* on three-fold experiments in Section 5. We conclude this paper and provide proofs for our theoretical statements in the end.

2 PRELIMINARIES

2.1 Hypergraph Notations

Research on graph-structured deep learning [26, 42] stems mostly from Laplacian matrix and vertex functions of simple graphs. Only recently, learning on hypergraphs starts to attract attentions from the community [2, 3, 9, 17].

Hypergraphs. Let $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$ denote a *hypergraph*, with a vertex set $\mathcal{V}$ and a edge set $\mathcal{E} \subset 2^{\mathcal{V}}$. A hyperedge $e \in \mathcal{E}$ (we also call it "edge" interchangeably in this paper) is a subset of $\mathcal{V}$. Given an arbitrary set $\mathcal{S}$, let $|\mathcal{S}|$ denote the cardinality of $\mathcal{S}$. A simple graph is thus a special case of a hypergraph, with $|e| = 2$ uniformly, which is also called a 2-regular hypergraph. A hyperedge e is said to be *incident* to a vertex v when $v \in e$. One can represent a hypergraph by a $|\mathcal{V}| \times |\mathcal{E}|$ *incidence matrix* $\mathbf{H}$ with its entry $h(v, e) = 1$ if $v \in e$ and 0 otherwise. For each vertex $v \in \mathcal{V}$ and hyperdge $e \in \mathcal{E}$, $d(v) = \sum_{e \in \mathcal{E}} h(v, e)$ and $\delta(e) = \sum_{v \in \mathcal{V}} h(v, e)$ denote their degree functions, respectively. The vertex-degree matrix $\mathbf{D}_v$ of a hypergraph $\mathcal{G}_H$ is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix with each diagonal entry corresponding to the node degree, and the edge-degree matrix $\mathbf{D}_e$ is $|\mathcal{E}| \times |\mathcal{E}|$, also diagonal, which is defined on the hyperedge degree.

2.2 Problem Setup

Following [3, 17], this paper studies the transductive learning problems on hypergraphs, specifically node classification, It aims to induce a labeling $f : \mathcal{V} \rightarrow \{1, 2, \ldots, C\}$ from the labeled data as well as the geometric structure of the graph and then assigns a class label to unlabeled vertices by transductive inference.

Specifically, given a hypergraph $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$ with the labeled vertex set $\mathcal{T} \subset \mathcal{V}$ and the labels, we minimize the empirical risk,

$$f^* = \underset{f(\cdot|\theta)}{\arg\min} \frac{1}{|\mathcal{T}|} \sum_{v_t \in \mathcal{T}} \mathcal{L}(f(v_t \mid \theta), L(v_t)), \tag{1}$$

where $L(v_t)$ is the ground truth label for node v_t and cross-entropy error [26] is commonly applied in $\mathcal{L}(\cdot)$.

Intuitively, node similarity implies similar labels on the same graph. Given the symmetric structure of hypergraphs, we posit that *vertex similarity* and *edge similarity* are equally important in learning the labels. This work focuses on node classification problems on hypergraphs, but it can be easily extended to other hypergraph related applications, such as hyper-edge representation (e.g., relation mining) by exploiting symmetry.

3 HYPERGRAPH LINE EXPANSION (LE)

Most well-known graph-based algorithms [20, 33] are defined for graphs instead of hypergraphs. Therefore, in real-world applications, hypergraphs are often transformed into simple graphs [1, 49] that are easier to handle.

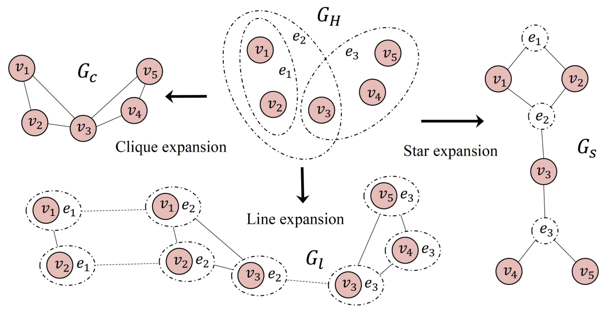


Figure 2: Various Hypergraph Expansions

3.1 Traditional Hypergraph Expansions

Two main ways of approximating hypergraphs by simple graphs are the clique expansion [40] and the star expansion [50]. The *clique expansion* algorithm (shown in left side of Fig. 2) constructs a graph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ from the original hypergraph by replacing each hyperedge with a clique in the resulting graph (i.e., $\mathcal{E}_c = \{(u, v) \mid u, v \in e, e \in \mathcal{E}\}$), while the *star expansion* algorithm (shown in right side of Fig. 2) constructs a new graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ by viewing both vertices and hyperedges as nodes in the resulting graph $\mathcal{V}_s = \mathcal{V} \cup \mathcal{E}$, where vertices and hyperedges are connected by their incident relations (i.e., $\mathcal{E}_s = \{(v, e) \mid v \in e, v \in \mathcal{V}, e \in \mathcal{E}\}$). Note that, the star expansion induces a heterogeneous graph structure.

Unfortunately, these two approximations cannot retain or well represent the higher-order structure of hypergraphs. Let us consider the co-authorship network, as $\mathcal{G}_H$ in Figure 2, where we view authors as nodes (e.g., v_1, v_2) and papers as hyperedges (e.g., e_1). Then we immediately know that author v_1 and v_2 have jointly written one paper e_1, and together with author v_3, they have another co-authored paper e_2. This hierarchical and multi-way connection is an example of *higher-order* relation. Assume we follow the clique expansion, then we obviously miss the information of author activity rate and whether the same persons jointly writing two or more articles. Though researchers have remedially used weighted edges [10, 28], the hyper-dependency still collapses or fuses into linearity. Star expansion expresses the whole incidence information, but the remaining heterogeneous structure (i) has no explicit vertex-vertex link and (ii) is too complicated for those well-studied graph algorithms, which are mostly designed for simple graphs. One can summarize [23] that these two expansions are not good enough for many real-world applications.

3.2 Our Line Expansion

Since the commonly used expansions cannot give a satisfactory representation, we seek a new expansion that preserves all the original higher-order relations, while presenting an easy-to-learn graph structure. Motivated by the special symmetric structure of hypergraphs that *vertices are connected to multiple edges and edges are conversely connected to multiple vertices*, we treat vertices and edges equally and propose hypergraph *Line Expansion* (LE).

The *Line Expansion* of the hypergraph $\mathcal{G}_H$ is constructed as follows (shown in Fig. 2, bottom): (i) each incident vertex-hyperedge pair is considered as a "line node"; (ii) "line nodes" are connected when they share the same vertex or hyperedge. Essentially, the induced structure is a graph, where each vertex or each hyperedge (from the original hypergraph) induces a clique (i.e., fully-connected subgraph). We now formally define the *line expansion*, denoted $\mathcal{G}_l$.

3.2.1 Line Expansion. Let $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ denotes the graph induced by the *line expansion* of hypergraph $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}_l$ of $\mathcal{G}_l$ is defined by vertex-hyperedge pair $\{(v, e) \mid v \in e, v \in \mathcal{V}, e \in \mathcal{E}\}$ from the original hypergraph. The edge set $\mathcal{E}_l$ and *adjacency* $\mathbf{A}_l \in \{0, 1\}^{|\mathcal{V}_l| \times |\mathcal{V}_l|}$ is defined by pairwise relation with $\mathbf{A}_l(u_l, v_l) = 1$ if either $v = v'$ or $e = e'$ for $u_l = (v, e), v_l = (v', e') \in \mathcal{V}_l$.

The construction of the *line expansion* follows the neighborhood aggregation mechanism. For graph node representation learning, researchers [14, 26] encode local structure by aggregating information from a node's immediate neighborhood. In *line expansion*, we view the incidence of vertex-hyperedge as a whole and generalize the "neighborhood" concept by defining that *two line nodes are neighbors when they contain the same vertex (vertex similarity) or the same hyperedge (edge similarity)*. We argue that the *line expansion* consequently preserves higher-order associations.

3.3 Entity Projection

In this section, we define the projection and back-projection matrices for hypergraph entities (i.e., vertices and hyperedges) between the topological map from $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$ to $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$. In $\mathcal{G}_l$, each line node (v, e) could be viewed as a vertex with hyperedge context or a hyperedge with vertex context. In a word, the *line expansion* creates information linkage in the higher-order space.

Vertex Projection Matrix. To scatter the information, a vertex $v \in \mathcal{V}$ from the original hypergraph $\mathcal{G}_H$ is mapped to a set of line nodes $\{v_l = (v, e) : e \in \mathcal{E}\} \subset \mathcal{V}_l$ in the induced graph $\mathcal{G}_l$. We introduce the *vertex projection matrix* $\mathbf{P}_{vertex} \in \{0, 1\}^{|\mathcal{V}_l| \times |\mathcal{V}|}$,

$$\mathbf{P}_{vertex}(v_l, v) = \begin{cases} 1 & \text{if the vertex part of } v_l \text{ is } v, \\ 0 & otherwise, \end{cases} \tag{2}$$

where each entry indicates whether the line node contains the vertex.

To re-obtain the information of a vertex v in $\mathcal{G}_H$, we aggregate a set of line nodes in $\mathcal{G}_l$ who shares the same vertex, for example $v_l = (v, e)$. Since each line node $v_l = (v, e)$ contains the edge context, we consider using the reciprocal of edge size, i.e., $\frac{1}{\delta(e)}$ or $\frac{1}{|e|}$ (check the definition of $\delta(\cdot)$ and $|\cdot|$ in Section 2.1), as the weights for aggregating the line nodes, such that if $\delta(e)$ is smaller (meaning that v is important under the context of e), the corresponding line node (v, e) will contribute more to the aggregation.

Vertex Back-projection Matrix. With this intuition, we fuse the higher-order information by defining the *vertex back-projection matrix* $\mathbf{P}'_{vertex} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}_l|}$,

$$\mathbf{P}'_{vertex}(v, v_l) = \begin{cases} \dfrac{\frac{1}{\delta(e)}}{\sum_{(v,e') \in \mathcal{V}_l} \frac{1}{\delta(e')}} & \text{if } v \text{ is the vertex part of } v_l, \\ 0 & otherwise. \end{cases} \tag{3}$$

Similarly, we could also design *edge projection and back-projection matrices*, $\mathbf{P}_{edge} \in \mathbb{R}^{|\mathcal{V}_l| \times |\mathcal{E}|}$ and $\mathbf{P}'_{edge} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{V}_l|}$, to exchange information from edges in $\mathcal{G}_H$ to line nodes in $\mathcal{G}_l$.

In fact, the uniqueness of topological inverse mapping from $\mathcal{G}_l$ to $\mathcal{G}_H$ is guaranteed by Theorem 1, where the complete information of vertex $v \in \mathcal{V}$ is re-obtained by aggregating all the distributed parts $(v, \cdot) \in \mathcal{V}_l$ from $\mathcal{G}_l$.

Theorem 1. *From the line expansion graph, we can uniquely recover the original hypergraph. Formally, the mapping ϕ from hypergraph to line expansion (i.e., $\phi : \mathcal{G}_H \to \mathcal{G}_l$) is bijective.*

3.4 Additional Properties of Line Expansion

In this section, we discuss additional properties of line expansion (LE). First, we present an observation between characteristic matrices from $\mathcal{G}_H$ and $\mathcal{G}_l$. Then, we connect our *line expansion* with the line graph from graph theory, based on which, some sound properties could be derived.

Observation 1. *Let $\mathbf{H}$ be the incidence matrix of a hypergraph $\mathcal{G}_H$. $\mathbf{D}_v$ and $\mathbf{D}_e$ are the vertex and hyperedge degree matrices. Let $\mathbf{P}_{vertex}$ and $\mathbf{P}_{edge}$ be the vertex and edge projection matrix, respectively. $\mathbf{A}_l$ is the adjacency matrix of line expansion $\mathcal{G}_l$. Let $\mathbf{H}_r = [\mathbf{P}_{vertex}, \mathbf{P}_{edge}] \in \{0,1\}^{|\mathcal{V}_l| \times (|\mathcal{V}|+|\mathcal{E}|)}$, it satisfies the following equations,*

$$\mathbf{H}_r^\top \mathbf{H}_r = \begin{bmatrix} \mathbf{D}_v & \mathbf{H} \\ \mathbf{H}^\top & \mathbf{D}_e \end{bmatrix}, \tag{4}$$

$$\mathbf{H}_r \mathbf{H}_r^\top = 2\mathbf{I} + \mathbf{A}_l. \tag{5}$$

In Observation 1, the left hand of both Eqn. (4) and Eqn. (5) are the projection matrices, and the right hand of these two equations are information respectively from the hypergraph and its *line expansion*. Essentially, these equations quantify the transition from $\mathcal{G}_H$ to $\mathcal{G}_l$. For Eqn. (5), we are interested in the product of $\mathbf{H}_r\mathbf{H}_r^\top$, leading to two orders of self-loop, which would be useful in the analytical aspects of *line expansion* (in Section 7.2).

Theorem 2. *For a hypergraph, its line expansion $\mathcal{G}_l$ is isomorphic to the line graph of its star expansion $L(\mathcal{G}_s)$, where $L(\cdot)$ is a line graph notation from graph theory.*

Theorem 2 is the foundation of Theorem 1, which provides a theoretical interpretation and enriches our expansion with sound graph theoretical properties (readers could refer to line graph theory [11]). That is why we name our formulation "line expansion". Note that the *line expansion* is significantly different from the "line graph of hypergraph" discussed in [4, 6]. Instead, our line expansion is the line graph of the star expansion. Thus, the proof of Theorem 2 is naturally established by the construction of line expansion.

Based on Theorem 2, we know that $\mathcal{G}_l$ is homogeneous and has the same connectivity with $\mathcal{G}_H$. The number of new edges in $\mathcal{G}_l$ could be calculated as $|\mathcal{E}_l| = \frac{\sum_v d(v)(d(v)-1)}{2} + \frac{\sum_e \delta(e)(\delta(e)-1)}{2}$ and line nodes as $|\mathcal{V}_l| = \frac{\sum_v d(v) + \sum_e \delta(e)}{2}$. In the worse case, for a fully-connected k-regular hypergraph ($k \ll |\mathcal{V}|$), $|\mathcal{V}_l| = \Theta(k|\mathcal{E}|)$ and $|\mathcal{E}_l| = \Theta(\frac{k^2}{2}|\mathcal{E}|^2)$, where Θ is the big Theta notation for the tightest bound. However, many real hypergraphs are indeed sparse (e.g., degrees of vertices and hyperedges follow long-tailed distribution [30], most of them have degree one, $|\mathcal{V}| \ll |\mathcal{E}|$ or $|\mathcal{E}| \ll |\mathcal{V}|$), so that the scale could usually reduce to $|\mathcal{V}_l| = \Theta(\frac{|\mathcal{V}|+|\mathcal{E}|}{2})$ and $|\mathcal{E}_l| = O(|\mathcal{V}||\mathcal{E}|)$.

4 HYPERGRAPH REPRESENTATION LEARNING

Transductive learning on graphs is successful due to the fast localization and neighbor aggregation [14, 26, 42]. It is easy to define the information propagation pattern upon simple structures. For real-world cases, relationships among objects are usually more complex than pairwise. Therefore, to apply these algorithms, we need a succinct informative structure of the higher order relations.

Shown in Section 3, the bijective map from $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$ to $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ equipped with four entity projectors ($\mathbf{P}_{vertex}$, $\mathbf{P}'_{vertex}$, $\mathbf{P}_{edge}$, $\mathbf{P}'_{edge}$) fills the conceptual gap between hypergraphs and graphs. With this powerful tool, it is possible to transfer the hypergraph learning problems into graph structures and address them by using well-studied graph representation algorithms. Note that, this work focuses on the generic hypergraphs without edge weights.

4.1 Hypergraph Learning with Line Expansion

In this section, we generalize *graph convolution networks* (GCNs) [26] to hypergraphs and introduce a new learning algorithm defined on *line expansion* for hypergraph representation. Note that, on our proposed structure, other graph representation algorithms could be migrated similarly [22, 36, 41, 42].

4.1.1 Overall Pipeline. To address the transductive node classification problems on hypergraphs, we organize the pipeline of our proposed model as the following three steps.

- STEP1: vertices of the hypergraph is mapped to line nodes in the induced graph. Specifically, we use the proposed *vertex projection matrix* $\mathbf{P}_{vertex}$ to conduct feature mapping.
- STEP2: we apply deep graph learning algorithms (e.g., GCNs) to learn the representation for each line node.
- STEP3: the learned representation is fused by the *vertex back-projection matrix* $\mathbf{P}'_{vertex}$ in an inverse edge degree manner. The vertex labels are predicted on the fused representation.

4.2 Convolution on Line Expansion

STEP 1: Feature Projection. Given the initial feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_i}$ (d_i is input dimension) from $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$, we transform it into the features in $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ by vertex projector $\mathbf{P}_{vertex}$,

$$\mathbf{H}^{(0)} = \mathbf{P}_{vertex}\mathbf{X} \in \mathbb{R}^{|\mathcal{V}_l| \times d_i}. \tag{6}$$

$\mathbf{H}^{(0)}$ is the initial node feature of the induced graph. This projection essentially scatters features from vertex of $\mathcal{G}_H$ to feature vectors of line nodes in $\mathcal{G}_l$. In *line expansion* (LE), a line node could be adjacent to another line nodes that contain the same vertex (*vertex similarity*) or the same hyperedge (*edge similarity*).

STEP 2: Convolution Layer. We then apply neighborhood feature aggregation by graph convolution. By incorporating information from both vertex-similar neighbors and hyperedge-similar neighbors, the graph convolution is defined as ($k = 0, 1, \dots, K$),

$$h_{(v,e)}^{(k+1)} = \sigma\left(\sum_{e'} w_e h_{(v,e')}^{(k)} \Theta^{(k)} + \sum_{v'} w_v h_{(v',e)}^{(k)} \Theta^{(k)}\right), \tag{7}$$

where $h_{(v,e)}^{(k)}$ denotes the feature representation of line node (v, e) in the k-th layer, $\sigma(\cdot)$ is a non-linear activation function like *ReLU*

[26] or *LeakyReLU* [42]. $\Theta^{(k)}$ is the transformation parameters for layer k. Two hyper-parameters w_v, w_e are employed to balance *vertex similarity* and *edge similarity*. Specifically, in Eqn. (7), the first term (i.e., $\sum_{e'} w_e h^{(k)}_{(v,e')}$) convolves information from neighbors who share the same hyperedges, whereas the second term (i.e., $\sum_{v'} w_v h^{(k)}_{(v',e)}$) convolves information from neighbors who share the same vertices.

Eqn. (7) can be written in matrix version by using the parameterized *adjacency matrix*(In experiment, we use the $w_v = w_e = 1$),

$$\mathbf{A}_l(u_l, v_l) = \begin{cases} w_e & u_l = (v,e),\ v_l = (v',e'),\ v = v', \\ w_v & u_l = (v,e),\ v_l = (v',e'),\ e = e', \\ 0 & \qquad\qquad\qquad\qquad otherwise, \end{cases} \tag{8}$$

and adopt the *renormalized trick* [26] with the adjustment two-orders of self-loop: $2\mathbf{I} + \mathbf{D}_l^{-\frac{1}{2}}\mathbf{A}_l\mathbf{D}_l^{-\frac{1}{2}} \rightarrow \tilde{\mathbf{D}_l}^{-\frac{1}{2}}\tilde{\mathbf{A}_l}\tilde{\mathbf{D}_l}^{-\frac{1}{2}}$ (here, $\tilde{\mathbf{A}_l} = 2I + \mathbf{A}_l$ and $\tilde{\mathbf{D}}_{l(ii)} = \sum_j \tilde{\mathbf{A}}_{l(ij)}$). Eqn. (7) can be re-written as,

$$\mathbf{H}^{(k+1)} = \sigma\left(\tilde{\mathbf{D}_l}^{-\frac{1}{2}}\tilde{\mathbf{A}_l}\tilde{\mathbf{D}_l}^{-\frac{1}{2}}\mathbf{H}^{(k)}\Theta^{(k)}\right),\ k = 0, 1, \ldots, K. \tag{9}$$

In real practice, we do not bother to calculate the adjacency $\mathbf{A}_l$ directly. An efficient trick is to use Eqn. (5).

STEP 3: Representation Back-projection. After K layers, $\mathbf{H}^{(K)}$ is the final node representation on the induced graph, from which we could derive fused representation for vertices in $\mathcal{G}_H$. Specifically, we use the back-projector $\mathbf{P}'_{vertex}$,

$$\mathbf{Y} = \mathbf{P}'_{vertex}\mathbf{H}^{(K)} \in \mathbb{R}^{|\mathcal{V}|\times d_o}, \tag{10}$$

where d_o is the dimension of output representation. Note that, in this work, we focus on the node classification task. However, due to the symmetry of vertex and edge, this work also sheds some light on the applications of learning hyper-edges (e.g., relation mining) by using $\mathbf{P}_{edge}$, $\mathbf{P}'_{edge}$. We leave it to future work.

In sum, the complexity of 1-layer convolution is of $O(|\mathcal{E}_l|d_i d_o)$, since the convolution operation could be efficiently implemented as the product of a sparse matrix with a dense matrix.

4.3 Unifying Hypergraph Expansion

As discussed in Section 3.1, for hypergraphs, common practices often collapse the higher order structure into simple graph structures by attaching weights on edges, and then the vertex operators are solely applied onto the remaining topology. Therefore, the interchangeable and complementary nature between nodes and edges are generally missing [31].

Theorem 3. *Line expansion is a generalization of clique expansion and star expansion. The convolution operator on* LE *is a generalization of simple graph convolution.*

In this work, instead of designing a local vertex-to-vertex operator [3, 17, 48, 49], we treat the vertex-hyperedge relation as a whole. Therefore, the neighborhood convolution on line expansion is equivalent to exchanging information simultaneously across vertices and hyperedges. Our proposed line expansion (LE) is powerful in that it unifies clique and star expansions, as well as simple graph cases, stated in Theorem 3. We formulate different hypergraph expansions and provide the proof in Section 7.2.

4.4 Acceleration: Neighborhood Sampling

For practical usage, we further accelerate the proposed model by neighborhood sampling. As is mentioned, the runtime complexity of our model is proportional to the number of connected edges, $|\mathcal{E}_l|$, of *line expansion*. Real-world hypergraphs are usually sparse, however, most of them often have vertices with large degrees, which would probably lead to a large or dense *line expansion* substructure.

On simple graphs, [21] proposes neighbor sampling for high-degree vertices, which randomly samples a few neighbors to approximate the aggregation of all neighbors through an unbiased estimator. This paper adopts neighborhood sampling techniques to hypergraph regime and mitigates the potential computational problems in real applications. Since our graph convolution involves both *vertex similarity* and *edge similarity* information, we design two threshold, δ_e and δ_v, for two neighboring sets, separately.

We use $\mathcal{N}_E(v,e)$ to denote the hyperedge neighbor set of line node (v,e). Essentially, $\mathcal{N}_E(v,e)$ contains line nodes with same hyperedge context e. Similarly, we use $\mathcal{N}_V(v,e)$ as the vertex neighbor set, which contains line nodes with v as the vertex part. For a line node with high "edge degree", i.e., $|\mathcal{N}_E(v,e)| > \delta_e$, we would randomly sample δ_e elements from $\mathcal{N}_E(v,e)$ to approximate the overall hyperedge neighboring information. Specifically, the first term in Eqn. (7), i.e., $\sum_{e'} w_e h^{(k)}_{(v,e')}$, is approximated by,

$$\sum_{e'} w_e h^{(k)}_{(v,e')} \approx \frac{|\mathcal{N}_E(v,e)|}{\delta_e} \sum_{i=1:\ e'_i\sim\mathcal{N}_E(v,e)}^{i=\delta_e} w_e h^{(k)}_{(v,e'_i)}. \tag{11}$$

Similarly, when $|\mathcal{N}_V(v,e)| > \delta_v$, we would sample δ_v elements from $\mathcal{N}_V(v,e)$ for the convolution,

$$\sum_{v'} w_v h^{(k)}_{(v',e)} \approx \frac{|\mathcal{N}_V(v,e)|}{\delta_v} \sum_{i=1:\ v'_i\sim\mathcal{N}_V(v,e)}^{i=\delta_v} w_v h^{(k)}_{(v'_i,e)}. \tag{12}$$

In sum, by adopting the neighbor sampling into hypergraphs, we could effectively reduce the receptive field and prevent the computational problem incurred by high-degree vertices. In the experiments, we empirically show that the running time of our model is comparable to state of the art baselines after sampling.

5 EXPERIMENTS

We comprehensively evaluated the proposed *line expansion* (LE) with the following experiments and released the implementations[1]:

- Real-world hypergraph node classification.
- Special case: simple graph node classification.
- Ablation study on the choice of w_e and w_v.

We name our proposed hypergraph learning approach as LE_{GCN}.

5.1 Hypergraph Node Classification

The main experiment is demonstrated on five real-world hypergraphs with four traditional and four SOTA hypergraph learning methods. The metric is classification accuracy.

Hypergraph Datasets. The first dataset *20Newsgroups* contains 16,242 articles with binary occurrence values of 100 words. Each word is regarded as a hyperedge and the news articles are vertices.

[1]https://github.com/ycq091044/LEGCN

Table 1: Statistics of Hypergraph Datasets

Dataset	Vertices	Hyperedges	Features	Class	Label rate	Training / Validation / Test
20News	16,242	100	100	4	0.025	400 / 7,921 / 7,921
Mushroom	8,124	112	112	2	0.006	50 / 4,062 / 4,062
Zoo	101	42	17	7	0.650	66 / – / 35
ModelNet40	12,311	12,321	2048	40	0.800	9,849 / 1,231 / 1,231
NTU2012	2,012	2,012	2048	67	0.800	1,608 / 202 / 202

Table 2: Structural Complexity Comparison of Baselines and Line Expansion (before and after sampling)

Dataset	* of the clique expansion graph			* of its line expansion (before)			* of its line expansion (after)		
	Node	Exp. edge	Exp. density	Line node	Line edge	Density	Line node	Line edge	Density
20News	16,242	26,634,200	2.0e-1	64,363	34,426,427	1.6e-2	64,363	240,233	1.2e-4
Mushroom	8,124	6,964,876	2.1e-1	40,620	11,184,292	1.2e-2	40,620	81,532	9.9e-5
Zoo	101	5,050	1.0e-0	1,717	62,868	4.3e-2	1,717	16,171	1.1e-2
ModelNet40	12,311	68,944	9.1e-4	61,555	317,083	1.7e-4	61,555	310,767	1.6e-4
NTU2012	2,012	10,013	4.9e-3	10,060	48,561	9.6e-4	10,060	48,561	9.6e-4

Exp. edge is given by the clique expansion, and Exp. density is computed by $2|E|/|V|(|V|-1)$ [13].

The next two datasets are from the UCI Categorical Machine Learning Repository [16]: *Mushroom, Zoo*. For these two, a hyperedge is created by all data points which have the same value of categorical features. We follow the same setting from [23] for 20Newsgroups, Mushroom, Zoo (which does not have validate set due to the small scale). Other two are from computer vision/graphics area: *Princeton CAD ModelNet40* [44] and *National Taiwan University (NTU) 3D dataset* [8]. Though semi-supervised learning usually requires a small training set, we copy the same settings from the original paper [17] and use 80% of the data as training and the remaining 20% is split into validation and test. The construction of hypergraphs also follows [17]. Each CAD model is regarded as a vertex. The formation of hyperedges is by applying MVCNN [39] on CAD models and then for each model, we assume that its 10 nearest models form a hyperedge. The initial vertex features are given by GVCNN representations [18]. Basic statistics of datasets are reported in Table 1. The graph structure of clique expansion, the line expansion before and after neighbor sampling are reported in Table 2.

Baselines. We select the following baselines.

- *Logistic Regression (LR)* works as a standard baseline, which only uses independent feature information.
- *Clique*$_{GCN}$ and *Star*$_{GCN}$ are developed by applying GCN on the clique or star expansions of the hypergraphs.
- *H-NCut* [49], equivalent to *iH-NCut* [28] with uniform hyperedge cost, is a generalized spectral method for hypergraphs. This paper considers *H-NCut* as another baseline.
- *LHCN* [4] considers the concept of line graph of a hypergraph, however, it irreversibly transforms into a weighted graph.
- Hyper-Conv [3] and *HGNN* [17] are two recent models, which uses hypergraph Laplacians to build the convolution operators.
- *HyperGCN* [46] approximates each hyperedge by a set of pairwise edges connecting the vertices of the hyperedge.

Experimental Setting. In the experiment, we set $w_v = w_e = 1$ for our model (computation of the adjacency matrix is by Eqn. (5)). All hyperparameters are selected: GCN with 2 hidden layers and 32 units, 50 as training epochs, $\delta_1 = \delta_2 = 30$ as sampling thresholds, Adam as the optimizer, $2e^{-3}$ as learning rate, $5e^{-3}$ as weight decay, 0.5 as dropout rate, and $1.5e^{-3}$ as the weight for L_2 regularizer. Note that hyperparameters might vary for different datasets, and we specify the configurations per dataset in code appendix. All the experiments are conducted 5 times (to calculate mean and standard deviation) with *PyTorch 1.4.0* and mainly finished in a 18.04 LTS Linux server with 64GB memory, 32 CPUs and 4 GTX-2080 GPUs.

Result Analysis. As shown in Table 3, overall our model beat SOTA methods on all datasets consistently. Basically, every model works better than LR, which means transductive feature sharing helps in the prediction. The performances of traditional Clique$_{GCN}$ and Star$_{GCN}$ are not as good as SOTA hypergraph learning baselines. H-Ncut method depends on linear matrix factorization and it also cannot beat graph convolution methods, which are more robust and effective with non-linearity. The remaining four are all graph based deep learning methods, and in essence, they approximate the original hypergraph as a weighted graph and then utilize vertex functions on the flattened graph. The result shows that Our LE$_{GCN}$ is more effective in terms of learning representation and could beat them by 2% consistently over all datasets.

Discussion of Complexity. We also report the running time comparison in Table 3, which already include the neighbor sampling time in our model. Basically, our model is also efficient compared to some state-of-the-arts, especially on 20News, ModelNet40 and NTU2012, which demonstrates that neighbor sampling does make our proposed model less expensive. Let us investigate this in depth. SOTA hypergraph baselines operate on a flattened hypergraph (identical to clique expansion) with designed edge weights. We calculate the number of edges and density for them, denoted as Exp. edge and Exp. density. As shown in Table 2, we find that the scale of *line expansion* before sampling is within 5 times of the flattened topology, except for Zoo (flattened topology is a complete graph). However, after sampling the neighbors, line expansion structure has been significantly simplified, and we could also observe that for most of the datasets, the density of the LE graph is much smaller ($\sim \frac{1}{1000}$) than the flattened clique expansion graph.

Table 3: Accuracy and Running Time Comparison on Real-world Hypergraphs (%)

Model	20News	Mushroom	Zoo	ModelNet40	NTU2012
LR	72.9 ± 0.7	81.6 ± 0.1	74.3 ± 0.0	59.0 ± 2.8	37.5 ± 2.1
Star$_{GCN}$	68.8 ± 0.4	91.8 ± 0.3	95.2 ± 0.0	90.0 ± 0.0	79.1 ± 0.0
Clique$_{GCN}$	69.0 ± 0.3	90.0 ± 0.6	94.8 ± 0.3	89.7 ± 0.4	78.9 ± 0.8
H-NCut [49]	72.8 ± 0.5	87.7 ± 0.2	87.3 ± 0.5	91.4 ± 1.1	74.8 ± 0.9
LHCN [4]	69.1 ± 0.4 (37.6s)	90.2 ± 0.3 (18.4s)	55.8 ± 0.1 (1.8s)	90.2 ± 0.2 (145.1s)	79.9 ± 0.5 (27.4s)
Hyper-Conv [3]	73.1 ± 0.7 (72.6s)	93.7 ± 0.6 (10.6s)	93.1 ± 2.3 (0.8s)	91.1 ± 0.8 (63.1s)	79.4 ± 1.3 (6.3s)
HGNN [17]	74.3 ± 0.2 (74.2s)	93.1 ± 0.5 (16.1s)	92.0 ± 2.8 (0.8s)	91.7 ± 0.4 (61.0s)	80.0 ± 0.7 (5.6s)
HyperGCN [46]	73.6 ± 0.3 (147.8s)	92.3 ± 0.3 (30.23s)	93.1 ± 2.3 (1.1s)	91.4 ± 0.9 (86.5s)	80.4 ± 0.7 (9.7s)
LE$_{GCN}$	**75.6 ± 0.2** (38.6s)	**95.2 ± 0.1** (18.9s)	**97.0 ± 0.0** (2.8s)	**94.1 ± 0.3** (85.9s)	**83.2 ± 0.2** (9.9s)

Table 4: Statistics of Citation Networks

Dataset	Nodes	Edges	Features	Class	Label rate
Cora	2,708	5,429	1,433	7	0.052
Citeseer	3,327	4,732	4,732	6	0.036
Pubmed	19,717	44,338	500	3	0.003

5.2 Simple Graph Node Classification

Since simple graphs are a special case of hypergraphs, 2-regular hypergraph, we apply *line expansion* to simple graphs to empirically verify our Theorem 3 and show that applying graph learning algorithm on line expansion can achieve comparable results.

Datasets. *Cora* dataset has 2,708 vertices and 5.2% of them have class labels. Nodes contain sparse bag-of-words feature vectors and are connected by a list of citation links. Another two datasets, *Citeseer* and *Pubmed*, are constructed similarly [38]. We follow the setting from [47] and show statistics in Table 4.

Common Graph-based Methods. We consider the popular deep end-to-end learning methods GCN [26] and well-known graph representation methods SpectralClustering (SC) [33], Node2Vec [20], DeepWalk [36] and LINE [41]. We first directly apply these methods on the simple graphs. Then, we apply them on the line expansion of the simple graph, named $\mathsf{LE}_{(\cdot)}$, for example, $\mathsf{LE}_{Node2Vec}$. Note that, GCNs could input both features and the graph structure (i.e., adjacency matrix), whereas other methods only use structural information.

Result Analysis. The accuracy results of node classification for three citation networks are shown in Table 5. The experiment clearly demonstrates that LE shows comparable results in graph node classification tasks. Specifically for those non-end-to-end methods, they consistently outperform the original algorithm on simple graphs. The reason might be that LE enriches the plain structure by providing a finer-grained structure and makes nodes edge-dependent, which might explain the improvement in structure-based non-end-to-end models. End-to-end GCNs can reach a much higher accuracy compared to other baselines. We observe that LE_{GCN} tie with original GCN on the three datasets.

5.3 Ablation Study on on w_e and w_v

In this section, we conduct ablation studies on w_v and w_e. Since only the fraction $\frac{w_e}{w_v}$ matters, we symmetrically choose $\frac{w_e}{w_v}$ = $0,\ 0.1,\ 0.2,\ 0.5,\ 1,\ 2,\ 5,\ 10, \infty$ and calculate the test accuracy.

Table 5: Graph Node Classification Accuracy (%)

Model	Cora	Citeseer	Pubmed
SC	53.3 ± 0.2	50.8 ± 0.7	55.2 ± 0.4
Planetoid	75.0 ± 0.9	64.0 ± 1.3	76.7 ± 0.6
ICA	74.5 ± 0.6	63.4 ± 0.6	72.9 ± 1.0
Node2Vec	66.3 ± 0.3	46.2 ± 0.7	71.6 ± 0.5
DeepWalk	62.8 ± 0.6	45.7 ± 1.2	63.4 ± 0.4
LINE	27.7 ± 1.1	30.8 ± 0.2	53.5 ± 0.8
GCN	82.6 ± 0.7	70.5 ± 0.3	78.2 ± 0.6
LE$_{SC}$	56.9 ± 0.2	50.7 ± 0.2	71.9 ± 0.7
LE$_{Planetoid}$	76.6 ± 0.4	66.0 ± 0.7	77.0 ± 0.2
LE$_{ICA}$	72.7 ± 0.4	68.6 ± 0.5	73.3 ± 0.7
LE$_{Node2Vec}$	74.3 ± 0.4	46.2 ± 0.1	74.3 ± 0.4
LE$_{DeepWalk}$	68.3 ± 0.1	50.4 ± 0.4	68.0 ± 0.8
LE$_{LINE}$	51.7 ± 0.2	34.9 ± 0.5	57.5 ± 0.3
LE$_{GCN}$	82.3 ± 0.5	70.4 ± 0.3	78.7 ± 0.4

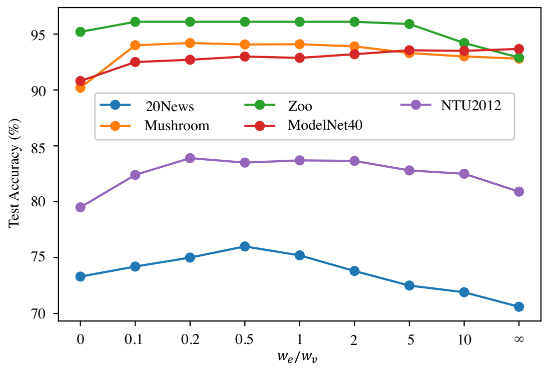


Figure 3: Ablation Study on $\frac{w_e}{w_v}$

Figure 3 provides some intuitions on how to select proper w_v and w_e in real hypergraph tasks. We can conclusion that these hypergraphs have different sensitivities and preferences for w_e and w_v. However, we do find all the curves follow a first-rise-and-then-down pattern, meaning that it is beneficial to aggregate information from both edge-similar and vertex-similar neighbors. Specifically, we find that for hypergraphs with fewer hyperedges, e.g., 20News and Mushroom, the peak appears before $\frac{w_e}{w_v}$ = 1, and for hypergraphs with sufficient hyperedges, e.g., ModelNet40 and NTU2012, the peak appears after $\frac{w_e}{w_v}$ = 1. Therefore, one empirical guide for practical usage is to set smaller w_e when there are fewer hyperedges

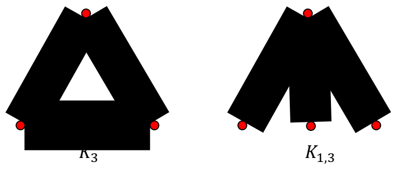


Figure 4: The Exception of Whitney's Theorem

and set larger w_e, vice versa. After all, using the binary version (i.e., $w_e = w_v = 1$) seems to be a simple and effective choice.

6 CONCLUSION AND FUTURE WORKS

In this paper, we proposed a new hypergraph transformation, *Line Expansion* (LE), which can transform the hypergraph into simple homogeneous graph in an elegant way, without loss of any structure information. With LE, we extend the graph convolution networks (GCNs) to hypergraphs and show that the extended model outperforms strong baselines on five real-world datasets.

A possible future direction is to exploit the hypergraph symmetry and apply LE for edge learning in complex graphs. Another interesting extension is to extend *line expansion* to directed graphs, where the relations are not reciprocal. In future works, we will further evaluate our model on large hypergraphs, such as DBLP or Yelp, against recent two-stage type hypergraph learning baselines, such as AllSet [9], which could be one limitation of the paper.

7 PROOFS

7.1 Proof of Theorem 1

First, we posit without proof that the hypergraph $\mathcal{G}_H$ has one-to-one relation with its star expansion $\mathcal{G}_s$. To prove the bijectivity of mapping $\phi : \mathcal{G}_H \to \mathcal{G}_l$, we can instead prove the bijectivity between $\mathcal{G}_s$ and $\mathcal{G}_l$. Our proof will be based on the Whitney graph isomorphism theorem [43] below.

Theorem 4. *(Whitney Graph Isomorphism Theorem.) Two connected graphs are isomorphic if and only if their line graphs are isomorphic, with a single exception: K_3, the complete graph on three vertices, and the complete bipartite graph $K_{1,3}$, which are not isomorphic but both have K_3 as their line graph.*

Definition 1. *(Maximum Independent set.) A maximum independent set is an independent node set (no two of which are adjacent) of largest possible size for a given graph $\mathcal{G}$.*

Proof. For the star expansion of the hypergraph, it could be unconnected when subsets of the vertices are only incident to subsets of the hyperedges. In that case, we could consider the expansion as a union of several disjoint connected components and apply the proof on each component. Below, we mainly discuss the case when $\mathcal{G}_s$ is connected.

The proof consists of two parts. First, we show that for the class of star expansion graphs, Theorem 4 holds without exception. Second, we show how to recover the star expansion $\mathcal{G}_s$ (equivalently, the original hypergraph $\mathcal{G}_H$) from $\mathcal{G}_l$.

First, for the exception in Whitney's theorem, it is obvious that K_3 (in Figure 4) cannot be the star expansion of any hypergraph.

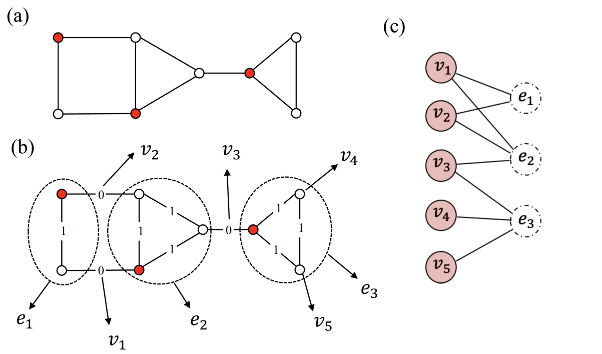


Figure 5: The construction from G_l to G_s

Therefore, for star expansion graphs (with is also a bipartite representation of the hypergraph), Theorem 4 holds without exception.

Second, given a line graph topology, we know from Theorem 4 immediately that the original bipartite structure is unique. We now provide a construction from $\mathcal{G}_l$ to $\mathcal{G}_s$. Given a line graph structure, we first find a maximum independent set (in Definition 1) and color them in red (shown in Figure 5 (a)). [35] proves that the maximum independent node could be found in polynomial time.

Since every vertex and every hyperedge from $\mathcal{G}_s$ spans a clique in $L(\mathcal{G}_s)$, let us think about the line node in the induced graph (which is Fig. 5.(b) here), which is potentially a vertex-hyperedge pair. Therefore, each node (v, e) must be connected to exactly two cliques: one spanned by vertex v and one spanned by hyperedge e. Essentially, we try to project these cliques back to original vertex or hyperedges in G_s. In fact, for each colored node, we choose one of two cliques connected to it so as to make sure: i) the selected cliques have no intersections (there are two choices. In this case, choose 1-edge cliques or 0-edge cliques) and ii) the set of cliques cover all nodes in the topology, shown in Fig. 5 (b).

For any given line graph topology (of a star expansion graph), we could always find the set of 1-edge cliques or the set of 0-edge cliques that satisfies i) and ii), guaranteed by Definition 1. Conceptually, due to the bipartite nature, one set will be the cliques spanned by original hyperedges and another set will be the cliques spanned by original vertices. Either will work for us. Note that the set of 1-edge cliques also includes two size-1 clique, denoted as v_4 and v_5 in Fig. 5 (b). They seem to only connect to one 1-edge clique, i.e, e_3 clique, however, they are actually size-1 cliques spanned by the original vertices which belongs to only one hyperedge.

The recovery of the star expansion $\mathcal{G}_s$ is as follows: First, find a maximum independent set. Second, choose the set of 1-edge clique and transform each selected clique as a hyperedge. Then, the vertex set is created two-folded: i) a clique with 0 on its edges is a vertex in $\mathcal{G}_H$; ii) nodes only connected to one 1-edge clique are also vertices. By the symmetry of hypergraph, the vertex set and the hyperedge set can also be flipped, but the resulting topology is isomorphic. □

7.2 Proof of Theorem 3

We first formulate the clique, star and our line expansion adjacency matrices and then show the unification evidence.

Clique and Star Expansion Adjacency. Given a hypergraph $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$, consider the clique expansion $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$. For each pair $(u,v) \in \mathcal{E}_c$,

$$\mathbf{A}_c(u,v) = \frac{w_c(u,v)}{\sqrt{d_c(u)}\sqrt{d_c(v)}}, \tag{13}$$

where in standard clique expansion, we have,

$$w_c(u,v) = \sum h(u,e)h(v,e), \tag{14}$$

$$d_c(u) = \sum h(u,e)(\delta(e)-1). \tag{15}$$

For the same hypergraph $\mathcal{G}_H = (\mathcal{V},\mathcal{E})$, star expansion gives $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$. We adopt adjacency formulation from [1], formally,

$$\mathbf{A}_s(u,v) = \sum_{e\in E} \frac{h(u,e)h(v,e)}{\delta(e)^2\sqrt{\sum_e h(u,e)}\sqrt{\sum_e h(v,e)}}. \tag{16}$$

Line Expansion Adjacency. To analyze the adjacency relation on *line expansion*, we begin by introducing some notations. Let us use $h^{(k)}_{(v,e)}$ (in short, h^k_{ve}) to denote the representation of line node $(v,e) \in V_l$ at the k-th layer. The convolution operator on *line expansion*, in main text Eqn. (7), can be presented,

$$h^{k+1}_{ve} = \frac{w_e \sum_{e'} h^k_{ve'} + w_v \sum_{v'} h^k_{v'e}}{w_e(d(v)-1) + w_v(\delta(e)-1)}. \tag{17}$$

We augment Eqn. (17) by applying 2-order self-loops (mentioned in Section 4.2), and it yields,

$$h^{k+1}_{ve} = \frac{w_e \sum_{e'} h^k_{ve'} + w_v \sum_{v'} h^k_{v'e}}{w_e d(v) + w_v \delta(e)}. \tag{18}$$

The above operator is defined on the induced graph, we equivalently convert it into the hypergraph domain by back-projector $\mathbf{P}'_{vertex}$. Formally, assume x^k_u as the converted representation for vertex u in hypergraph, Eqn. (18) can be written as,

$$x^{k+1}_u = \frac{\sum_e h(u,e)\frac{1}{\delta(e)} \frac{w_v \sum_{u'} x^l_{u'} + w_e \sum_u x^l_u}{w_v\delta(e) + w_e d(u)}}{\sum_e h(u,e)\frac{1}{\delta(e)}}. \tag{19}$$

After organizing the equation, we calculate that for each hypergraph vertex pair $(u,v) \in \mathcal{V}\times\mathcal{V}$, they are adjacent by,

$$\mathbf{A}_l(u,v) = \frac{\sum_e \frac{w_v h(u,e)h(v,e)}{\delta(e)(w_v\delta(e)+w_e d(u))}}{\sum_e h(u,e)\frac{1}{\delta(e)}}, \tag{20}$$

or by the following form after symmetric re-normalization,

$$\mathbf{A}_l(u,v) = \frac{\sum_e \frac{w_v h(u,e)h(v,e)}{\delta(e)\sqrt{w_v\delta(e)+w_e d(u)}\sqrt{w_v\delta(e)+w_e d(v)}}}{\sqrt{\sum_e h(u,e)\frac{1}{\delta(e)}}\sqrt{\sum_e h(v,e)\frac{1}{\delta(e)}}}. \tag{21}$$

Unifying Star and Clique Expansion. We start by considering the clique expansion graph with weighting function,

$$w_c(u,v) = \sum_{e\in E} \frac{h(u,e)h(v,e)}{(\delta(e)-1)^2}. \tag{22}$$

Note that this is equivalent to vanish Eqn. (14) by a factor of $\frac{1}{(\delta(e)-1)^2}$. We plug the value into Eqn. (15), then adjacency of clique expansion transforms into,

$$\mathbf{A}_c(u,v) = \frac{\sum_e \frac{h(u,e)h(v,e)}{(\delta(e)-1)^2}}{\sqrt{\sum_e h(u,e)\frac{1}{\delta(e)-1}}\sqrt{\sum_e h(v,e)\frac{1}{\delta(e)-1}}}. \tag{23}$$

Note that when we set $w_e = 0$ (no message passing from hyperedge-similar neighbors). The higher-order relation of *line expansion*, in Eqn. (21) degrades into,

$$\mathbf{A}_l(u,v) = \frac{\sum_e \frac{h(u,e)h(v,e)}{\delta(e)^2}}{\sqrt{\sum_e h(u,e)\frac{1}{\delta(e)}}\sqrt{\sum_e h(v,e)\frac{1}{\delta(e)}}}. \tag{24}$$

Eqn. (24) is exactly the adjacency of star expansion in Eqn. (16), and Eqn. (23) (adjacency of clique expansion) is the 1-order self-loop form of the degraded *line expansion*.

Unifying Simple Graph Adjacency. The convolution operator [26] on a simple graph can be briefly present,

$$\mathbf{A}(u,v) = \frac{\sum_e h(u,e)h(v,e)}{\sqrt{d(u)}\sqrt{d(v)}}. \tag{25}$$

A graph could be regarded as a 2-regular hypergraph, where hyperedge e has exactly two vertices, i.e., $\delta(e) = 2$ and each pair of vertices $(u,v) \in \mathcal{V}\times\mathcal{V}$ has at most one common edge. Plugging the value into Eqn. (24), and it yields,

$$\mathbf{A}_l(u,v) = \frac{\sum_e h(u,e)h(v,e)}{2\sqrt{d(u)}\sqrt{d(v)}}. \tag{26}$$

Comparing Eqn. (25) and (26), the only difference is a scaling factor 2, which could be absorbed into filter Θ.

7.3 Proof of Observation 1

First, we have (use $\mathbf{P}_v$ to denote $\mathbf{P}_{vertex}$ and $\mathbf{P}_e$ for $\mathbf{P}_{edge}$):

$$\mathbf{H}_r^\top \mathbf{H}_r = \begin{bmatrix} \mathbf{P}_v^\top \\ \mathbf{P}_e^\top \end{bmatrix} \begin{bmatrix} \mathbf{P}_v \mathbf{P}_e \end{bmatrix} = \begin{bmatrix} \mathbf{P}_v^\top\mathbf{P}_v & \mathbf{P}_v^\top\mathbf{P}_e \\ \mathbf{P}_e^\top\mathbf{P}_v & \mathbf{P}_e^\top\mathbf{P}_e \end{bmatrix} = \begin{bmatrix} \mathbf{D}_v & \mathbf{H} \\ \mathbf{H}^\top & \mathbf{D}_e \end{bmatrix}, \tag{27}$$

where the last equality is easy to verify since i) $\mathbf{P}_v^\top\mathbf{P}_v$ implies the vertex degree matrix, which is $\mathbf{D}_v$. ii) $\mathbf{P}_e^\top\mathbf{P}_e$ implies the hyperedge degree matrix, which is $\mathbf{D}_e$; iii) $\mathbf{P}_v^\top\mathbf{P}_e$ implies the vertex-hyperedge incidence, which is $\mathbf{H}$.

For Eqn. (5), each row of $\mathbf{H}_r$ is a 0/1 vector of size $|\mathcal{V}|+|\mathcal{E}|$ with each dimension indicating a vertex or a hyperedge. Therefore, the vector has exactly two 1s, which is due to that a line node contains exactly one vertex and one hyperedge.

For the (i,j)-th entry of $\mathbf{H}_r\mathbf{H}_r^\top$, it is calculated by the dot product of row i (line node i) and row j (line node j) of $\mathbf{H}_r$. If $i = j$, then this entry will be 2 (dot product of the same 0/1 vector with two 1s). If $i \neq j$, the result will be 0 if line node i and line node j has no common vertex or hyperedge and be 1 if they share vertex or hyperedge (the corresponding dimension gives 1 and 0 for other dimensions, summing to 1). In sum, $\mathbf{H}_r\mathbf{H}_r^\top$ is equal to the adjacency $\mathbf{A}_l$ with 2-order self-loops, quantitatively,

$$\mathbf{H}_r\mathbf{H}_r^\top = 2\mathbf{I} + \mathbf{A}_l. \tag{28}$$

ACKNOWLEDGEMENTS

This work was in part supported by DARPA award HR001121C0165, DoD Basic Research Office award HQ00342110002, NSF grants IIS-2107200 and CPS-2038658.

REFERENCES

[1] Sameer Agarwal, Kristin Branson, and Serge J. Belongie. 2006. Higher order learning with graphs. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006.* 17–24.

[2] Devanshu Arya, Deepak K Gupta, Stevan Rudinac, and Marcel Worring. 2020. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv preprint arXiv:2010.04558* (2020).

[3] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2019. Hypergraph Convolution and Hypergraph Attention. *CoRR* abs/1901.08150 (2019).

[4] Sambaran Bandyopadhyay, Kishalay Das, and M Narasimha Murty. 2020. Line Hypergraph Convolution Network: Applying Graph Convolution for Hypergraphs. *arXiv preprint arXiv:2002.03392* (2020).

[5] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2013. Random walks in hypergraph. In *International Conference on Applied Mathematics and Computational Methods.*

[6] Jean-Claude Bermond, Marie-Claude Heydemann, and Dominique Sotteau. 1977. Line graphs of hypergraphs I. *Discrete Mathematics* 18, 3 (1977), 235–241.

[7] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. 2018. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)* 65, 3 (2018), 1–48.

[8] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On visual similarity based 3D model retrieval. In *Computer graphics forum.* Wiley Online Library, 223–232.

[9] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. 2021. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264* (2021).

[10] Uthsav Chitra and Benjamin J. Raphael. 2019. Random Walks on Hypergraphs with Edge-Dependent Vertex Weights. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA.* 1172–1181.

[11] Fan RK Chung and Fan Chung Graham. 1997. *Spectral graph theory.* American Mathematical Soc.

[12] EJ Cockayne, ST Hedetniemi, and DJ Miller. 1978. Properties of hereditary hypergraphs and middle graphs. *Canad. Math. Bull.* 21, 4 (1978), 461–468.

[13] Thomas F Coleman and Jorge J Moré. 1983. Estimation of sparse Jacobian matrices and graph coloring blems. *SIAM journal on Numerical Analysis* 20, 1 (1983), 187–209.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain.* 3837–3845.

[15] Yihe Dong, Will Sawin, and Yoshua Bengio. 2020. HNHN: Hypergraph Networks with Hyperedge Neurons. *arXiv preprint arXiv:2006.12278* (2020).

[16] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[17] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[18] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. 2018. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 264–272.

[19] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning.* PMLR, 1263–1272.

[20] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* 855–864.

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems.* 1024–1034.

[22] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.

[23] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. 2013. The Total Variation on Hypergraphs - Learning on Hypergraphs Revisited. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.* 2427–2435.

[24] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[25] Jing Huang and Jie Yang. 2021. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956* (2021).

[26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26.*

[27] Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. 2009. Hypergraphs and cellular networks. *PLoS computational biology* 5, 5 (2009), e1000385.

[28] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems.* 2308–2318.

[29] Pan Li and Olgica Milenkovic. 2018. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. *arXiv preprint arXiv:1803.03833* (2018).

[30] Zemin Liu, Trung-Kien Nguyen, and Yuan Fang. 2021. Tail-GNN: Tail-Node Graph Neural Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 1109–1119.

[31] Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M. Bronstein. 2018. Dual-Primal Graph Convolutional Networks. *CoRR* abs/1806.00770 (2018).

[32] Ranjan N Naik. 2018. On Intersection Graphs of Graphs and Hypergraphs: A Survey. *arXiv preprint arXiv:1809.08472* (2018).

[33] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems.* 849–856.

[34] Xavier Ouvrard, Jean-Marie Le Goff, and Stéphane Marchand, Maillet. 2017. Adjacency and tensor representation in general hypergraphs part 1: e-adjacency tensor uniformisation using homogeneous polynomials. *arXiv preprint arXiv:1712.08189* (2017).

[35] Vangelis Th Paschos. 2010. *Combinatorial optimization and theoretical computer science: interfaces and perspectives.* Vol. 24. John Wiley & Sons.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 701–710.

[37] Li Pu and Boi Faltings. 2012. Hypergraph learning with hyperedge expansion. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 410–425.

[38] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[39] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision.* 945–953.

[40] Liang Sun, Shuiwang Ji, and Jieping Ye. 2008. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 668–676.

[41] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web.* International World Wide Web Conferences Steering Committee, 1067–1077.

[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018.*

[43] Hassler Whitney. 1992. Congruent graphs and the connectivity of graphs. In *Hassler Whitney Collected Papers.* Springer, 61–79.

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1912–1920.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[46] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2018. HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification. *arXiv preprint arXiv:1809.02589* (2018).

[47] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* (2016).

[48] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and T.-H. Hubert Chan. 2017. Re-revisiting Learning on Hypergraphs: Confidence Interval and Subgradient Method. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* 4026–4034.

[49] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006.* 1601–1608.

[50] Jason Y Zien, Martine DF Schlag, and Pak K Chan. 1999. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design of integrated circuits and systems* 18, 9 (1999), 1389–1399.