A Survey on Hypergraph Representation Learning

ALESSIA ANTELMI, Università degli Studi di Torino, Italy
GENNARO CORDASCO, Università della Campania "Luigi Vanvitelli", Italy
MIRKO POLATO, Università degli Studi di Torino, Italy
VITTORIO SCARANO and CARMINE SPAGNUOLO, Università degli Studi di Salerno, Italy
DINGQI YANG, University of Macau, Macau SAR, China, China

Hypergraphs have attracted increasing attention in recent years thanks to their flexibility in naturally modeling a broad range of systems where high-order relationships exist among their interacting parts. This survey reviews the newly born hypergraph representation learning problem, whose goal is to learn a function to project objects—most commonly nodes—of an input hyper-network into a latent space such that both the structural and relational properties of the network can be encoded and preserved. We provide a thorough overview of existing literature and offer a new taxonomy of hypergraph embedding methods by identifying three main families of techniques, i.e., spectral, proximity-preserving, and (deep) neural networks. For each family, we describe its characteristics and our insights in a single yet flexible framework and then discuss the peculiarities of individual methods, as well as their pros and cons. We then review the main tasks, datasets, and settings in which hypergraph embeddings are typically used. We finally identify and discuss open challenges that would inspire further research in this field.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Mathematics of computing** → **Hypergraphs**; • **Computing methodologies** → **Machine learning;**

Additional Key Words and Phrases: Hypergraph representation learning, hypergraph embedding, hypergraph neural networks, hypergraph convolution, hypergraph attention

ACM Reference format:
Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. 2023. A Survey on Hypergraph Representation Learning. *ACM Comput. Surv.* 56, 1, Article 24 (August 2023), 38 pages.
https://doi.org/10.1145/3605776

This work has been partially supported by the spoke "FutureHPC & BigData" of the ICSC–Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing funded by European Union–NextGenerationEU, University of Macau (SRG2021-00002-IOTSC), and the Science and Technology Development Fund, Macau SAR (0047/2022/A1).
Authors' addresses: A. Antelmi (corresponding author) and M. Polato (corresponding author), Università degli Studi di Torino, Torino, Italy; email: {alessia.antelmi, mirko.polato}@unito.it. G. Cordasco, Università della Campania "Luigi Vanvitelli", Caserta, Italy; email: gennaro.cordasco@unicampania.it. V. Scarano and C. Spagnuolo, Università degli Studi di Salerno, Fisciano, Italy; email: {vitsca, cspagnuolo}@unisa.it. D. Yang, University of Macau, Macau SAR, China; email: dingqiyang@um.edu.mo.

0360-0300/2023/08-ART24 $15.00
https://doi.org/10.1145/3605776

1 INTRODUCTION

Hypergraphs are the natural representation of a broad range of systems where group (or high-order or many-to-many) relationships exist among their interacting parts. Technically speaking, a hypergraph is a generalization of a graph where a (hyper)edge allows the connection of an arbitrary number of nodes [24]. Such structures can easily abstract social systems where individuals interact in groups of any size [17, 102]; for instance, in the case of a co-authorship collaboration network, a hyperedge may represent an article and link together all authors (nodes) having collaborated on it [10, 48, 49, 74, 86, 120, 139, 143, 159, 171] (see Figure 1). Similar situations, characterized by high-order interactions, also exist in biology [56, 132], ecology [47, 58], and neuroscience [83, 106]. Despite their powerful expressiveness, hypergraphs have been underexplored in the literature (in favor of their graph counterpart) because of their inherent complexity and the lack of appropriate tools and algorithms. Recently, the trend has been drifting, thanks to a rising number of systematic studies demonstrating how the transformation of a hypergraph to a classical graph either leads to an inevitable loss of information or creates a large number of extra nodes/edges that increases space and time requirements in downstream graph analytic tasks [2, 17, 175, 178]. Specifically, hypergraphs have been proven to be a critical tool to use when the underlying system to study exhibits highly non-linear interactions between its constituents [17]. In practice, this consideration translates into using hyperedges to model (possibly indecomposable) group interactions that cannot be described simply in terms of dyads (and, hence, via graphs). For instance, hypergraph modeling has been exploited to embed key sociological concepts such as homophily (i.e., the influence a group exerts on a single individual) and conformity (i.e., group pressure, namely the tendency of an individual to align their beliefs to those of their peers, often reinforced by the nature of shared opinions) to investigate the dynamics of opinion formation [81, 115, 116, 130] and social influence diffusion [6, 142, 157, 195, 196] when groups are explicitly taken into account. Similarly, hypergraphs have been used to model epidemic-spreading processes to expressly account for community structure and non-linear infection pressure [5, 22, 71], and group dynamics [39, 82, 92, 107]. Throughout this review, we will discuss other application scenarios in which hypergraph modeling could be more beneficial than traditional graph modeling and elaborate on the characteristics of the many-to-many relations abstracted and the specific tasks addressed.

All graph-related problems and corresponding challenges still hold for the hypergraph-based setting, where the computational cost is even more significant due to the presence of high-order interactions [112, 135]. In this sense, the task of hypergraph representation learning (a.k.a. hypergraph embedding) further assumes a critical role in effectively and efficiently solving analytic problems. Embedding a network—either a graph or a hypergraph—means projecting its structure and possibly additional information onto a low-dimensional space where the structural and semantic information (e.g., nodes' neighborhood and features) is ideally preserved. The underlying idea of this procedure is that representing the nodes and (hyper)edges as a set of low-dimensional

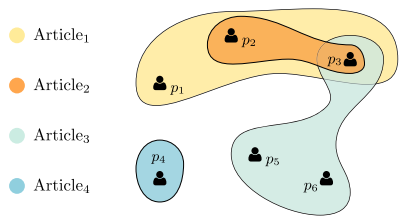


Fig. 1. An example of co-authorship hypergraph where each node represents an author, and each hyperedge connects all authors that have contributed to the same article.

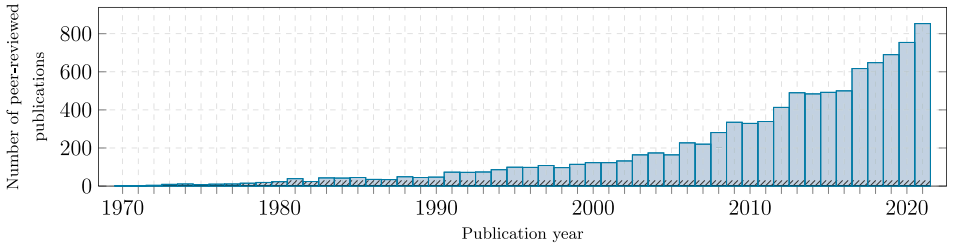


Fig. 2. Number of peer-reviewed publications related to hypergraphs from 1970 to 2021 (source: Scopus).

vectors allows the efficient execution of the traditional vector-based machine learning algorithms on the (hyper)graph. As for graphs, the problem of hypergraph embedding, thus, lies in the overlap of two traditional research problems: hypergraph analytics and representation learning [26]. While the first problem aims to mine useful information from the hyper-network structure, representation learning seeks to learn compact representations (i.e., latent feature vectors) when addressing, for instance, classification [10, 138, 194], link prediction [45, 176, 191], and recommendation [152, 153, 181] tasks. Learning a latent representation of hypergraphs, rather than graphs, enables the exploration of the high-order correlation among data and the indecomposable nature of certain group relations to build more comprehensive representations, leading to better performance in practice [136, 179].

As witnessed in Figure 2, for the last decade, a growing body of work has devoted its efforts to investigating hypergraphs to design more effective solutions in various domains [17]. Still, so far, there is no systematic exploration of hypergraph embedding methods.

This survey aims to fill this gap by providing a thorough overview of existing literature and offering a taxonomy of hypergraph embedding techniques. Our broadest intent is to develop a comprehensive understanding and critical assessment of the knowledge of this newly born research area.

Differences with previous surveys. Current surveys [54, 186] closely related to ours substantially differ in the topics discussed and the literature covered.

The work from Gao et al. [54] deals with the hypergraph learning problem (sometimes called hypergraph regularization), which is a related but different topic than hypergraph representation learning. According to [54], hypergraph learning is the process of passing information along the hypergraph topology in analyzing the structured data and solving problems such as node classification. Learning hypergraph embeddings is not the goal of hypergraph learning, although both tasks share some concepts and ideas (e.g., [194]). Further, the authors thoroughly analyze hypergraph generation methods (not covered in this survey).

The most recent survey from Zhang et al. [186] presents a shallow and brief excursus on hypergraph representation learning techniques, offering a single-level taxonomy similar to the one provided in this survey (see Section 5). However, the authors limit the discussion to a few representative works, focusing on how to handle uncertain data using hypergraphs. Their survey also covers some graph representation learning and hypergraph generation methods.

For the above reasons, our work can be regarded as a complement to the surveys by Gao et al. [54] and Zhang et al. [186] since it emphasizes the task of hypergraph representation learning by providing a series of novel contributions that are listed below.

Contributions. Our contributions can be summarized as follows:

- *Inherent challenges.* As hypergraphs are a generalization of graphs, some challenges are directly inherited by the graph representation learning problem. However, the high-order

nature of hypergraphs imposes additional difficulties. We discuss the classical challenges of (hyper)graph embedding to then detail the peculiar challenges these structures pose.
- *New taxonomies.* We propose three taxonomies, classifying hypergraph embedding methods based on (1) their learning approach (spectral, proximity-preserving, and neural network techniques), (2) the structure of the input hypergraph (homogeneous/heterogeneous, undirected/directed, uniform/non-uniform, static/dynamic, attributed/not attributed nodes, transformation into a graph), and (3) the desired output (node/hyperedge embedding).
- *Comprehensive review.* The recent booming of the hypergraph representation learning field enabled us to collect, systematically review, and characterize the whole evolution of this research area. We describe the most representative approaches of each hypergraph embedding class, unraveling the pros and cons of each learning mechanism and highlighting the high-level connections among the described techniques within each category.
- *Hypergraph analytic tasks.* We review the discussed methods under the lens of representation learning applications, categorizing them according to node and hyperedge-related tasks.
- *Future directions.* We examine the limitations of the current state of the art and propose six potential future directions in the area in terms of problem setting, modeling techniques, interpretability, and scalability.

Article organization. The remainder of this survey is organized as follows. Section 2 details the paper collection process and the inclusion criteria for selecting the articles included in this survey. Section 3 introduces the concepts and the notation we will use throughout this work. Section 4 formally defines the problem of hypergraph representation learning (see Section 4.1), describes a taxonomy of the problem setting in terms of hypergraph embedding input and output (see Section 4.2), and discusses the problem's inherent challenges (see Section 4.3). Section 5 categorizes the literature based on the embedding technique, describing spectral representation learning (see Section 5.1), proximity-preserving (see Section 5.2), and (deep) neural network (see Section 5.3) methods, unraveling their pros and cons and comparing these three methodologies (see Section 5.4). Section 6 presents examples of applications enabled by the hypergraph embedding methods previously described. Section 7 identifies and discusses open research challenges and future directions in this field. Finally, Section 8 concludes this survey.

2 METHODOLOGY

Please refer to Appendix A in the online Supplemental Material.

3 FUNDAMENTALS

This section introduces the concepts we will use throughout the article, from formally defining hypergraphs to describing how these structures can be transformed into their graph counterparts. Table 1 lists the mathematical notation and the hypergraph-related concepts that will be explicitly referred to in the remainder of this survey.

3.1 Hypergraphs

A **hypergraph** is an ordered pair $H = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes,[1] and $\mathcal{E}$ is the set of hyperedges (Figure 3(a)). Each hyperedge is a non-empty subset of nodes. The structure of a hypergraph is usually represented by an **incidence matrix** $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, with each entry $\mathbf{H}(v,e)$ indicating whether the vertex v is in the hyperedge e, i.e., $\mathbf{H}(v,e) = [\![v \in e]\!]$.

[1]In this survey, we use the terms "node" and "vertex" interchangeably.

Table 1. Summary of the Notation Used throughout This Survey

Category	Symbol/Concept	Interpretation
General	$\mathbf{v}$	Boldfaced lowercase letters are used to identify vectors.
	$\mathbf{A}$	Boldfaced capital letters are used to identify matrices.
	$\mathbf{A}(i,j)$	The indexed notation identifies the jth element in the ith row of the matrix $\mathbf{A}$.
	$\mathcal{A}$	Calligraphic capital letters are used to identify sets.
	$\lvert\mathcal{A}\rvert$	The cardinality (i.e., number of elements) of the set $\mathcal{A}$.
	$[\![P]\!]$	Indicator function: It equals 1 if the predicate P is true, 0 otherwise.
	$[n]$	Set of integers from 1 to n, i.e., $\mathbb{N}_{\leq n}$.
	$\mathbf{I}_n$	The n by n identity matrix.
	$\text{diag}(\mathbf{A})$	Diagonal of the matrix $\mathbf{A}$.
	$\text{diag}(\mathbf{v})$	Diagonal matrix with its diagonal equal to $\mathbf{v}$.
Hypergraph related	$\mathcal{V}$	Vertex set, where $\lvert\mathcal{V}\rvert = n$.
	$\mathcal{E}$	Hyperedge set, with $\lvert\mathcal{E}\rvert = m$.
	$\mathcal{E}_v \equiv \{e \in \mathcal{E} \mid v \in e\}$	Hyperedges containing the vertex v.
	$\mathbf{H}$	Incidence matrix of the hypergraph H.
	$\kappa(v)$	Degree of a vertex v.
	$\delta(e)$	Degree or cardinality of a hyperedge e.
	$w(e)$	Weight of a hyperedge e.
	$\mathbf{W}$	Diagonal matrix of hyperedge weights.
	$\mathbf{D}_v$	Diagonal matrix of node degrees.
	$\mathbf{D}_e$	Diagonal matrix of hyperedge cardinalities.
	H^*	Dual hypergraph H^* of H, constructed by swapping the role of nodes and hyperedges.
	k-uniform	Hypergraph in which each hyperedge has cardinality k.
Representation learning	$\mathbf{L}$	Hypergraph Laplacian.
	$\mathbf{X}$	Nodes' feature matrix.
	σ	Non-linear activation function.
	$\mathbf{\Theta}, \mathbf{\Phi}$	Learnable neural network parameters.

The **degree of a vertex** v and the **degree of a hyperedge** e are defined as

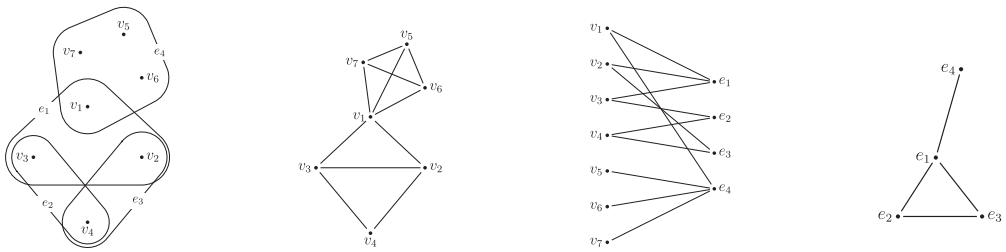
$$\deg(v) = \sum_{e \in \mathcal{E}} \mathbf{H}(v,e) \quad \text{and} \quad \delta(e) = \sum_{v \in \mathcal{V}} \mathbf{H}(v,e),$$

respectively. $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ indicate their corresponding diagonal matrix. When all hyperedges have the same degree k, i.e., $\delta(e) = k \; \forall e \in \mathcal{E}$, we say that H is a **k-uniform** hypergraph. k-uniform hypergraphs also have a tensor representation [89, 180]. As for classical networks, both nodes and hyperedges may be of more than one type; in such cases, the hypergraph is **heterogeneous**.

The **dual** H^* of a hypergraph H is the hypergraph constructed by swapping the roles of nodes and hyperedges, i.e., $H^* = (\mathcal{V}^*, \mathcal{E}^*)$, where $\mathcal{V}^* \equiv \{i \mid e_i \in \mathcal{E}\}$ and $\mathcal{E}^* \equiv \{\{i \mid v \in e_i \in \mathcal{E}\} \mid v \in \mathcal{V}\}$.

In a **weighted** hypergraph, denoted by a tuple $H = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, each hyperedge $e \in \mathcal{E}$ has a weight $w(e)$, representing the importance of that relation in the whole hypergraph. $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ denotes the diagonal matrix of the hyperedge weights, i.e., $\text{diag}(\mathbf{W}) = [w(e_1), w(e_2), \ldots, w(e_{|\mathcal{E}|})]$.

In a weighted hypergraph, the degree of a vertex v is defined as $\deg(v) = \sum_{e \in \mathcal{E}} w(e)\mathbf{H}(v,e)$. A non-weighted hypergraph can be seen as a special case of the weighted one where $\mathbf{W} = \mathbf{I}_n$.



(a) Hypergraph. (b) Two-section Graph (a.k.a clique graph). (c) Incidence graph (a.k.a. star expansion). (d) Line graph.

Fig. 3. Hypergraph-to-graph transformations.

3.2 Hypergraph-to-graph Transformations

Hypergraphs have usually been converted into a corresponding graph representation in the literature. Even though such transformation has often been preferred over hypergraphs, especially for computation convenience and the easiness of dealing with graphs rather than higher-order structures, this process may either bring a loss of information or introduce redundant vertices/edges regarding the original hypernetwork structure. The typical transformation of a hypergraph into a graph relies on its two-section, incidence, or line graph representation [24]. Figure 3 shows a toy hypergraph and its corresponding two-section, incidence, and line graphs.

Two-section graph. The *two-section* or *clique graph* of H is the graph, denoted with $[H]_2$, whose vertices are the vertices of H and where two distinct vertices form an edge if and only if they are in the same hyperedge of H (see Figure 3(b)). In other words, each hyperedge of H appears as a complete sub-graph in $[H]_2$.

The major drawback of using such a transformation is that clique graphs completely lose the notion of groups since pairwise connections substitute each high-order interaction. Consequently, we have a high probability of materializing interactions that did not exist in the original hypergraph. This intuitive concept of losing the notion of groups is formalized by the fact that different hypergraphs can be transformed in the same clique graph; hence, we cannot uniquely reconstruct the original hypergraph from its clique graph. Further, clique graphs can yield computational issues as each hyperedge of size k is transformed into $\frac{k\times(k-1)}{2}$ edges.

Incidence graph. The *incidence graph* or *star expansion* of H is the bipartite graph $I(H) = (\mathcal{V}, \mathcal{E}, \mathcal{E}')$, where $v \in \mathcal{V}$ and $e \in \mathcal{E}$ are adjacent if and only if $v \in e$, i.e., $\mathcal{E}' \equiv \{(v, e) \mid v \in e\}$ (see Figure 3(c)).

Bipartite graph representations effectively describe group interactions (see Figure 3(c)). In this model, one vertex set corresponds to the hypergraph's vertices, the other to the hyperedges. Hence, a link in this graph connects a vertex to the interactions—of arbitrary order—in which it takes part. However, bipartite graphs also have a critical shortcoming inherent in their structure. Vertices in the original system do not interact directly anymore as the interaction layer always mediates their relationship. This interaction structure implies that any measure or dynamic process defined on the bipartite representation must consider this additional complexity.

Line graph. The *line graph* of H is the graph $L(H) = (\mathcal{V}', \mathcal{E}')$ such that $\mathcal{V}' \equiv \mathcal{E}$, and $\{i, j\} \in \mathcal{E}', i \neq j \iff e_i \cap e_j \neq \emptyset$ (see Figure 3(d)). Essentially, each hyperedge is transformed into a node and an edge is added between two nodes if the corresponding hyperedges intersect in the original hypergraph.

As happens for clique graphs, distinct hypergraphs can have identical line graphs as they lose the information about the composition of each hyperedge, storing only whether two hyperedges intersect but not in which manner. Further, sparse hypergraphs can yield relatively dense line graphs as a vertex of degree d in the hypergraph yields $\binom{d}{2}$ edges in its line graph.

4 THE HYPERGRAPH REPRESENTATION LEARNING PROBLEM

This section formally introduces the hypergraph representation learning problem to then detail the problem setting in terms of input and output. Finally, it discusses the intrinsic challenges of learning latent representations of a hypergraph.

4.1 Problem Formulation

Definition 4.1[Hypergraph Embedding] For a given hypergraph $H = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of hyperedges, a hypergraph embedding is a mapping function $\Phi : \mathcal{V} \rightarrow \mathbb{R}^{|\mathcal{V}| \times d}$, where $d \ll |\mathcal{V}|$, such that Φ defines the latent representation (a.k.a. embedding) of each node $v \in \mathcal{V}$, which captures certain network topological information in $\mathcal{E}$.

In other words, given a hypergraph $H = (\mathcal{V}, \mathcal{E})$ and a predefined embedding dimension d, with $d \ll |\mathcal{V}|$, the problem of hypergraph representation learning (a.k.a. hypergraph embedding) is to map H into a d-dimensional vector space (a.k.a. latent space), in which the structural properties of H are preserved as much as possible. When features are attached to nodes/hyperedges, the learned latent representation should also encode such additional information. Following this definition, each hypergraph is represented as either a d-dimensional vector (for a whole hypergraph) or a set of d-dimensional vectors, with each vector representing the embedding of part of the hypergraph such as nodes, hyperedges, or substructures.

4.2 Problem Setting

In this section, we compare existing hypergraph representation learning literature from the perspective of the problem setting, describing different types of input/output and the specific characteristics of each setting.

4.2.1 Input Setting. In this survey, we analyze the hypergraph embedding input along six axes: the nature of the high-order relation, its directionality and size, the temporal dimension, whether nodes have attached additional information, and whether the hypergraph is converted into a graph. Figure 3 in the Supplemental Material shows the different types of hypergraphs, while Figure 4 outlines the input settings. Next, we introduce each category and summarize its specific characteristics.

Nature of the relation. As happens for graphs, hypergraphs can encode relations among nodes of one or more types. Similarly, hyperedges may represent a single or different kinds of possible interactions.

Homogeneous hypergraphs. Homogeneous hypergraphs represent the most trivial input setting, as both nodes and hyperedges belong to a single type. In this context, each hyperedge tells us that a given subset of nodes shares a common property or feature. Homogeneous hypergraphs have been widely used to model the most various user-item relations for recommendation purposes (e.g., [33, 84, 166, 181, 185]), citation networks for the link prediction task (e.g., [10, 48, 49, 86, 171]), and other relational data for classification problems (e.g., [88, 120, 131, 169, 194]). In the literature, it is also possible to find homogeneous hypergraphs built upon non-relational data [54], where, for instance, hyperedges represent vertex connections based on some notion of distance in the feature space. In these cases,

such structures are usually abstract, possibly multimodal features for object classification (e.g., [48, 103, 117, 160, 177]), traffic or passenger flow forecast [105, 154, 155], and gas or taxi demand [177].

Homogeneous hypergraphs are often employed in their weighted version to convey information about the importance of a relation through hyperedge weights (e.g., [48, 80, 103, 121, 134]). The intuition behind this choice is that the weights should drive the algorithm to be more accurate in learning the embedding of nodes within more important hyperedges.

The biggest challenge of learning a vector representation of a homogeneous hypergraph is to preserve its connectivity patterns in the latent space as only structural information is available. Typically, embedding methods defined over such structures are more general and can be re-used off the shelf without any particular tweak as the task they are defined for has no particular constraint.

Heterogeneous hypergraphs. Hypergraphs capture the heterogeneity of the underlying hypernetwork via nodes of different types or through nodes and hyperedges of multiple kinds. In the first case, all hyperedges semantically encode the same type of interaction among diverse entities. Most of the heterogeneous hypergraphs belong to this category, and they are usually employed to model relations among users, items, and domain-specific properties or actions for ranking/recommendation (e.g., [18, 33, 152, 179, 181]), link prediction (e.g., [147, 172, 175, 176, 191]), or classification tasks (e.g., [36, 147, 198]). Hypergraphs with heterogeneous nodes and hyperedges add further expressiveness as relations can be of more than one type. In such cases, hyperedges can represent different types of events [37, 61, 62]; documents, tags, and annotation relations [197]; users, songs, and tags interactions [91]; actions on social media [138]; or even components in chemical/mechanical processes [164].

There are two major challenges when dealing with the embedding of heterogeneous hypergraphs. The first challenge relates to how to effectively encode different types of nodes and relations to preserve structural and semantic properties. The second problem refers to a possible imbalance of objects of different types. Further, the more complex the heterogeneous hypergraph is, the more specific the representation learning method will be. This situation often leads to embedding algorithms that are strictly related to the application task and are tricky to generalize.

Directionality of the relation. This feature refers to whether a direct interaction exists between (groups of) nodes in the underlying hypernetwork. Hence, we can find both undirected and directed hypergraphs in the literature.

Undirected hypergraphs. By far, undirected hypergraphs represent the typical input setting in the task of hypergraph representation learning. In this case, dependencies between nodes or hyperedges are not considered.

Directed hypergraphs. Conversely to graphs, there is not a standard definition of directed hypergraphs, and the notion of direction may be applied either between hyperedges (i.e., set of nodes) [51] or between nodes within the same hyperedge [8]. Yadati et al. [172] and Gao et al. [53] follow the first interpretation, proposing an embedding algorithm working on directed hypergraphs based on the definition of Gallo et al. [51]. In this case, the head and the tail of the directed relation are embodied by two hyperedges. Luo et al. [105] and La Gatta et al. [91] exploit the second interpretation, grounding their embedding technique on the definition proposed by Ausiello et al. [8]. This time the directionality of the interaction is defined within a single hyperedge, which consists of two node sets (head and tail).

Liao et al. [97] propose a generalized definition of this concept, in which each directed hyperedge comprises a sequence of node sets.

Regardless of the definition of directionality used, the most critical challenge of learning a latent representation of a directed hypergraph is how to incorporate hierarchy and reachability to preserve the asymmetric transitivity in the embedding space [19, 118].

Size of the relation. This property refers to the cardinality of each hyperedge and is strictly related to the nature of the relations the hypergraph is modeling. Hyperedges can represent interactions between either a fixed or unbounded number of nodes.

k-uniform. In k-uniform hypergraphs, each hyperedge has cardinality k and usually models an existing interaction between k nodes of possible k different types. Put differently, each hyperedge represents an indecomposable relation between heterogeneous entities that cease to exist when one or more components disappear. In this case, hyperedges represent relations between user mobility data and friendships (linking social, semantic, temporal, and spatial information) [146, 175, 176] or interactions between users, items, and domain-specific properties, e.g., [33, 78, 89, 147, 151]. Sometimes, a single hypergraph encodes k-way relationships of different sizes and/or types, e.g., [146, 175, 176, 181, 197].

k-uniform hypergraphs introduce more constraints regarding how the domain of interest should be modeled, but, at the same time, they ease the definition of hyperedge-related tasks, such as link prediction (see Section 4.3).

Non-uniform. In non-uniform hypergraphs, each hyperedge encodes a relation among an arbitrary number of homogeneous or heterogeneous nodes. In contrast with k-uniform hypergraphs, a hyperedge can continue to exist even when one or more nodes are removed from the network.

Heterogeneous and non-uniform hypergraphs have been used to model event data [37, 61, 62] or, as in the previous case, interactions between users, items, and domain-specific properties (e.g., [18, 45, 138, 152, 191]). Homogeneous and non-uniform hypergraphs have been exploited to represent similarities or shared properties of relational and non-relational data (e.g., [103, 124, 131, 194]), citation networks (e.g., [10, 48, 74, 159, 185]), and several user-item relations (e.g., [33, 84, 95, 181, 185]).

Thanks to their flexibility, non-uniform hypergraphs represent the most common input setting; however, conversely to k-uniform hypergraphs, they introduce further complexity to hyperedge-related tasks (see Section 4.3).

Temporal dimension. With their structure, hypergraphs can capture either a static view or the dynamicity of the underlying hypernetwork.

Static. Static hypergraphs represent the most common input setting given the early age of this research field. Static hypergraphs can model either existing connections at a fixed moment (e.g., [146, 147, 175, 176]) or node interactions over time that are aggregated into a single static snapshot (e.g., [33, 62, 78, 171, 179]).

Dynamic. Real (hyper)networks are often characterized by a dynamic behavior, meaning that both nodes and (hyper)edges can be added or removed from the system or that labels and other properties can change over time [16]. In the literature, a dynamic hypergraph is usually represented by a sequence of static hypergraphs, i.e., $H = \langle H(t_0), \ldots, H(t_{T-1}) \rangle$, where $H(t_k) = (V(t_k), E(t_k))$ is a static hypergraph with timestamp t_k, with $k \in \{0, \ldots, T-1\}$, T is

the number of snapshots, $V(t_k)$ is the node set at timestamp t_k, and $E(t_k)$ is the hyperedge set including all edges within the period $[t_k, t_{k+1})$ [95, 152].

Temporal hypergraphs have been used to model users' preferences over time in recommendation tasks [70, 87, 95, 152], in user trust relationships for rumor detection [140], or for time series forecasting [134, 154, 155, 164, 177].

Developing an approach to embedding dynamic hypergraphs implies dealing with a series of challenges that arise when the temporal component comes into play [16]. Such methods should consider (1) how to model the time domain (discrete-time or continuous-time), (2) which dynamic behaviors have to be embedded, and (3) which temporal granularity will be represented in the vector space.

Node features. This category refers to whether additional semantic information is associated with the nodes as input to the embedding process.

Not attributed. In this setting, nodes convey only structural information through their connectivity patterns.

Attributed. Besides the structural patterns, nodes can carry additional information about their nature in the form of feature vectors. These vectors may encode information related to user (e.g., [72, 87]) or item features (e.g., [41, 70, 166]), pre-trained word embeddings [40], image spectral features (e.g., [103, 117, 141]), and values derived from sensors (e.g., [105, 154, 155]), to name a few examples.

Although the presence of extra attributes can boost the performance of the intended task [52], this auxiliary information may not be trivial to embed, especially if it is not in a vector form.

Transformation to graphs. In the literature, the most common approach to deal with the high-order nature of hypergraphs is to split the single relation encoded by a hyperedge into a set of pairwise interactions. As described in Section 3.2, hypergraphs can be transformed into either clique graphs, incidence graphs, or some intermediate representation. Considerations about the drawback of each transformation can be found in Section 3.2.

Clique graphs. Transforming a hypergraph into the corresponding clique graph means instantiating a direct interaction between each pair of nodes in a given hyperedge. Such connections can be materialized, for instance, when the hypergraph adjacency matrix is exploited to consider linking patterns (e.g., [23, 128, 131]), or used to compute the pairwise similarity of two embedding vectors of nodes within the same hyperedge (e.g., [37, 62, 146, 176]). The clique graph transformation is implicitly used by most of the hypergraph convolutional methods based on the Message Passing Framework as thoroughly described in Section 5.3.

Incidence graphs. In an incidence graph, hyperedges are represented as nodes, and the high-order relation of the nodes in a hyperedge is mediated by these special nodes via pairwise links.

From a computational perspective, incidence graphs let the information flow from nodes to hyperedges and back. For this reason, this graph transformation is implicitly used by the Message Passing techniques based on the two-stage update procedure (e.g., [7, 79, 136, 145]),[2] in which, in a single convolutional layer, the messages pass from nodes to the hyperedges, where they are aggregated, and then back from the hyperedges to the nodes (see Section 5.3).

[2] `Hyper-SAGE` [7] does not appear in our statistics because it is not peer-reviewed.

Line graphs. Line graphs capture the relationship between the edges of a hypergraph. This representation has been used to preserve the indecomposability of the high-order relations encoded by hyperedges in Bandyopadhyay et al. [15], which designed a hypergraph convolutional network relying on an implicit transformation to weighted line graphs. Xia et al. [166] also introduce a hypergraph transformation to a line graph to alleviate the sparsity problem in recommendation tasks. In LE [174], the line graph transformation is applied to leverage standard graph convolutional networks.

Other transformations. Other approaches convert the hypergraph into a modified form of the transformations described in Section 3.2 to better capture structural properties and use them as an implicit regularizer [171], alleviate the sensitivity toward the node distribution [123], or model domain-specific characteristics [33]. For instance, Pu and Faltings [123] construct a directed weighted line graph in which each hyperedge is replaced with two nodes (positive and negative), and an edge is added between positive and negative node pairs based on the same *ratio* of line graphs. Yadati et al. [171] propose a modified version of the classical star expansion. Also in this case, each hyperedge is replaced by two nodes, joined by an edge, and linked to all nodes originally contained in the corresponding hyperedge. A third example can be found in Chen et al. [33], where the authors define a revised version of the clique graph converting each hyperedge (of a 3-uniform hypergraph) into two pairwise interactions.

4.2.2 Output Setting. The output of hypergraph embedding techniques[3] is a (set of) low-dimensional vector(s) representing (part of) a hypergraph. The embedding output is strictly task-driven, and finding the most suitable embedding type is crucial to meeting the specific application's needs.

In this survey, we consider *node* and *hyperedge* as embedding output settings. The most common embedding output is node embedding, while only a few approaches propose hyperedge embedding. Currently, no method deals with whole-hypergraph or hypergraph substructure embedding (see Section 7 for more details).

It is worth clarifying that there is no one-to-one correspondence between the specific embedding output setting and the addressed task since node (resp. hyperedge) embeddings can be used to evaluate hyperedge (resp. node)-related tasks.

Node embedding. Node embedding represents each node as a vector in a low-dimensional space. The underlying idea is to learn similar latent representations for nodes that are close in the original hypergraph. In practice, this concept of closeness often refers to nodes included in the same hyperedge. When nodes have additional features attached, both structural and semantic closeness are considered in the embedding process. As the most common embedding output setting, node embeddings have been used in a variety of downstream hypergraph analytic tasks, such as clustering (e.g., [37, 131, 194]), classification (e.g., [120, 138, 169]), regression (e.g., [134, 154, 177]), link prediction (e.g., [45, 74, 191]), and recommendation (e.g., [152, 153, 185]). The complexity of defining a proper similarity metric heavily depends upon the properties of the input hypergraph.

Hyperedge embedding. Hyperedge embedding methods learn low-dimensional vector representations for the hyperedges. In contrast to graphs where each edge encodes a pairwise relationship, in this context, the hyperedge embedding vector needs to capture the interaction between an arbitrary number of nodes.

The most common approach is combining the embedding vectors of the nodes within the specific hyperedge, for instance, by summing [23], averaging [62], performing more

[3]We are considering the embedding outputs and not the output related to the specific task (e.g., classification, regression).

complex aggregations [37], or learning the aggregation function through a (deep) neural network [40, 43, 45, 53, 60, 63, 79, 120, 136, 145]. Other approaches consist in *translating* the problem of hyperedge embedding in the node domain by operating (1) on the dual hypergraph (so that the nodes of the new hypergraph represent the hyperedges of the original one) [74, 88] or (2) on the corresponding bipartite graph (in which each hyperedge is modeled as an additional node) [143].

In this survey, we consider falling under this category only those techniques that explicitly learn hypergraph embeddings. As expected, hyperedge embedding benefits edge-related tasks, such as link prediction [45, 74, 143] or link classification [120, 136]. Nonetheless, hyperedge embedding vectors are also exploited to capture contextual information to improve node-related tasks, like clustering [37], classification [37, 43, 62, 74, 140], and recommendation [145].

4.3 Challenges of Hypergraph Representation Learning

Obtaining an accurate representation of a hypergraph into low-dimensional spaces is not a trivial task [26, 31, 184]. As hypergraphs are a generalization of graphs, the following challenges are directly inherited by the graph representation learning problem. The first challenge lies in finding the optimal embedding dimension of the representation. Using a lower dimension is more resource-efficient and may also help reduce noise in the original network; on the other hand, critical information may be lost in the process. Conversely, a higher-dimension representation tends to preserve more information at the expense of storage requirement and computation time. Further, the proper dimension depends on the input hypergraph as well as the application domain. The second issue relates to choosing the right hypergraph property to embed, as it can be reflected by node features, link structures, or meta-data information. Again, determining which feature is the most suitable for the task is strictly domain-dependent. Such obstacles relate to preserving both the structure and the rich content that may be attached to the network elements [31, 184]. A third challenge resides in the data's nature: due to countless reasons—such as privacy or legal restrictions, among others—the problem of data sparsity may corrupt both the structure and the additional content of a network. This condition may thus lead to extra difficulties in discovering structural-level relatedness and semantic similarities between vertices not directly connected [184]. In the case of heterogeneous networks, the hypergraph representation learning task is even harder [42, 158]. In addition to the previous challenges, the core issue in this context is effectively fusing heterogeneous information to encode different types of entities and relations into latent spaces so that both structural and semantic properties are preserved. Capturing the inherent organization of such networks may require prior knowledge of the application domain to be included in the embedding process (for instance, in the design of meta-paths [60, 78]). This consideration leads to embedding techniques that are strictly application dependent and difficult to generalize and reuse in other contexts [158]. Nonetheless, embedding hypergraphs enforces to tackle two more issues deriving from the higher-order nature of these structures.

- *Capturing group relations*. Hypergraphs model many-to-many relationships among entities. Nonetheless, the existence of a hyperedge bonding of some nodes only tells us that they share a common property and does not necessarily imply direct interactions between them. For instance, a hypergraph can encode a character co-occurrence (hyper)network, in which each node is a movie character and each hyperedge is a movie scene (hence, all characters appearing in the same scene also occur in the same hyperedge) [4]. The fact that two characters belong to the same hyperedge only tells us they have appeared in the same scene, but not that they have interacted in some way. We can argue a similar observation for a heterogeneous and k-uniform hypergraph. Let us consider a hypergraph modeling a Location-based Social Network (LBSN) in which each hyperedge links a user's presence at

a point of interest (POI) at a specific time along with the semantic information about the user's activity there [146, 175] (i.e., each hyperedge is a four-element set). In this case, each hyperedge represents an indecomposable relation between four types of entities, which all relate to the same real-world event. Removing a single of these entities would cause the hyperedge to disappear. A critical challenge of hypergraph representation learning lies in capturing such high-order relations in the embedding process. Mathematical tools able to abstract such relationships are fundamental in achieving this goal.

- *Task definition.* The key feature of hypergraphs is that each hyperedge embodies a relation that can potentially connect more than two nodes. This structural characteristic implies that all tasks defined over relations (i.e., formalized on the edges of a graph)—such as link prediction or network reconstruction—need to be generalized.
  The critical issue is that the definition of such tasks needs to be adapted to the specific application context. For instance, let us focus on the link prediction problem. Even for newbies, it is straightforward to think what is the goal of this task when instantiated on a graph, namely predicting whether a given relation between two nodes (will) exists. When dealing with hypergraphs, this problem takes on different meanings based on the nature of the underlying hypernetwork. Heterogeneous and k-uniform hypergraphs usually model fixed-sized relations among a specific group of entities with different types. In this case, we are interested in finding whether a relation (a subset of nodes) linking a fixed number of different elements exists (e.g., [78, 147, 176]). In the case of non-uniform hypergraphs, the problem becomes harder: this time, there is no (a priori) constraint on the size of the hyperedge to predict (e.g., [45, 79, 169]). As a consequence, the computational complexity of the link prediction problem explodes as the number of different hyperedges is exponential (as it corresponds to the number of all possible subsets of a set).
  Another direct consequence deriving from the structural nature of hypergraphs is that these structures can be exploited to also study problems on sets (a hyperedge being a subset of nodes). For this reason, tasks borrowed from set theory can be easily adapted in this context. A good example is the task of *hyperedge completion* [136], deriving from the set expansion task [183]. This problem involves finding those nodes that may fit into an existing hyperedge, thus completing it. Clearly, there is no direct equivalence of this problem for graphs.

5 HYPERGRAPH REPRESENTATION LEARNING METHODS

Hypergraph representation learning aims to embed hypergraphs into a low-dimensional space while preserving the original hypernetwork properties as much as possible. We divide hypergraph embedding techniques into three macro categories: spectral learning methods, proximity-preserving methods, and (deep) neural-network-based methods. The main difference between these three families lies in the way the representation learning problem is approached.

- The spectral learning family is the pioneer in hypergraph representation learning. Although the very first works (e.g., [23, 128]) were not focused on the learning aspects, the basic concepts and ideas apply in most of the later works (e.g., [194]). Basically, spectral learning approaches define the hypergraph representation learning problem in terms of the decomposition of the Laplacian matrix (hence, the term "spectral") of the hypergraph (Section 5.1). The output of such decomposition (a.k.a. factorization) is the (vertex) embedding matrix, where the topological proximity of the embedding vectors is ensured by the nature of the factorization.
- Proximity-preserving methods rely on a more flexible approach to the problem: vertex proximity in the embedding space is measured in terms of a similarity function (e.g., cosine

similarity), and such embeddings are learned by optimizing an objective function defined to preserve the topological proximity (i.e., close vertices should be similar) along with other factors and constraints related to the given task. The flexibility of this family of techniques lies in the modularity of the overall approach (i.e., similarity function → objective function → optimization) that makes it easily adaptable to very specific tasks and/or input settings.

- (Deep) Neural-network-based hypergraph embedding, as the name suggests, is a family of techniques based on (deep) neural networks. As it will be clear later, nowadays, deep-learning-based approaches are predominant w.r.t. the other families of hypergraph learning techniques. This success is due to the well-known strengths of Deep Neural Networks (DNNs), like automatic feature learning, scalability, and generalization, to name a few. Akin to DNN for graphs, a significant role is played by hypergraph convolutions that allow the generalization of well-established neural models to work on arbitrarily structured hypergraphs. Arguably, (many) proximity-preserving methods can be implemented in terms of shallow neural networks; however, in this survey, we stick to the methodology described in the original papers, and thus we consider proximity-preserving methods not based on neural networks.

In the following, we describe each approach proposed in our taxonomy, discussing its motivation, the most representative methods following the given technique, and its pros and cons. Finally, we summarize and compare the hypergraph representation learning approaches. Figure 5 in the online Supplemental Material outlines this taxonomy. Application tasks using the methods covered in this section are discussed in Section 6.

Please refer to Appendix C.3 in the Supplemental Material for a discussion about the temporal distribution of the papers covered in this survey.

5.1 Spectral Representation Learning Methods

Motivation. Spectral hypergraph embedding (a.k.a. spectral representation learning) methods are historically the first representation learning methods for hypergraphs. Interestingly, as far as we know, the very first discussion about vector representation learning for nodes/hyperedges dates back to the 1980s [50], where the goal was to learn a 2D representation suitable for visualization. Despite the approach to the problem being somewhat unorthodox w.r.t. today's standards, the work in [50] captures the essence of spectral representation learning. Generally speaking, this methodology aims to learn a low-dimensional latent representation for vertices in such a way that "similar" vertices are close to each other in the learned (Euclidean) latent space. Although, in principle, vertices' similarity could be modeled in several ways, the vast majority of the spectral hypergraph embedding methods define it in terms of the number of common incident hyperedges. The name "spectral" stems from the usage of concepts related to spectral (hyper)graph theory [28, 38], specifically, the relationships between the eigenvectors and eigenvalues of the (hyper) graph Laplacian matrix. As we will see, the common denominator of this family of methods is the definition of the node embeddings' features as the eigenvectors corresponding to the minimal eigenvalues of the Laplacian matrix.

Methods. Before diving into the characteristics of each spectral embedding technique, we provide a general overview of the methodology.

5.1.1 Overview. Let $H(\mathcal{V}, \mathcal{E}, \mathbf{W})$ be a weighted hypergraph, and let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the matrix where each row $\mathbf{x}_i \in \mathbb{R}^d$, with $i \in [n]$, is the latent representation of the vertex $v_i \in \mathcal{V}$ that we aim to learn. Ideally, the learned vertices' representation should encode the structural information of the hypergraph; thus, similar vertices should be mapped onto nearby points in the latent space.

If we only consider information about the hypergraph structure, the similarity between two nodes $v_i, v_j \in \mathcal{V}$ can be expressed as the (normalized) sum of the weights of the common incident hyperedges. Given this notion of similarity, we can define an optimization problem whose objective is to minimize the weighted Euclidean distance between nodes sharing the same hyperedges. Formally:

$$\arg\min_{\mathbf{X}} \frac{1}{2} \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{e \in \mathcal{E}} [\![v_i \in e]\!] [\![v_j \in e]\!] \frac{w(e)}{\delta(e)} \left\| \frac{\mathbf{x}_i}{\sqrt{\kappa(v_i)}} - \frac{\mathbf{x}_j}{\sqrt{\kappa(v_j)}} \right\|^2 = tr(\mathbf{X} \underbrace{(\mathbf{I}_n - \mathbf{D}_v^{-\frac{1}{2}} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-\frac{1}{2}})}_{\mathbf{L}_{\text{Zhou}}} \mathbf{X}^\top) \quad (1)$$

subject to $\mathbf{X}^\top \mathbf{X} = \mathbf{I}_d$ to avoid trivial solutions or arbitrary scaling factors, and where $tr(\cdot)$ is the trace operator, i.e., $tr(\mathbf{A}) = \sum_i \mathbf{A}(i,i)$. According to Zhou et al. [194], Eqatuion (1) is the relaxed version of the (NP-complete) problem of finding the optimal normalized cut, whose minimum is obtained by setting as columns of $\mathbf{X}$ (i.e., the features in the latent space) the d orthonormal eigenvectors corresponding to the d smallest nonzero eigenvalues of the Laplacian $\mathbf{L}_{\text{Zhou}}$.

5.1.2 Structural-only Spectral Embedding. The hypergraph Laplacian plays a key role in the definition of the loss function because it encodes the concept of "closeness" between nodes in the hypergraph. As a consequence, different Laplacians lead to different embeddings. In fact, the main difference between the methods discussed in this section lies in how the hypergraph Laplacians are defined. To ease the exposition of the hypergraph spectral embedding methods, we define a unified framework for learning the optimal d-dimensional embedding $\mathbf{X}$. The framework is based on the seminal works of Bolla [23] and Zhou et al. [194], but it provides a factorized view of the hypergraph Laplacian, allowing instantiating the different techniques by setting those factors. Formally, the framework is defined as

$$\arg\min_{\mathbf{X}:\mathbf{X}^\top\mathbf{X}=\mathbf{I}_d} tr(\mathbf{X}^\top \underbrace{\mathbf{Z}\,(\mathbf{D}_v - \mathbf{S}_v)\,\mathbf{Z}^\top}_{\mathbf{L}} \mathbf{X}) = tr(\mathbf{X}^\top \mathbf{L} \mathbf{X}), \quad (2)$$

where $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the hypergraph Laplacian, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a normalization matrix, and $\mathbf{S}_v \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ can be interpreted as a similarity matrix between nodes. When not specified differently, we assume $\mathbf{S}_v = \mathbf{H}\mathbf{S}_e\mathbf{H}^\top$, where $\mathbf{S}_e \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is a function of the weight and/or cardinality of the hyperedges. So, the factor $(\mathbf{D}_v - \mathbf{S}_v)$ represents a generalization of the not normalized graph Laplacian (i.e., the graph Laplacian can be obtained with $\mathbf{S}_v = \mathbf{H}\mathbf{H}^\top$). The matrix $\mathbf{Z}$ acts as a Laplacian normalizer, and when it is not the identity (i.e., no normalization), it is equal to $\mathbf{D}_v^{-1/2}$. Normalized Laplacians are often preferred because the influence of hub nodes (i.e., nodes within many hyperedges) is reduced, and their definition is tied to the probability transition matrix.

By definition, $\mathbf{L}$ is positive semi-definite, and, as previously described, we know that the solution to Equation (2) can be computed using the eigendecomposition of $\mathbf{L}$ from standard results in linear algebra [20].

Equation (2) was first introduced by Bolla in [23], where it is instantiated to find the "minimal variance placement" of the vertices in the latent space, leading to a formulation with an unnormalized Laplacian, i.e., $\mathbf{Z} = \mathbf{I}_{|\mathcal{V}|}$ and $\mathbf{S}_e = \mathbf{D}_e^{-1}$. Bolla also showed that, given the optimal $\mathbf{X}^*$, the optimal hyperedge embeddings can be computed as $\mathbf{X}^*\mathbf{H}\mathbf{D}_e^{-1}$.

Zhou et al. [194] made a step forward by providing a general framework for spectral hypergraph partitioning. The proposed normalized hypergraph Laplacian (i.e., $\mathbf{L}_{\text{Zhou}}$) generalizes [23] by including hyperedge weights, i.e., $\mathbf{S}_e = \mathbf{W}\mathbf{D}_e^{-1}$, as well as node normalization, i.e., $\mathbf{Z} = \mathbf{D}_v^{-1/2}$. Closely related to Zhou's Laplacian are the ones proposed by Zhu et al. [197] (HHE [197]) and Pu and Faltings [123]. In HHE [197], the Laplacian corresponds to Zhou's Laplacian without the node normalization ($\mathbf{Z} = \mathbf{I}_{|\mathcal{V}|}$), while in Pu and Faltings, the hyperedge degree matrix is omitted in $\mathbf{S}_e$. Pu and Faltings also present a novel Laplacian matrix (that does not fit the framework in

Equation (2)) based on the hyperedge expansion so that the learning result is invariant to the distribution of vertices among hyperedges. This novel formulation seems to get clusters of nodes with larger margins, which could be beneficial in the case of clustering tasks.

In [124], Ren et al. state that Zhou's Laplacian is not ideal for vision problems. Specifically, they argue that $\mathbf{S}_e$ contains redundant information while the normalization is empirically ineffective. For this reason, they proposed a Laplacian analogous to the standard graph Laplacian, i.e., $\mathbf{S}_e = \frac{1}{2}\mathbf{I}_{|\mathcal{E}|}$ and $\mathbf{Z} = \sqrt{2}\mathbf{I}_{|\mathcal{V}|}$. Saito et al. [131] proposed a Laplacian strongly related to Zhou's where the node similarity matrix has the diagonal zeroed out, i.e., $\mathbf{S}_v \leftarrow \mathbf{S}_v - \text{diag}(\mathbf{S}_v)$. From a random walk standpoint, Saito's Laplacian is consistent with standard graph Laplacians, whereas Zhou's setting can be regarded as a lazy random walk with self-loops.

The direct connection with graphs is also present in Rodriguez [128], where the hypergraph Laplacian of $H(\mathcal{V}, \mathcal{E})$ corresponds to the Laplacian matrix of the weighted graph on $\mathcal{V}$, in which two vertices v, u are adjacent if they are adjacent in H, and the edge-weight is the number of edges in H containing both u and v, i.e., $|\mathcal{E}_u \cap \mathcal{E}_v|$. According to Rodriguez, this Laplacian has nice spectral properties; however, the impact in terms of node embeddings is unclear.

5.1.3 Spectral Embedding for Nodes with Additional Features. Despite its generality, Equation (2) assumes that nodes only carry structural information. However, there are many applications in which additional features can be attached to nodes. We can generalize the framework to include such cases and consider an initial node representation. Let $\mathbf{V} \in \mathbb{R}^{n \times f}$ be the matrix containing each node (on the rows) and its features (on the columns); then, we can rewrite Equation (2) as

$$\underset{\mathbf{X}:\mathbf{X}^\top\mathbf{V}^\top\mathbf{V}\mathbf{X}=\mathbf{I}_d}{\arg\min} \quad tr(\mathbf{X}^\top\mathbf{V}^\top\mathbf{Z}\,(\mathbf{D}_v - \mathbf{S}_v)\,\mathbf{Z}^\top\mathbf{V}\mathbf{X}) = tr(\mathbf{X}^\top\mathbf{V}^\top\mathbf{L}\mathbf{V}\mathbf{X}), \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{f \times d}$ is the projection matrix that we want to learn. So, differently from Equation (2), here we learn a linear transformation rather than the embedding that is obtained as $\mathbf{V}\mathbf{X} \in \mathbb{R}^{n \times d}$. It is worth noticing that Equation (3) generalizes Equation (2) and they are equivalent if $\mathbf{V}$ is the identity matrix with $f = n$.

Both the approaches proposed by Sun et al. in [141] and [137] fall under Equation (3), while the approaches presented by Luo et al. in [103] and [104] are closely related to Equation (3) but the optimized loss is different. Specifically, Lou et al. use Zhou's Laplacian, but embeddings are learned in a supervised fashion where the supervision is used to define an objective function that simultaneously ensures high intra-class compactness and high inter-class separability. Very similar approaches to Luo's are proposed by Yuan and Tang [182] and Huang et al. [80].

Remarks. As demonstrated by Agarwal et al. [1], all hypergraph Laplacians introduced in this survey do not consider the higher-order nature of the interactions encoded by hypergraphs as, in practice, those are manipulated as graphs. Precisely, with the appropriate weighting function, the Laplacians in [194, 197] correspond to the Laplacian of the hypergraph star expansion, while the Laplacians in [23, 128, 131] are equivalent to the Laplacian of the hypergraph clique expansion. Agarwal et al. further showed that, in a linear setting, these two transformations are equivalent.

These embedding methods benefit from spectral theory on hypergraphs, making their definition elegant and easy to implement since they are based on standard linear algebra operations. However, the overall spectral framework is not flexible enough to easily handle the injection of node and hyperedge heterogeneity, and it is not designed to deal with temporal dependencies. Such settings can be tackled in a naive way by constructing different hypergraph Laplacians based on the node/hyperedge types [197] or by building discrete hypergraph snapshots and learning node embeddings by solving the generalized eigenproblem (as for dynamic graphs [168]), respectively. Further, as in the case of graphs, spectral theory on hypergraphs does not straightforwardly

translate to directed hypergraphs (whichever their definition is, see Section 4.2.1), and solutions to make the adjacency matrix symmetrical would add further (computational) complexity to an already demanding problem [193]. In addition, spectral methods scale poorly with $|\mathcal{V}|$ as they require the in-memory storage of all the matrices involved in Equation (1), even though sparse representations (of sparse hypergraphs) may help alleviate this problem. Still, this requirement hugely hinders their applicability to very large hypergraphs. This scaling issue explains why most spectral embedding methods were proposed in early 2000, while nowadays, more scalable approaches are preferred, e.g., neural-network-based methods.

Table 1 in the online Supplemental Material (Appendix D) provides an overview of the spectral methods described in this section. We provide the implementation of most of these methods at https://github.com/alessant/HEE.

5.2 Proximity-preserving Methods

Motivation. Proximity-preserving (PP) algorithms aim to design a proper similarity function able to catch various structural and semantic information conveyed by the hypergraph. The similarity can be naturally computed in graphs based on first-order or second-order proximity information. While the former tells how similar two nodes are based on the weight of the link between them, the latter compares the similarity of the nodes' neighborhood structures [26]. When it comes to hypergraphs, a desirable property of the embedding is to simultaneously retain the proximity among the nodes in the same group, as a hyperedge encodes high-order relationships that are not necessarily meaningful when fractured into pairwise links [179].

Methods. Proximity-preserving and spectral embedding algorithms share the idea of retaining the nodes' topological closeness; however, the approach to the problem is very different. PP techniques are less theoretically grounded, but the methodology is much more flexible and easy to adapt to the specific task at hand.

Although it is impossible to provide a single sensible framework general enough to cover all the methods falling into this category, we can highlight the concepts and ideas shared among most, if not all, of them.

Similarity as a function of the dot-product. In the case of Euclidean latent spaces, the proximity/closeness of two vectors is commonly measured using the dot-product [26]. Based on this general idea, PP methods designed for hypergraphs generally aim to jointly optimize the tuple-wise (or n-wise) proximity of the nodes in a hyperedge via the dot-product (`HEBE(-PO/PE)` [61, 62], `FOBE` and `HOBE` [144])) or one of its variants, like the cosine similarity (`Lbsn2Vec` [175], `Lbsn2Vec++` [176], and `MSC-LBSN` [146]).

Negative sampling. Regardless of the specific proximity function used, optimizing the similarity of related nodes comes with the risk of overfitting. In order to avoid overfitting, PP methods need (at least) to consider negative hyperedges (i.e., hyperedges in $2^E \setminus \mathcal{E}$) by maximizing the distance of the incident nodes. Given the combinatorial explosion of the number of non-existing hyperedges, negative sampling represents the most common technique to solve this problem (`Lbsn2Vec(++)` [175], `HGE` [179], `HEBE(-PE/PO)` [61, 62], `FOBE`, and `HOBE` [144]).

Objective function. The objective function presents a similar structure across all PP methods. Specifically, we can identify three primary sub-objectives:

(1) maximization of the node embeddings' similarity for nodes that are topologically or semantically close to each other;
(2) minimization of the node embeddings' similarity for the negative samples (when present);

(3) maximization of a task-specific sub-objective that can be embedded into the objective in Equation (1). An example of such a sub-objective is the maximization of the *inter-event proximity* used in Event2Vec [37], which describes the pairwise proximity between hyperedges.

No regularization. Surprisingly, most of the PP methods do not employ any kind of regularization over the learned embeddings (with the only exceptions of MSC-LBSN [146] and HOBE [144]). For methods based on the cosine similarity, e.g., Lbsn2Vec(++) [175], the regularization may not be necessary since the magnitude of the embedding vectors does not impact the cosine of the angle between them. However, it is unclear whether the lack of regularization may lead to overfitting for methods based on the dot-product, like HEBE [61, 62], Event2Vec [37], FOBE [144], and HGE [179].

Despite being similar at a higher level, each PP technique is characterized by the optimization objective, usually optimized via Stochastic Gradient Descent, which shapes how the node/hyperedge embeddings are learned. The optimization criterion can take different forms like similarity maximization (e.g., Lbsn2Vec(++) [175], HGE [179], MSC-LBSN [146]), mean squared error (e.g., HOBE [144]), Kullback–Leibler divergence (e.g., FOBE [144], Event2Vec [37]), or Bayesian Personalized Ranking [126] (e.g., HEBE(-PO/PE) [61, 62]). Clearly, the loss function takes into account the characteristics of the task to solve.

PP methods also differ in how they transfer the high-order relation conveyed by a hyperedge and whether they explicitly learn a representation for the hyperedge itself. Almost all methods of this family maximize the embedding similarity of nodes within the same hyperedge by pairwise computing the cosine similarity (Lbsn2Vec(++) [175], MSC-LBSN [146]) or the dot-product (HEBE [61], Event2Vec [37], HOBE/FOBE [144]) between two embedding vectors. An exception is made by HGE [179], which treats a hyperedge as a single set of nodes. Specifically, HGE [179] generalizes the standard dot-product operation between two vectors to an indefinite number of vectors for computing the similarity score of the nodes in a hyperedge. However, this dot-product generalization, called multilinear map, is meaningful only from an algebraic perspective while having no geometric interpretation. Among all PP methods, only HEBE-PE [61, 62] and Event2Vec [37] also learn a latent representation for hyperedges. In HEBE-PE [61, 62], the hyperedge embedding is learned by maximizing the similarity between the hyperedge vector and the average representation of the nodes within that hyperedge. In Event2Vec [37], the latent hyperedge representation is given by combining via a weighted dot-product the embedding of the nodes within it.

Another important observation is that all PP methods but Event2vec [37] only consider the first-order proximity, which could be sub-optimal. In particular, Event2Vec [37] captures the second-order proximity by modeling hyperedge proximity. Inspired by Factorization Machines [125], the authors proposed a new operation combining all the interactions between pairs of embedding vectors in the same hyperedge. Then, hyperedge proximity is modeled with the dot-product between the embedding and the just described combination.

Remarks. Proximity-preserving methods share the same overall design: the definition of (1) a similarity measure as a function of the embeddings and (2) a criterion to optimize for preserving the node proximity and possibly task-specific goals. This simple design makes PP methods versatile enough to adapt to peculiar tasks easily by means of specifically crafted optimization (sub-)objectives. This elementary structure also allows to naturally handle heterogeneous, non-uniform, and directed hypergraphs. In particular, the notion of directionality can be encoded within the objective function. For instance, in the case of inter-hyperedge directed hypergraphs, the criterion to optimize could enforce the embedding of the tail node(s) to have some particular characteristics (e.g., being the centroid of the embedding of all nodes within a hyperedge). The injection

of dynamic information into the latent representation via a PP method deserves separate considerations. In this case, the dynamic setting would make the overall method's structure harder to design, consequently conflicting with the underlying spirit of these techniques, which tend to be straightforward and efficient. Currently, no PP method is defined over directed or dynamic hypergraphs.

In contrast with deep representation learning (Section 5.3), we can define these methods as shallow because they optimize a unique embedding vector (i.e., a single layer of abstraction) for each node/hyperedge. These approaches are inherently transductive, which prevents them from being used in inductive applications. A drawback of most of these methods is their lack of capturing structural hypergraph information besides single-hop neighbors. Moreover, since similarity is usually defined as a function of the dot-product, these approaches may fail to model the potential non-linear relationship between nodes/hyperedges. Finally, PP methods on hypergraphs often treat the hypernetwork as its clique expansion counterpart, which may lose the benefit of modeling the problem in terms of high-order interactions in the first place. Table 2 in the online Supplemental Material (Appendix E) summarizes the methods described in this section.

5.3 (Deep) Neural Network Models

Motivation. Deep Learning (DL) has gained remarkable impetus in speech, language, and visual detection systems [55]. This success drove the research community to either directly apply DL models from other fields on graphs or design novel neural network models for specifically embedding graph-shaped data [26]. Hypergraph neural networks (HNNs) stem from graph NNs (GNNs), which currently embody the state-of-the-art in graph representation learning [161]. For this reason, the motivation and intuition behind GNNs also apply to hypergraph-like input data. (Deep) HNNs have been utilized to capture the high-order non-linear relationships between the nodes of a hypergraph without requiring hand-crafted features (i.e., providing end-to-end solutions). Furthermore, state-of-the-art NN frameworks (e.g., Pytorch, Keras, TensorFlow) leverage hardware acceleration (e.g., GPUs) to speed up the computation, making it possible to train very deep, and thus powerful, models efficiently.

Methods. Unlike (most of the) non-neural embedding techniques, the (deep) HNN framework requires having node features $\mathbf{x}_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$ (usually arranged into a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) as input to the model. When node features are not available, these can be initialized using a (parametric) encoding (e.g., [172, 191]), one-hot encoding (e.g., [40, 70, 88]), or ad hoc initializations (e.g., [34, 60, 110, 172]). Such initial representation is then fed to a (deep) neural network that learns the node (hyperedge/hypergraph) representation either in an end-to-end fashion or in a two-step procedure. In the former case, the learning process is guided by the specific task; thus, the embeddings are optimized (according to the loss function) to solve the problem at hand better. In the latter case, the embeddings are learned in an unsupervised fashion or by solving a fake task (e.g., [36, 74, 78]). Then, such embeddings are fed to another model, e.g., a classifier that may not be a neural network, to solve the task (e.g., [120, 191]).

In the following, we focus our description on the parts of the NN that explicitly or implicitly learn the latent hypergraph representation, giving less importance to the readout layers.

5.3.1 Hypergraph Convolutional Networks. Graph convolution networks are by far the most popular technique for learning graph embedding. The convolution operator on (hyper)graphs generalizes the operation of convolution from grid data to graph data [161] and can be elegantly motivated using the theory of (hyper)graph signal processing as a generalization of Euclidean convolutions to non-Euclidean data.

In the literature, (hyper)graph convolutions are usually classified between spatial and spectral based on how the convolution operator is defined [161, 192]. On the one hand, spectral graph convolution uses the Fourier transform to transform the graph signal to the spectral domain, where it carries out the convolution operation. On the other hand, spatial graph convolution aggregates the node features from the perspective of the spatial domain [187]. However, as thoroughly shown in [13, 14], this distinction is becoming less and less clear since many graph convolutions can be defined in both spectral and spatial domains. Based on this consideration, we prefer to avoid such a blurry categorization in our survey in favor of a more general framework encompassing almost all convolutional methods while still giving some space to peculiar spectral convolutions.

As Hamilton et al. [65] show, graph convolutions are a particular case of the more general and reader-friendly Message-passing Framework (MPF), whose underlying intuition is straightforward. Given an initial hypergraph representation (e.g., nodes' feature matrix $\mathbf{X}^{(0)}$), the MPF iteratively updates it according to the following process. At each iteration, every node aggregates information from its local neighborhood, and, as the iterations progress, each node embedding contains more and more information from further reaches of the hypergraph. Hence, node embeddings encode two-fold knowledge: structural and feature-based information, both deriving from iteratively gathering neighbors' representations according to the hypergraph structure. Each message-passing step is represented as a layer of a (deep) hypergraph convolutional NN (convHNN), which may act as input to the next layer.

Conversely to graphs where every edge simply connects two nodes, hyperedges in hypergraphs encode many-to-many interactions among a set of two or more nodes. Thus, to properly harness the high-order nature of hypergraphs, the propagation of the information over nodes should consider such richer relations. This intuition can be accomplished using a two-stage (a.k.a. two-level or two-step) update/aggregation process [7]:

$$\mathbf{x}_e^{(l+1)} = f_{\mathcal{V}\to\mathcal{E}}\left(\mathbf{x}_e^{(l)}, \{\mathbf{x}_v^{(l)}\}_{e\ni v}; \boldsymbol{\Theta}^{(l)}\right), \quad \mathbf{x}_v^{(l+1)} = f_{\mathcal{E}\to\mathcal{V}}\left(\mathbf{x}_v^{(l)}, \{\mathbf{x}_e^{(l+1)}\}_{v\in e}; \boldsymbol{\Phi}^{(l)}\right), \tag{4}$$

where $\mathbf{x}_e^{(l)}, \mathbf{x}_v^{(l)}$ are the latent representations of hyperedges and nodes at layer l, respectively; $f_{\mathcal{V}\to\mathcal{E}}, f_{\mathcal{E}\to\mathcal{V}}$ are two (non-linear) parametric permutation-invariant aggregation functions known as *node-to-hyperedge* and *hyperedge-to-node* aggregation, respectively; and $\boldsymbol{\Theta}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ and $\boldsymbol{\Theta}^{(l)} \in \mathbb{R}^{d'_l \times d'_{l+1}}$ are the learnable convolutional parameters. As for standard NNs, parameters are learned via the optimization of objective functions specifically designed for the given task. The optimization is performed using backpropagation and gradient-descent-based algorithms like Stochastic Gradient Descent (SGD) and Adam [90]. For simplicity, we omit the input argument H and the potential set of hyper-parameters from Equation (4). Equation (4) is very similar to the one proposed by Chien et al. [35] (`AllSet` [35]), and it generalizes many existing works (`UNIGNN` [79], `HyperSaR` [145], `HyperGAT` [40], `HNHN` [43], `HGC-RNN` [177], [7]). Peculiar two-stage MP processes are the ones proposed in [60] and [63]. In [60], Guan et al. define $f_{\mathcal{V}\to\mathcal{E}}$ as a *hyperedge shrinking* function in which they select the most informative (like an attention mechanism) nodes to update the hyperedge representation. In [63], instead, one of the stages is fixed. Specifically, node embeddings are aggregated using a mean and are not updated in the MP phase. Node embeddings are learned beforehand using a Graph Convolutional NN, and their weighted mean is used as the initial hyperedge representations.

A slightly different two-stage MPF is proposed by Srinivasan et al. [136], where the nodes and the hyperedges are updated simultaneously (in Equation (4), the updates happen sequentially) and where the aggregation functions also consider the embeddings of the second order neighbors.

While only the few works cited above rely on the two-stage MPF, most of the hypergraph convolutional NNs usually resort to the hypergraph's (weighted) clique expansion, thus reducing

Equation (4) to a single-stage MPF (as for graphs) first introduced by Feng et al. [48] (HGNN). In this case, the tth (single-stage) hypergraph convolutional layer can be elegantly defined as

$$\mathbf{X}^{(l+1)} = \overbrace{\sigma(\underbrace{\mathbf{R}\mathbf{X}^{(l)}}_{\text{aggregate}}\mathbf{\Theta}^{(l)})}^{\text{update}}, \tag{5}$$

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is the so-called *reference operator* [57], usually instantiated as a "surrogate" of the adjacency matrix; $\mathbf{\Theta}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are the learnable convolutional parameters (with d_l the dimension of the latent representation at layer l); and σ is a non-linear activation function (e.g., sigmoid). It is worth noting that although the MPF is flexible enough to allow to send different messages across the neighbors, the convolutional layer in Equation (5) assumes that every node sends the same message to each of its neighbors. Moreover, since the update of the node embeddings happens directly, hyperedge embeddings are not explicitly learned in the process. However, one can learn hyperedge embeddings by resorting to the dual transformation and applying the MPF (EHGNN [88]). To reduce the two-section graph connectivity and speed up the learning process, Yadati et al. [171] ((Fast)HyperGCN) proposed replacing each hyperedge with an incomplete clique.

Equation (5) covers most of the hypergraph convolutional models (discussed later); however, there are exceptions that may only fit in Equation (4). For instance, H^{2}SeqRec [95] defines an ad hoc convolutional operation in the hyperbolic space to alleviate the sparsity issue in hypergraphs for recommendation tasks, while DHGNN [86] and KHNN [99] use a 1D convolution and a parametric aggregation function to better model discriminative information among nodes. In [170], Yadati proposed G-MPNN [170], a generalized MPF able to handle multi-relational ordered hypergraphs, and MPNN-R to handle recursive hypergraphs.

There are also methods that use a non-parametric aggregation function just to propagate information in the hypergraph and to initialize node representations, such as DHCN [166] and MHCN [181].

In the following, we discuss five convolutional design factors (i.e., reference operator, skip-connections, attention mechanism, gated update, and spectral convolution) that impact how the information is propagated and aggregated during the learning process. These design choices may be coupled together in the same network architecture.

Reference operator. In Equation (5), the matrix $\mathbf{R}$ describes how embeddings are aggregated. In its simplest form, $\mathbf{R}$ is exactly the adjacency matrix of the hypergraph clique expansion, defined as $\mathbf{H}\mathbf{H}^\top - \mathbf{D}_v$ (NHP [172], SHCN [33], HHNE [18], DHCN [166] HGNN$^+$ [53]). In this case, the aggregation function is the sum of the representations of the neighbor nodes.

However, a not normalized adjacency matrix may hinder the optimization process as node features may be in very different ranges and, further, feature values may increase indefinitely on deep networks. For these reasons, the adjacency matrix is commonly normalized. The row-normalized adjacency matrix becomes $\mathbf{R}_{\text{row}} = \mathbf{D}_v^{-1}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{W}\mathbf{H}^\top$ (HCHA [10], HHGR [185], GC-HGNN [121], and IHGNN [34] without the hyperedge-related matrices), while it has the form $\mathbf{R}_{\text{sym}} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{W}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}$ if symmetrically normalized (HGNN [48], GC-HGNN [121], DH-HGCN [68], HyperCTR [70], DualHGNN [159], DualHGCN [169], MHGNN [9], STHAN-SR [134], MultiHGNN [76], HGDD [119], STHGCN [155], F^{2}HNN [110], S^{2}HCN [109], AdaHGNN [160], HybridHGCN [77], HyperINF [87], MT-HGCN [154], HGNN$^+$ [53], HGNNA [41]).

This latter formulation is usually preferred since row normalization does not consider the connectivity of the neighbor nodes and, as such, gives the same relative importance to highly and sparsely connected nodes. It is worth noticing that both normalized versions also consider self-loops (i.e., $\mathbf{R}$ has a nonzero diagonal); thus, the representation of a given node at layer l is transformed and included in its representation at layer $l + 1$ (i.e., skip-connection). Node-level

normalization is also possible, and in such a case, $\mathbf{W}$ (DHCF [84], HAIN [15]) and/or $\mathbf{D}_e^{-1}$ (HHNE [18], MGCN [32], HyperRec [152], [164]) can be removed from the computation of $\mathbf{R}_{\text{sym/row}}$. The set of learnable parameters is shared among all nodes.

An interesting approach is devised by Liu et al. in HGCELM [100]. Specifically, the authors defined a single-layered random hypergraph convolution based on Equation (5) with $\mathbf{R}_{\text{sym}}$, where the parameters are initialized randomly and used to perform a random projection (inspired by extreme learning machines [73]). Such representations are then fed to a hypergraph convolutional regression layer to predict node labels.

In both HCCF [162] and SHT [163], the hypergraph topology is not fixed but jointly learned with the embeddings. To do so, the reference operator is parametric, and it also includes a nonlinear activation function (i.e., LeakyReLU [167]).

Skip-connections. Over-smoothing is a well-known problem of deep (hyper)graph convolutional networks [30]. Over-smoothing occurs when the information aggregated from the neighbors during the message passing starts to dominate the updated node representations. This phenomenon happens as the receptive field of the convolution grows exponentially with respect to the model depth. A possible solution to alleviate this issue is to use skip-connections (a.k.a., residual connections/layer), which try to directly preserve the node information from the previous round of message passing in the update. This concept can also be generalized by including all (or some of the) previous node representations. Hypergraph convolution can be integrated with generalized skip-connections. In this case, Equation (5) becomes

$$\mathbf{X}^{(l+1)} = \sigma\Big(\mathbf{R}\mathbf{X}^{(l)}\boldsymbol{\Theta}^{(l)} + \underbrace{\sum_{j=0}^{l} \mathbf{X}^{(j)}\boldsymbol{\Phi}^{(j)}}_{\text{skip-connections}}\Big), \tag{6}$$

where $\boldsymbol{\Phi}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are the skip-connection parameters.

If the skip-connection is non-parametric and involves only the last layer, i.e., $\boldsymbol{\Phi}^{(l)} \equiv \mathbf{I}_{d_l}$ and $\boldsymbol{\Phi}^{(<l)} \equiv \mathbf{0}$, it is called identity mapping, and it simply "pushes forward" the representation of the nodes at layer l (DHCF [84], HCHA [10]). If parametric, the skip-connection can either have a different set of parameters w.r.t. the convolution (DualHGCN [169], HAIN [15]) or share the same set of parameters, i.e., $\boldsymbol{\Theta}^{(l)} \equiv \boldsymbol{\Phi}^{(l)}$ with $\boldsymbol{\Phi}^{(<l)} \equiv \mathbf{0}$ (SHCN [33], HHNE [18], MGCN [32], HyperRec [152]). Note that if $\mathbf{R}$ has a non-zero diagonal (i.e., it considers self-loops), the skip-connection of the previous layer with shared parameters is already included in Equation (5) (e.g., HGNN [48]). The identity mapping can also be performed on the aggregated messages rather than the previous node representation(s), and it can be combined with the "standard" skip-connection as in ResHGNN and UniGCNII [79].

A particular case of the skip-connection is the initial residual connection that pushes forward only the initial representation instead of the representation of the previous layer, i.e., $\boldsymbol{\Phi}^{(>0)} \equiv \mathbf{0}$. For instance, in ResHGNN [76] and HyperINF [87], the residual connection is not parameterized but weighted, specifically, $\boldsymbol{\Phi}^{0} \equiv \alpha_l\mathbf{I}$ and $\boldsymbol{\Theta}^{(l)} \equiv (1-\alpha_l)\mathbf{I}$, where α_l is a hyperparameter that balances the contributions of the previous representations. An alternative to the additive skip-connection is the multiplicative-skip connection that aims to learn a nonlinear gate to modulate the embeddings at a feature-wise granularity through dimension re-weighting (MHCN [181]).

Attention mechanism. The aggregation function in Equation (5) represents the (normalized) average overall neighbor nodes. Even though straightforward, this choice might be sub-optimal as it gives all neighbors the same importance. A natural way to overcome this limitation is to weigh each neighbor differently. This solution is the basic idea behind the attention mechanism, i.e., a

neural network that learns how to weight the neighbor nodes, first introduced for graphs by Velickovic et al. [150]. In a nutshell, (hyper)graph attention is a particular case of (H)GNN in which the message sent to the neighbors is not the same.

The literature offers different types of attention mechanisms, and the most popular in hypergraphs are additive attention (DHCN [166], DHGNN [86], DualHGNN [159], HNN [139], HGC-RNN [177]) and multiplicative attention (HyperGAT [40], SHARE [153], [164], HGNNA [41], HGAT [29], DH-HGCN [68], [140]). Although less popular on hypergraphs, it is also possible to add multiple attention "heads" (HCHA [10], DHAT [105], SHT [163]), where each one computes k different attention weights on independently parameterized attention layers. Eventually, the aggregated messages using the k attentions are linearly combined. Attention mechanisms can also help combine node embeddings in tasks like group recommendation (HHGR [185], HCR [85]) or handling temporal sequences (e.g., STHAN-(S)R [134], (D)STHGCN [155]). Recently, thanks to the success of the transformer architecture [149], self-attention mechanisms are starting to appear in convHNN (DHGNN [86], HOT [89], SSF [72], Hyper-SAGNN [191], HyperRec [152]), as widely used to combine sequential node/hyperedge embeddings for task-specific purposes (Higashi [190], DualHGNN [159], DHAT [105], HyperTeNet [151], H^2SeqRec [95]).

Gated updates. Before the advent of transformer architectures, recurrent neural networks were the standard tool to deal with sequential inputs. In hypergraphs, the sequential inputs can refer to (1) node level, i.e., node features change over time, or (2) structural level, i.e., the hypergraph topology changes over time. In the node-level case, the neighbors' information contains not only their current hidden state but also a completely new observation. In this setting, the main mechanisms used are Gated Recurrent Units (GRUs) ([164], HGC-RNN [177]), Gated GNN [96] ([156]) and Gated Linear Units (GLUs) stacked on top of convolutional layers (MT-HGCN [154], (D)STHGCN [155]). As far as we know, there are no relevant hypergraph embedding techniques dealing with structural sequential input.

Spectral convolution. Spectral hypergraph convolution relies on the spectral (hyper)graph theory, and it is the basis of the very first convHNN method proposed by Feng et al. [48] (HGNN). As previously discussed, HGNN can also be interpreted as a special case of the MPF (see Equation (5)); however, it could not be straightforward to directly map a particular spectral hypergraph convolution method in the MPF, though it could be argued that it is possible [13].

The main concept behind spectral hypergraph convolution is the connection between the Fourier transform and the Laplace operator. We can define hypergraph Fourier transforms of a hypergraph signal through the eigendecomposition of the hypergraph Laplacian, where the eigenvalues represent hypergraph frequencies (i.e., the spectrum of the hypergraph), and eigenvectors denote frequency components (i.e., hypergraph Fourier basis). Then the convolution in the spectral domain is defined via point-wise products in the transformed Fourier space (HpLapGCN [49], pLapHGNN [108]).

Instead of Fourier basis, it is possible to employ Wavelet basis as in HGWNN [117] and HWNN [138], where the hypergraph wavelet-transform projects graph signal from the vertex domain onto the spectral domain.

5.3.2 Random Walk-based Approaches. Random walk (RW) is a common alternative to convolution to encode the concept of closeness. A nice and commonly used property of an RW on a (hyper)graph is its analogy with sentences in a text: a path is a sequence of nodes (and hyperedges) like a phrase is a sequence of words. This similitude allows leveraging the distributional hypothesis in linguistics, stating that words that occur in the same contexts tend to have similar meanings [69]. Likewise, nodes sharing the same structural context should be close to each other

in the latent space. This analogy enables the application of natural language processing techniques to (hyper)graphs (DeepWalk [122]).

In this context, an established approach follows the two-step framework introduced in DeepWalk: First, walks in the hypernetwork capture the structural context, and then node embeddings are learned via a natural language processing model, usually Skip-gram and CBOW [114]. Finally, the so-obtained node representations can be the input either of a readout layer (Hyper2Vec/NHNE [74, 75], Hyper-gram [78], DHHE [36], HEMR [91]) or of another neural network module that fine-tunes the embeddings (DHE [120], Hyper-SAGNN [191]). In [198] (HRSC), the just described method (using CBOW) is applied on the two-section graph, which, however, fails to learn the representation of the hyperedges. Thus, to address this issue, the authors proposed a novel negative sampling-based set constraint objective function.

The nature of hypergraphs allows to inherently generalize existing graph RW approaches to these structures, but their richer semantics requires defining hypergraph-specific strategies. For instance, resembling the exploration-exploitation strategy proposed by node2vec [59], Hyper2Vec [75] exploits a degree-biased second-order RW to sample paths in a homogeneous hypergraph. A similar strategy is also proposed in Hyper-SAGNN [191]. Conversely, the hyper-path-based RW approach of Hyper-gram [78] captures the idea that hyperedges may encode indecomposable relationships in a heterogeneous hypergraph. On the same line, DHE [120] also proposes a new RW model to acquire co-member information in each hyperedge.

5.3.3 Encoder-based Approaches. When it comes to learning low-dimensional embeddings, the encoder-decoder architecture is the go-to approach in machine learning. However, this type of architecture may fail in capturing structural information encoded by hypergraph-like structured data. For this reason, these networks are usually included as a sub-module of bigger networks. For instance, Hyper-SAGNN [191] uses this approach to initialize node representations, while DHNE [147] uses an autoencoder to learn a first latent representation for the nodes, then fine-tune using a neural network specifically designed to preserve the second-order proximity. More "sophisticated" approaches make use of Variational Autoencoders (HeteHG-VAE [45]) and DeepSets (DHE [120]). In general, when coupled with specific decoders, these methods are mostly used in task-specific hypergraph embeddings (e.g., HeteHG-VAE [45], Event2Vec [37], HNN-HM [97]).

Remarks. Thanks to their versatility, HNNs can be easily adapted to a broad spectrum of tasks, such as (multi-class) classification (e.g., HpLapGCN [49], HyperGCN [171], DualHGNN [159], DHE [120]), regression (e.g., [164], HGC-RNN [177]), link prediction (e.g., HeteHG-VAE [45], DHCF [84], DHNE [147]), and recommendation (e.g., HHGR [185], HyperRec [152], SHARE [153]). The modularity of NN methods also eases the fusion of different input channels. For instance, using the dual of a hypergraph makes it possible to have the same architecture designed for hypergraph node embeddings to learn hyperedge embeddings (e.g., NHNE [74]). In general, multi-channel input can be helpful in domain-specific tasks, for example, when dealing with heterogeneous hypergraphs (e.g., MHCN [181], DHCF [84]). HNNs can also manage dynamic hypergraphs [164] or, in general, tasks that require modeling sequential information (e.g., H^2SeqRec [95]).

Moreover, HNNs allow performing transfer learning and reusing well-established architectures on various tasks by simply designing the readout layer(s) or a task-specific loss function. The typical learning setting of HNNs is (semi-)supervised transductive learning with only few exceptions like DHE [120], G-MPNN [170], AllSet [35], and HGAT [29] that can be used in inductive settings.

As for (deep) NNs in general, HNNs require much data to be adequately trained, and the hyperparameter tuning can be costly. Another drawback of HNNs, especially on very deep architectures,

is that the training phase may require high computational power (e.g., GPUs, TPUs), hampering the applicability on edge devices.

Among all the HNNs techniques, convHNNs are nowadays predominant in the literature thanks to their superior performance. This trend is evident from Table 3 in the online Supplemental Material (Appendix F): both encoder-based and random-walk-based methods have almost disappeared since 2020. The success of such a methodology is mainly due to the reuse of the knowledge coming from the graph representation learning literature. Akin to GNNs, these approaches learn the node representations that can then be aggregated to learn the latent representations of hyperedges (e.g., `NHP` [172]) or the whole hypergraph (e.g., [164]). Recently the trend seems to be drifting toward convolutional methods specifically designed for HGs (e.g., [136], `AllSet` [35]), where the hyperedge embeddings are explicitly learned.

However, convHNNs are generally more demanding in terms of computational requirements than encoder-based and random-walk-based methods; thus, such approaches should be preferred in case of limited computational capabilities. Further, the clique-expansion transformation used by many convolutional approaches is a possible limitation since the high-order nature of the relations is only indirectly exploited in the learning process. An important strength of random-walk-based methods is the universality of the approach that can be (almost) seamlessly applied to any hypergraphs. Encoder-based methods, instead, may require some effort in the design of the encoder/decoder architecture. Still, when combined together, random-walk- and encoder-based methods can achieve state-of-the-art performance, e.g., `Hyper-SAGNN` [191] and `DHE` [120].

Task-wise, we do not notice any particular trend: all (deep) neural network approaches have been applied successfully on classification, link prediction, and recommendation tasks. Nonetheless, when dealing with dynamic or multi-channel hypergraphs, convHNNs are preferable since they can easily be designed to handle recurrent or multi-channel inputs.

5.4 Comparison of Hypergraph Representation Learning Models

All methods discussed in the previous sections have distinctive characteristics (see Table 2), making them fit specific application domains and constraints. In general, we can observe that *spectral-based* embedding algorithms have been little used in real-world, large-scale applications due to their scalability issues. Nonetheless, these methods represent a viable choice when dealing with small datasets. The simple design of *proximity-preserving* approaches makes them particularly suitable when we aim to take advantage of prior domain knowledge that can easily embed in the learning procedure via the design of specific similarity functions. Akin to spectral methods, PP methods work better on small to medium-scale datasets. Currently, *(Deep) NN* methods embody the state of the art in hypergraph representation learning, and the vast majority of embedding methods belong to this category. This trend directly derives from the high-quality performance these methods achieve and the possibility of performing end-to-end learning. Further, the flexibility of the MPF allows capturing the high-order relationships encoded by each hyperedge. Hypergraph NNs scale better than the other families of HG embedding techniques and, thus, can be applied to large-scale datasets. However, the training of particularly deep networks may be hardware-demanding.

6 LEARNING TASKS AND APPLICATIONS

Hypergraph representation learning is functional to solve a plethora of graph analytic tasks as the learned latent representations allow the efficient execution (in both time and space) of traditional vector-based algorithms. Based on consolidated surveys on graph representation learning [16, 25], in this work, we classify existing applications in which hypergraph embedding came into play according to whether they focused on nodes or hyperedges. It is worth noting that we only match

Table 2. Comparison of Hypergraph Representation Learning Techniques

Embedding Method	Pros	Cons
Spectral learning	(1) Consider global structure; (2) Solid theoretical foundation; (3) Easy to implement and parallelize.	(1) Cannot capture high-order interactions (transformation to clique graphs); (2) Scalability issues (in terms of computational time and storage).
Proximity-preserving	(1) Simple and versatile design based on node similarity maximization; (2) Easy to adapt to specific tasks.	(1) May not model high-order non-linear relations beyond single-hop neighbors (transformation to clique graphs); (2) May suffer from scalability issues.
(Deep) Neural networks	(1) Capture high-order and non-linear relations; (2) High versatility; (3) Can easily embed both structural and additional information; (4) Transfer learning capabilities.	(1) Hardware demanding training; (2) High computational cost; (3) May require an extensive phase of hyper-parameter tuning.

each work with the task explicitly reported in the corresponding article. Please refer to Table 4 (Appendix G) for a list of the most commonly used datasets across the reviewed works.

6.1 Node-related Learning Tasks and Applications

In this section, we describe the role of hypergraph representation learning in relation to classical learning tasks defined on the nodes of a hypergraph, e.g., classification, clustering, and recommendation. For each task, we further describe the application domain(s) where it has been exploited.

6.1.1 Node Classification. Node classification is a supervised task in which the goal is to assign the correct label to each unlabeled node in the hypergraph based on the patterns learned from the other already labeled nodes (supervision). Node classification is one of the most common tasks discussed in the hypergraph embedding literature and finds wide application in (possibly multi-label) event (HEBE-(PE/PO) [61, 62], Event2Vec [37]), movie (e.g., DHNE [147], HRSC [198], DualHGNN [159]), authorship (e.g., HyperGCN [171], Hyper2Vec/NHNE [74, 75], HWNN [138]), citation (e.g., DHGNN [86], HCHA [10], UNIGNN [79], LE [174]), image (e.g., SSHGDA [103], AdaHGNN [160], HGWNN [117]), and item (e.g., DualHGCN [169], AllSet [35], DualHGCN [169]) classification, among others (e.g., HOT [89], EHGNN [88], DHE [120]).

Node classification is applied in a (semi-)supervised setting, and, in the literature, there are two main approaches to tackle it. The first approach consists of a three-step sequential procedure, in which (1) the embedding method first learns the latent representation of the nodes, (2) then an existing classifier is trained on the labeled instances, and (3) finally, the trained classifier predicts the class of an unlabeled node given its embedding. SVM (e.g., DHNE [147], HEBE-(PE/PO) [61, 62], SSHGDA [103]), logistic regression (e.g., HEBE-(PE/PO) [61, 62], Hyper2Vec/NHNE [74, 75], Event2Vec [37]), and k-nearest neighbors (e.g., SSHGDA [103]) are the most commonly used classifiers. The second approach relates to (deep) neural networks. Specifically, the task is solved in an end-to-end fashion where the first part of the network explicitly or implicitly learns the node embeddings, while the readout layer(s) solve the specific classification (e.g., AllSet [35], HOT [89], HNHN [43]).

6.1.2 Node Clustering. The goal of node clustering is to group similar nodes together in a way that nodes in the same cluster are more similar to each other than nodes in other groups. Node clustering is an example of an unsupervised learning setting, and it is usually applied when no node labels are available to find affinities between groups of nodes. The classical workflow is first to learn the low-dimensional representations of the nodes to then apply the desired traditional

clustering algorithm on the latent vectors. All works included in this survey (Zhou et al. [194], Ren et al. [124], Saito et al. [131], Event2Vec [37]) adopt k-means as the clustering algorithm. Specifically, Event2Vec [37] exploits this task to test the expressiveness of the learned node embeddings in capturing within-event and cross-event relationships in event data, while Ren et al. address the problem of clustering object views.

6.1.3 Node Recommendation. The task of node recommendation consists in finding the best nodes of interest (e.g., items) for a given node (e.g., user) based on certain criteria [44, 127]. In this context, hypergraphs embody a valuable tool to structurally encode the inherent high-order relationships that may arise beyond the classical user-item relation. For instance, hypergraphs can explicitly model additional information attached to the user-item interaction, such as emotions (e.g., HGE [179], SHCN [33]), item categories or properties (e.g., HHE [197], HEMR [91], HyperCTR [70]), search query (e.g., HyperSAR [145], IHGNN [34]), purchase history (e.g., HyperREC [152], H^{2}SeqRec [95]), social relationships (e.g., MHCN [181]), user sessions (e.g., SHARE [153], DHCN [166], GC-HGNN [121]), or user/item groups (e.g., DHCF [84], HHGR [185], HCR [85]). In general, recommender systems based on hypergraphs have been used for document/book (e.g., HHE [197], H^{2}SeqRec [95], KHNN [99]), movie (e.g., HGE [179], DHCF [84], HHGR [185]), product (e.g., HyperRec [152], SHCN [33], SHARE [153]), song (e.g., FOBE/HOBE [143], MHCN [181], HEMR [91]), video (e.g., HyperCTR [70]), business/place (e.g., HyperSAR [145], MHCN [181], HHGR [185], DH-HGCN [68]), and news (e.g., DHCF [84]) recommendations, and others (e.g., GC-HGNN [121], HCR [85], [156]).

6.2 Hyperedge-related Tasks and Applications

Next, we discuss the role of hypergraph representation learning concerning the link prediction task, its variations, and associated application domain(s).

6.2.1 Link Prediction. In traditional graphs, the link prediction task involves inferring new relationships or still unknown interactions between pairs of entities based on their properties and the currently observed links [98]. In hypergraphs, the problem of hyperlink prediction involves predicting missing hyperedges from the set $2^{|\mathcal{V}|} \setminus \mathcal{E}$ based on the current set of observed hyperedges $\mathcal{E}$. When dealing with k-uniform hypergraphs, the size of the hyperedge to predict is bounded by the nature of the modeled relation [89, 147, 191]. However, in the more general setting of non-uniform hypergraphs, the variable cardinality of a hyperedge makes link prediction methods defined for graphs infeasible as they are based on exactly two input features, i.e., those of the two vertices potentially forming a link [113, 172].

To overcome the difficulties mentioned above, all methods devised for hyperlink prediction heavily rely on negative sampling techniques to discard not meaningful relations in both training and evaluation phases (e.g., MSC-LBSN [146], NHP [172]). Then, the existence of a given hyperedge is based on a similarity score evaluated over pairs of embedding vectors corresponding to the nodes that should be contained in the relation. The similarity score can be computed with either some similarity measures, such as Euclidean distance and cosine similarity (e.g., NHNE [74], LBSN2Vec [175, 176]), or a more complex edge classifier, such as logistic regression. In this last case, the link prediction task is treated as a classification problem where the target class label indicates the presence or absence of a link between a pair of nodes (e.g., G-MPNN [170], HNN [139], DualHGCN [169]).

In the literature, this task has been applied to detect ⟨user, location, activity⟩, ⟨user, movie, tag⟩, ⟨user, drug, reaction⟩, and ⟨synset, relation type, synset⟩ relations (e.g., DHNE [147], HRSC [198], Hyper-SAGNN [191], HOT [89]). Other application domains relate to predicting hyperlinks in non-uniform (possibly) heterogeneous collaboration (e.g., HeteHG-VAE [45], NHP [172], NHNE [74]),

user-movie interest (e.g., HeteHG-VAE [45], NHNE [74]), user-item (e.g., DualHGCN [169], [156], HNN [139]), and chemical/drug reaction (e.g., HHNE [18], HGDD [119]) networks.

6.2.2 Network Reconstruction. The task of network reconstruction can be seen as a particular case of link prediction, in which all hyperedges of the original hypernetwork need to be inferred (DHNE [147], Hyper-gram [78]). Another variation of the problem, a.k.a. hyperedge expansion, has been proposed by Srinivasan et al. [136] and is based on set theory. In this case, the task consists of predicting the missing nodes (elements) from a given hyperedge (set).

6.3 Other Applications

In the following, we provide a brief description of other relevant applications.

6.3.1 Time-series Forecasting. Time-series forecasting is a subclass of regression problems, and, specifically, it is the task of fitting a model to historical, timestamped data in order to predict future values. This umbrella term includes several domain-specific tasks in which hypergraphs have been exploited, such as traffic prediction (MT-HGCN [154], DHAT [105]), passenger flow prediction (STHGCN [155]), gas and taxi demand (HGC-RNN [177]), and stock selection (STHAN-SR [134]). In this context, homogeneous and non-uniform hypergraphs model spatio-temporal information.

6.3.2 Visualization. Visualizing the outcomes of a hypergraph representation learning algorithm embodies a strong demonstration of whether the devised embedding method is preserving the desired characteristic of the input network. After learning the low-dimensional latent vectors, those are further projected into a two-dimensional space, usually via t-SNE [148], the state-of-the-art tool to visualize high-dimensional data. Another way to go, common for spectral-based methods, is to plot each node representation considering the two or three smallest eigenvectors of the Laplacian matrix [194]. In the case of node classification or, when possible, clustering tasks, each node category is colored differently: in this way, it is easy to grasp whether nodes belonging to the same category or sharing the same characteristics are embedded close to each other. Some methods discussed in this survey (e.g., Event2Vec [37], HyperGCN [171], DHE [120], HHNE [18]) use this technique to compare their computed embeddings against some baselines.

6.3.3 Knowledge Hypergraphs. When it comes to model knowledge bases (KBs), graphs are the go-to approach since they naturally represent ternary relations, i.e., facts of the form ⟨*head, relation, tail*⟩, in KBs. Nonetheless, with n-ary relations, hypergraphs could be a valid alternative [46, 173]. Knowledge hypergraph embedding techniques are strictly interlinked to the task (usually predicting facts); hence, it is hard to transfer these methodologies to different contexts.

6.3.4 Natural Sciences. Over the last decades, network science has become an established framework to analyze interactions between biological and chemical agents [17]. In this context, the task of hypergraph representation learning has been leveraged to predict the material removal rate in chemical-mechanical processes [164, 165], multi-way chromatic interactions [188, 189], genome features [190], and drug-disease association [119]. Other work also focused on medical-related issues, such as cancer tissue classification [12], autism diagnosis [111], and plant disease detection [133].

7 FUTURE DIRECTIONS

In the following, we discuss challenges and potential future directions related to hypergraph representation learning.

Deep hypergraph representation learning. The last years have seen the rise of (deep) neural hypergraph embeddings. This trend follows the trail of GNNs, which have shown outstanding per-

formance on graphs. However, most of the efforts have been devoted to adapting existing ideas on graphs to hypergraphs. A real challenge for the future is developing neural models exclusively designed for hypergraphs that can exploit their peculiar characteristics. For instance, most convHNN methods approximate each hyperedge with a clique and, hence, lose the relation's indecomposable nature. A possible direction is to work toward MP functions that aggregate neighbors' representations by directly exploiting high-order interactions rather than pairwise connections, following the footsteps of [35, 136].

Beyond node embeddings. Most of the methods presented in this survey deal with node embeddings, with only a few exceptions that also consider hyperedges. Nonetheless, there is a need to develop solutions able to learn representations for whole hypergraphs or sub-structures. These methods can be useful in visualization [11], community detection [27], and hyper knowledge-graph embedding [129] tasks. Following the spirit of the message-passing framework, a promising direction is learning permutation-invariant functions able to aggregate the node/hyperedge representations in a meaningful way for the task at hand.

Dynamic hypergraph embedding. Dynamicity is an essential characteristic of many networks and can manifest either as time-varying features or structural interaction patterns [5, 16]. (Hyper)graphs may evolve in terms of node/edge structure (addition/removal) or information. These characteristics cause the static embedding approach to fail as the hypergraph size is not fixed, features may drift over time, and adding new nodes/edges requires efficiently updating representations [31]. Current work on dynamic hypergraph embedding assumes a transductive setting, with a fixed node set and a variable set of hyperedges [87] or node features [95, 152, 154, 164, 177]. Still, a more challenging problem is predicting newly added nodes' representations. As inductive frameworks exist, the major challenge here is how to update the existing node representations incrementally and adapt to the concept drift of the graph structure. Inspired by the graph representation learning literature [66], a promising direction could be to learn functions that generate embeddings by aggregating features from nodes' neighborhoods instead of training individual node embeddings.

Interpretability. Most state-of-the-art hypergraph embedding methods are built using convolutions, usually in a layered architecture. The complexity of such models makes them expressive enough to extract a good condensed hypergraph representation. Yet, the highly non-linear nature of these models harms their interpretability. In general, to interpret an embedding, we need to find an association between latent features and features of the original hypergraph. A possible direction could be exploring the so-called disentangled representations [21]. Disentangled representation learning [64, 94] may help to learn uncorrelated latent features, which may allow characterizing the various underlying explanatory factors behind the learned factorized representations.

Scalability. Scalability is a critical requirement when dealing with many real-world networks. Hypergraphs add further complexity as not only the number of nodes and hyperedges can scale to millions but also the same hyperedges cardinality may be huge. Currently, little work exists to improve hypergraph algorithms' scalability (e.g., [171]); thus, scalable computational paradigms and models are a critical challenge for future work. NetVec [112] embodies an effort toward this direction by applying a coarsening strategy to preprocess data. In general, hypergraph coarsening or partitioning strategies combined with parallel approaches may represent a valid approach.

Reproducibility. How to directly measure the quality of data representation is a long-standing problem in representation learning [93]. Currently, the quality of the learned representation is measured by indirectly executing a given set of relevant tasks. However, no standard benchmark is currently available in terms of data and methods. In addition to the (not so rare) unavailability of open-source implementations, this fragmentation leads to approaches difficult to reuse and

compare. As suggested by Hamilton et al. [67] in the case of graph representation learning, effort should be put into defining a common framework describing—for a given task—the expected network structure to encode, how we expect the models to encode this information, and possible constraints on the learned representation to clarify which approach should be used when in real-world applications.

8 CONCLUSION

The ever-increasing development of computational capabilities and the advancements of theoretical models demand structures to unify and abstract real-world data with complex relations. Hypergraphs are a promising answer to this need; yet, the overall hypergraph research, or, specifically, the hypergraph representation learning field, is still in its infancy, but the body of work on the topic is growing rather quickly.

In this survey, we presented a systematic and thorough discussion about hypergraph representation learning. We identified in the literature three main families of approaches to the hypergraph embedding problem: spectral, proximity-preserving, and (deep) neural network hypergraph embedding methods.

- Spectral hypergraph embedding methods represent the first efforts toward learning vectorial representations for hypergraphs. Despite their solid theoretical foundation, spectral methods struggle with large-scale hypergraphs, thus making them less appealing to the research community in recent years.
- Both proximity-preserving and (deep) neural network methods overcome this scalability issue offering two very different approaches to the hypergraph representation learning problem.
  - Proximity-preserving methods use a standard machine learning pipeline where the proximity of the nodes in the latent space is preserved through a loss function defined on the distance (or similarity) between nodes. The loss function can also be designed to embed additional constraints or information tied to the specific task at hand. Such an easy design makes proximity-preserving methods adaptable to more disparate contexts.
  - Nowadays, deep-learning-based methods are predominant thanks to their flexibility, capacity, and performance. They take advantage of the experience from the graph representation learning literature and the ever-increasing availability of high computational power that allows training very large (i.e., deep) models. On the one hand, this inheritance from graph representation learning has sped up the development of learning techniques for hypergraphs. On the other hand, it has biased the researchers' efforts toward methods for hypergraphs that are adaptations of the ones for graphs. However, although similar, hypergraphs have inherent characteristics that must be considered; e.g., they encode higher-order relationships. Works like Chien et al. [35] and Srinivasan et al. [136] have opened up new avenues thanks to message-passing functions working on (multi-)sets rather than graph structures. We believe such contributions may inspire the research community to develop novel representation learning frameworks designed explicitly for hypergraphs.

One of the biggest challenges in hypergraph representation learning is the lack of (at least) an affirmed tool that researchers could use as a reference to implement new embedding techniques, while, for the graph counterpart, the community offers many alternatives like DGL,[4] Spektral,[5]

[4] https://docs.dgl.ai.
[5] https://graphneural.network.

and Karate Club,[6] just to name a few. Unluckily, there is no equivalent for hypergraphs, and even libraries for hypernetwork analysis are not so numerous (e.g., HyperNetX[7] [2], SimpleHypergraph.jl[8] [3, 4], Hypergraphx[9] [101]) and as mature as, for example, NetworkX[10] for graphs.

With this survey, we aim to provide future researchers and practitioners with a comprehensive understanding of current trends in addressing the hypergraph embedding problem and sketching some guidelines for designing novel and practical hypergraph representation learning techniques.

REFERENCES

[1] S. Agarwal, K. Branson, and S. Belongie. 2006. Higher order learning with graphs. In *Proc. of the 23rd Int. Conf. on Machine Learning*. ACM, 17–24.

[2] S. G. Aksoy, C. Joslyn, C. Ortiz Marrero, B. Praggastis, and E. Purvine. 2020. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science* 9, 1 (2020), 16.

[3] A. Antelmi, G. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, and P. Szufel. 2019. SimpleHypergraphs.jl—Novel software framework for modelling and analysis of hypergraphs. In *Algorithms and Models for the Web Graph*. Springer International Publishing, Cham, 115–129.

[4] A. Antelmi, G. Cordasco, B. Kamiński, P. Prałat, V. Scarano, C. Spagnuolo, and P. Szufel. 2020. Analyzing, exploring, and visualizing complex networks via hypergraphs using simplehypergraphs.jl. *Internet Mathematics* 1, 1 (2020), 1–32.

[5] A. Antelmi, G. Cordasco, C. Spagnuolo, and V. Scarano. 2020. A design-methodology for epidemic dynamics via time-varying hypergraphs. In *Proc. of the 19th Int. Conf. on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 61–69.

[6] A. Antelmi, G. Cordasco, C. Spagnuolo, and P. Szufel. 2021. Social influence maximization in hypergraphs. *Entropy* 23, 7 (2021). arxiv https://arxiv.org/abs/2010.04558

[7] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring. 2020. HyperSAGE: Generalizing Inductive Representation Learning on Hypergraphs.

[8] G. Ausiello, P. G. Franciosa, and D. Frigioni. 2001. Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *Proc. of the 7th Italian Conf. on Theoretical Computer Science*. Springer-Verlag, 312–327.

[9] J. Bai, B. Gong, Y. Zhao, F. Lei, C. Yan, and Y. Gao. 2021. Multi-scale representation learning on hypergraph for 3d shape retrieval and recognition. *IEEE Trans. on Image Processing* 30 (2021), 5327–5338.

[10] S. Bai, F. Zhang, and P. H. S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.

[11] Y. Bai, H. Ding, Y. Qiao, A. Marinovic, K. Gu, T. Chen, Y. Sun, and W. Wang. 2019. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence*. Int. Joint Conferences on Artificial Intelligence Organization, 1988–1994.

[12] A. B. Bakht, S. Javed, H. AlMarzouqi, A. Khandoker, and N. Werghi. 2021. Colorectal cancer tissue classification using semi-supervised hypergraph convolutional network. In *2021 IEEE 18th International Symposium on Biomedical Imaging*. 1306–1309.

[13] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine. 2020. Bridging the Gap between Spectral and Spatial Domains in Graph Neural Networks.

[14] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine. 2020. When spectral domain meets spatial domain in graph neural networks. In *37th Int. Conf. on Machine Learning - Workshop on Graph Representation Learning and Beyond (GRL+'20)*. 1–9.

[15] S. Bandyopadhyay, K. Das, and M. N. Murty. 2020. Hypergraph attention isomorphism network by learning line graph expansion. In *2020 IEEE Int. Conf. on Big Data*. 669–678.

[16] C. D. T. Barros, M. R. F. Mendonça, A. B. Vieira, and A. Ziviani. 2021. A survey on embedding dynamic graphs. *Comput. Surveys* 55, 1, Article 10 (2021), 37 pages.

[17] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G Young, and G. Petri. 2020. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports* 874 (2020), 1–92.

[18] I. M. Baytas, C. Xiao, F. Wang, A. K. Jain, and J. Zhou. 2018. Heterogeneous hyper-network embedding. In *2018 IEEE Int. Conf. on Data Mining*. 875–880.

[6] https://karateclub.readthedocs.io.

[7] https://pnnl.github.io/HyperNetX.

[8] https://github.com/pszufe/SimpleHypergraphs.jl.

[9] https://github.com/HGX-Team/hypergraphx.

[10] https://networkx.org/.

[19] N. C. Behague, A. Bonato, M. A. Huggan, R. Malik, and T. G. Marbach. 2021. The iterated local transitivity model for hypergraphs. *CoRR* abs/2101.12560 (2021).
[20] M. Belkin and P. Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15, 6 (2003), 1373–1396.
[21] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.
[22] Á. Bodó, G. Y. Katona, and Pé. L. Simon. 2016. SIS epidemic propagation on hypergraphs. *Bulletin of Mathematical Biology* 78, 4 (2016), 713–735.
[23] M. Bolla. 1993. Spectra, euclidean representations and clusterings of hypergraphs. *Discrete Mathematics* 117, 1–3 (1993), 19–39.
[24] A Bretto. 2013. *Hypergraph Theory: An Introduction*. Springer International Publishing.
[25] D. Cai, X. He, and J. Han. 2007. Spectral regression: A unified subspace learning framework for content-based image retrieval. In *Proc. of the 15th ACM Int. Conf. on Multimedia*. ACM, 403–412.
[26] H. Cai, V. W. Zheng, and K. C. Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.
[27] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *Proc. of the 2017 ACM on Conf. on Information and Knowledge Management*. ACM, 377–386.
[28] T. H. H. Chan, A. Louis, Z. G. Tang, and C. Zhang. 2018. Spectral properties of hypergraph Laplacian and approximation algorithms. *J. of the ACM* 65, 3, Article 15 (2018), 48 pages.
[29] C. Chen, Z. Cheng, Z. Li, and M. Wang. 2020. Hypergraph attention networks. In *2020 IEEE 19th Int. Conf. on Trust, Security and Privacy in Computing and Communications*. 1560–1565.
[30] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proc. of the AAAI Conf. on Artificial Intelligence* 34, 4 (2020), 3438–3445.
[31] F. Chen, Y. Wang, B. Wang, and C.-C. J. Kuo. 2020. Graph representation learning: A survey. *APSIPA Trans. on Signal and Information Processing* 9 (2020), e15.
[32] H. Chen, H. Yin, X. Sun, T. Chen, Bogdan G., and K. Musial. 2020. Multi-level graph convolutional networks for cross-platform anchor link prediction. In *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*. ACM, 1503–1511.
[33] X. Chen, K. Xiong, Y. Zhang, L. Xia, D. Yin, and J. X. Huang. 2020. Neural feature-aware recommendation with signed hypergraph convolutional network. *ACM Trans. on Information Systems* 39, 1, Article 8 (2020), 22 pages.
[34] D. Cheng, J. Chen, W. Peng, W. Ye, F. Lv, T. Zhuang, X. Zeng, and X. He. 2022. IHGNN: Interactive hypergraph neural network for personalized product search. In *Proc. of the ACM Web Conference 2022*. ACM, 256–265.
[35] E. Chien, C. Pan, J. Peng, and O. Milenkovic. 2022. You are allset: A multiset function framework for hypergraph neural networks. In *Int. Conf. on Learning Representations*.
[36] Y. Chu, C. Feng, and C. Guo. 2018. Social-guided representation learning for images via deep heterogeneous hypergraph embedding. In *2018 IEEE Int. Conf. on Multimedia and Expo*. 1–6.
[37] Y. Chu, C. Feng, C. Guo, Y. Wang, and J. N. Hwang. 2019. Event2vec: Heterogeneous hypergraph embedding for event data. In *IEEE Int. Conf. on Data Mining Workshops*, 1022–1029.
[38] F. R. K. Chung. 1997. *Spectral Graph Theory*. American Mathematical Society.
[39] G. F. de Arruda, G. Petri, and Y. Moreno. 2020. Social contagion models on hypergraphs. *Physical Review Research* 2, 2 (2020), 023032.
[40] K. Ding, J. Wang, J. Li, D. Li, and H. Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. In *Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 4927–4936.
[41] M. Ding, X. Lin, B. Zeng, and Y. Chai. 2021. Hypergraph neural networks with attention mechanism for session-based recommendation. *J. of Physics: Conference Series* 2082, 1 (2021), 012007.
[42] Y. Dong, Z. Hu, K. Wang, Y. Sun, and J. Tang. 2020. Heterogeneous network representation learning. In *Proc. of the 29th Int. Joint Conf. on Artificial Intelligence*. Int. Joint Conferences on Artificial Intelligence Organization, 4861–4867.
[43] Y. Dong, W. Sawin, and Y. Bengio. 2020. HNHN: Hypergraph networks with hyperedge neurons. In *Graph Representation Learning and Beyond Workshop at ICML 2020*.
[44] G. Faggioli, M. Polato, and F. Aiolli. 2020. Recency aware collaborative filtering for next basket recommendation. In *Proc. of the 28th ACM Conf. on User Modeling, Adaptation and Personalization*. ACM, 80–87.
[45] H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, and Q. Dai. 2021. Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4125–4138.

[46] B. Fatemi, P. Taslakian, D. Vazquez, and D. Poole. 2021. Knowledge hypergraphs: Prediction beyond binary relations. In *Proc. of the 29th Int. Joint Conf. on Artificial Intelligence*. Article 303.
[47] M. R. Felipe-Lucia, A. M. Guerrero, S. M. Alexander, J. Ashander, J. A. Baggio, M. L. Barnes, O. Bodin, A. Bonn, M.-J. Fortin, R. S. Friedman, J. A. Gephart, K. J. Helmstedt, A. A. Keyes, K. Kroetz, F. Massol, M. J.O. Pocock, J. Sayles, R. M. Thompson, S. A. Wood, and L. E. Dee. 2022. Conceptualizing ecosystem services using social–ecological networks. *Trends in Ecology & Evolution* 37, 3 (2022), 211–222.
[48] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao. 2019. Hypergraph neural networks. *Proc. of the AAAI Conf. on Artificial Intelligence* 33, 1 (2019), 3558–3565.
[49] S. Fu, W. Liu, Y. Zhou, and L. Nie. 2019. HpLapGCN: hypergraph P-laplacian graph convolutional networks. *Neurocomputing* 362 (2019), 166–174.
[50] Fukunaga, Yamada, Stone, and Kasai. 1984. A representation of hypergraphs in the Euclidean space. *IEEE Trans. on Computers* C-33, 4 (1984), 364–367. https://ieeexplore.ieee.org/document/1676443.
[51] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics* 42, 2 (1993), 177–201.
[52] H. Gao and H. Huang. 2018. Deep attributed network embedding. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence*. AAAI Press, 3364–3370.
[53] Y. Gao, Y. Feng, S. Ji, and R. Ji. 2023. HGNN+: General hypergraph neural networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 45, 3 (2023), 3181–3199.
[54] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou. 2022. Hypergraph learning: Methods and practices. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 44, 5 (2022), 2548–2566.
[55] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.
[56] S. Gopalakrishnan, S. Sridharan, S. R. Nayak, J. Nayak, and S. Venkataraman. 2022. Central hubs prediction for bio networks by directed hypergraph - GA with validation to COVID-19 PPI. *Pattern Recognition Letters* 153 (2022), 246–253.
[57] D. Grattarola. 2021. *Graph Neural Networks Operators and Architectures*. Ph. D. Dissertation. Faculty of Informatics of the Università della Svizzera Italiana.
[58] J. Grilli, G. Barabás, M. J. Michalska-Smith, and S. Allesina. 2017. Higher-order interactions stabilize dynamics in competitive network models. *Nature* 548, 7666 (2017), 210–213.
[59] A. Grover and J. Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 855–864.
[60] Y. Guan, X. Sun, and Y. Sun. 2021. Sparse relation prediction based on hypergraph neural networks in online social networks. *World Wide Web* 26 (2021), 7–31.
[61] H. Gui, J. Liu, F. Tao, M. Jiang, B. Norick, and J. Han. 2016. Large-scale embedding learning in heterogeneous event data. In *2016 IEEE 16th Int. Conf. on Data Mining*, 907–912.
[62] H. Gui, J. Liu, F. Tao, M. Jiang, B. Norick, L. Kaplan, and J. Han. 2017. Embedding learning with events in heterogeneous information networks. *IEEE Trans. on Knowledge and Data Engineering* 29, 11 (2017), 2428–2441.
[63] L. Guo, H. Yin, T. Chen, X. Zhang, and K. Zheng. 2021. Hierarchical hyperedge embedding-based representation learning for group recommendation. *ACM Trans. on Information Systems* 40, 1, Article 3 (2021), 27 pages.
[64] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, and Y. Ye. 2020. Interpretable deep graph generation with node-edge co-disentanglement. In *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 1697–1707.
[65] W. L. Hamilton. 2020. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14. Morgan & Claypool Publishers.
[66] W. L. Hamilton, R. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*. Curran Associates Inc., 1025–1035.
[67] W. L. Hamilton, R. Ying, and R. Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin* 40, 3 (2017), 52–74.
[68] J. Han, Q. Tao, Y. Tang, and Y. Xia. 2022. DH-HGCN: Dual homogeneity hypergraph convolutional network for multiple social recommendations. In *Proc. of the 45th International ACM SIGIR Conf. on Research and Development in Information Retrieval*. ACM, 2190–2194.
[69] Z. S. Harris. 1954. Distributional structure. *WORD* 10, 2–3 (1954), 146–162.
[70] L. He, H. Chen, D. Wang, S. Jameel, P. Yu, and G. Xu. 2021. Click-through rate prediction with multi-modal hypergraphs. In *Proc. of the 30th ACM Int. Conf. on Information & Knowledge Management*. ACM, 690–699.
[71] D. J. Higham and H.-L. de Kergorlay. 2021. Epidemics on hypergraphs: Spectral thresholds for extinction. *Proc. of the Royal Society A: Mathematical, Physical and Engineering Sciences* 477, 2252 (2021), 20210232.
[72] Z. Hu, J. Wang, S. Chen, and X. Du. 2021. A semi-supervised framework with efficient feature extraction and network alignment for user identity linkage. In *Proc. of the 26th Int. Conf. of Database Systems for Advanced Applications*. Springer-Verlag, 675–691.

[73] G.-B. Huang, Q.-Y. Zhu, and C.-k Siew. 2006. Extreme learning machine: Theory and applications. *Neurocomputing* 70, 1 (2006), 489–501.
[74] J. Huang, Ch. Chen, F. Ye, W. Hu, and Z. Zheng. 2020. Nonuniform hyper-network embedding with dual mechanism. *ACM Trans. on Information Systems* 38, 3, Article 28 (2020), 1–18.
[75] J. Huang, C. Chen, F. Ye, J. Wu, Z. Zheng, and G. Ling. 2019. Hyper2vec: Biased random walk for hyper-network embedding. In *Database Systems for Advanced Applications*, Vol. 11448 LNCS. Springer International Publishing, Cham, 273–277.
[76] J. Huang, X. Huang, and J. Yang. 2021. Residual enhanced multi-hypergraph neural network. In *2021 IEEE Int. Conf. on Image Processing*. 3657–3661.
[77] J. Huang, F. Lei, S. Wang, S. Wang, and Q. Dai. 2021. Hypergraph convolutional network with hybrid higher-order neighbors. In *Pattern Recognition and Computer Vision*. Springer International Publishing, Cham, 103–114.
[78] J. Huang, X. Liu, and Y. Song. 2019. Hyper-path-based representation learning for hyper-networks. In *Proc. of the 28th ACM Int. Conf. on Information and Knowledge Management*. ACM, 449–458.
[79] J. Huang and J. Yang. 2021. UniGNN: A unified framework for graph and hypergraph neural networks. In *Proc. of the 30th Int. Joint Conf. on Artificial Intelligence*. Int. Joint Conferences on Artificial Intelligence Organization, 2563–2569.
[80] S. Huang, D. Yang, Y. Ge, and X. Zhang. 2016. Discriminant hyper-Laplacian projections and its scalable extension for dimensionality reduction. *Neurocomputing* 173 (2016), 145–153.
[81] I. Iacopini, G. Petri, A. Baronchelli, and A. Barrat. 2022. Group interactions modulate critical mass dynamics in social convention. *Communications Physics* 5, 1 (2022), 64.
[82] B. Jhun, M. Jo, and B. Kahng. 2019. Simplicial SIS model in scale-free uniform hypergraph. *J. of Statistical Mechanics: Theory and Experiment* 2019, 12 (2019), 123207.
[83] J. Ji, Y. Ren, and M. Lei. 2022. FC-HAT: Hypergraph attention network for functional brain network classification. *Information Sciences* 608 (2022), 1301–1316.
[84] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao. 2020. Dual channel hypergraph collaborative filtering. In *Proc. of the Int. Conf. on Knowledge Discovery & Data Mining*. ACM, 2020–2029.
[85] R. Jia, X. Zhou, L. Dong, and S. Pan. 2021. Hypergraph convolutional network for group recommendation. In *2021 IEEE Int. Conf. on Data Mining*. 260–269.
[86] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao. 2019. Dynamic hypergraph neural networks. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence*. Int. Joint Conferences on Artificial Intelligence Organization, 2635–2641.
[87] H. Jin, Y. Wu, H. Huang, Y. Song, H. Wei, and X. Shi. 2022. Modeling information diffusion with sequential interactive hypergraphs. *IEEE Trans. on Sustainable Computing* 7, 3 (2022), 644–655.
[88] J. Jo, J. Baek, S. Lee, D. Kim, M. Kang, and S. J. Hwang. 2021. Edge representation learning with hypergraphs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 7534–7546.
[89] J. Kim, S. Oh, and S. Hong. 2021. Transformers generalize deepsets and can be extended to graphs and hypergraphs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 28016–28028.
[90] D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980.
[91] V. La Gatta, V. Moscato, M. Pennone, M. Postiglione, and G. Sperli. 2022. Music recommendation via hypergraph embedding. *IEEE Trans. on Neural Networks and Learning Systems* (2022), 1–13. https://ieeexplore.ieee.org/document/9709542.
[92] N. W. Landry and J. G. Restrepo. 2020. The effect of heterogeneity on hypergraph contagion models. *Chaos: An Interdisciplinary J. of Nonlinear Science* 30, 10 (2020), 103117.
[93] B. Li and D. Pi. 2020. Network representation learning: A systematic literature review. *Neural Computing and Applications* 32, 21 (2020), 16647–16679.
[94] H. Li, X. Wang, Z. Zhang, Z. Yuan, H. Li, and W. Zhu. 2021. Disentangled contrastive learning on graphs. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 21872–21884.
[95] Y. Li, H. Chen, X. Sun, Z. Sun, L. Li, L. Cui, P.S. Yu, and G. Xu. 2021. Hyperbolic hypergraphs for sequential recommendation. In *Proc. of the 30th ACM Int. Conf. on Information & Knowledge Management*. ACM, 988–997.
[96] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. 2016. Gated graph sequence neural networks. In *Int. Conf. on Learning Representations*.
[97] X. Liao, Y. Xu, and H. Ling. 2021. Hypergraph neural networks for hypergraph matching. In *2021 IEEE/CVF Int. Conf. on Computer Vision*. 1246–1255.
[98] D. Liben-Nowell and J. Kleinberg. 2007. The link-prediction problem for social networks. *J. of the American Society for Information Science and Technology* 58, 7 (2007), 1019–1031.
[99] B. Liu, P. Zhao, F. Zhuang, X. Xian, Y. Liu, and V. S. Sheng. 2021. Knowledge-aware hypergraph neural network for recommender systems. In *Database Systems for Advanced Applications*. Springer International Publishing, Cham, 132–147.

[100] Z. Liu, Z. Zhang, Y. Cai, Y. Miao, and Z. Chen. 2021. Semi-supervised classification via hypergraph convolutional extreme learning machine. *Applied Sciences* 11, 9 (2021), 3867. https://www.mdpi.com/2076-3417/11/9/3867.
[101] Q. F. Lotito, M. Contisciani, C. De Bacco, L. Di Gaetano, L. Gallo, A. Montresor, F. Musciotto, N. Ruggeri, and F. Battiston. 2023. Hypergraphx: A library for higher-order network analysis. *J. of Complex Networks* 11, 3 (2023). https://academic.oup.com/comnet/article-abstract/11/3/cnad019/7180959.
[102] Q. F. Lotito, F. Musciotto, A. Montresor, and F. Battiston. 2022. Higher-order motif analysis in hypergraphs. *Communications Physics* 5, 1 (2022), 79.
[103] F. Luo, B. Du, L. Zhang, L. Zhang, and D. Tao. 2019. Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image. *IEEE Trans. on Cybernetics* 49, 7 (2019), 2406–2419.
[104] F. Luo, G. Guo, Z. Lin, J. Ren, and X. Zhou. 2020. Semisupervised hypergraph discriminant learning for dimensionality reduction of hyperspectral image. *IEEE J. of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), 4242–4256.
[105] X. Luo, J. Peng, and J. Liang. 2022. Directed hypergraph attention network for traffic forecasting. *IET Intelligent Transport Systems* 16, 1 (2022), 85–98.
[106] M. Jichao, W. Yanjiang, L. Baodi, and L. Weifeng. 2021. Accurately modeling the human brain functional correlations with hypergraph laplacian. *Neurocomputing* 428 (2021), 239–247.
[107] T. Ma and J. Guo. 2018. Study on information transmission model of enterprise informal organizations based on the hypernetwork. *Chinese J. of Physics* 56, 5 (2018), 2424–2438.
[108] X. Ma, W. Liu, Q. Tian, and Y. Gao. 2022. Learning representation on optimized high-order manifold for visual classification. *IEEE Trans. on Multimedia* 24 (2022), 3989–4001.
[109] Z. Ma, Z. Jiang, and H. Zhang. 2021. Hyperspectral image classification using spectral-spatial hypergraph convolution neural network. In *Image and Signal Processing for Remote Sensing XXVII*, Vol. 11862. SPIE, 118620I.
[110] Z. Ma, Z. Jiang, and H. Zhang. 2022. Hyperspectral image classification using feature fusion hypergraph convolution neural network. *IEEE Trans. On Geoscience and Remote Sensing* 60 (2022), 1–14.
[111] M. Madine, I. Rekik, and N. Werghi. 2020. Diagnosing autism using T1-W MRI with multi-kernel learning and hypergraph neural network. In *2020 IEEE Int. Conf. on Image Processing*. 438–442.
[112] S. Maleki, D. P. Wall, and K. Pingali. 2021. NetVec: A scalable hypergraph embedding system. In *Proc. of the Int. Conf. on Machine Learning - Workshops*.
[113] V. Martínez, F. Berzal, and J.-C. Cubero. 2016. A survey of link prediction in complex networks. *Comput. Surveys* 49, 4, Article 69 (2016), 33 pages.
[114] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *Proc. of Int. Conf. on Learning Representation*.
[115] L. Neuhäuser, R. Lambiotte, and M. T. Schaub. 2021. Consensus dynamics on temporal hypergraphs. *Physics Review E* 104, 6 (2021), 064305.
[116] L. Neuhäuser, A. Mellor, and R. Lambiotte. 2020. Multibody interactions and nonlinear consensus dynamics on networked systems. *Physics Review E* 101, 3 (2020), 032310.
[117] L. Nong, J. Wang, J. Lin, H. Qiu, L. Zheng, and W. Zhang. 2021. Hypergraph wavelet neural networks for 3d object classification. *Neurocomputing* 463, C (2021), 580–595.
[118] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 1105–1114.
[119] S. Pang, K. Zhang, S. Wang, Y. Zhang, S. He, W. Wu, and S. Qiao. 2021. HGDD: A drug-disease high-order association information extraction method for drug repurposing via hypergraph. In *Bioinformatics Research and Applications*. Springer International Publishing, Cham, 424–435.
[120] J. Payne. 2019. Deep hyperedges: A framework for transductive and inductive learning on hypergraphs. In *Proc. of Neural Information Processing Systems*.
[121] D. Peng and S. Zhang. 2022. GC–HGNN: A global-context supported hypergraph neural network for enhancing session-based recommendation. *Electronic Commerce Research and Applications* 52 (2022), 101129.
[122] B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 701–710.
[123] L. Pu and B. Faltings. 2012. Hypergraph learning with hyperedge expansion. In *Machine Learning and Knowledge Discovery in Databases*. Springer, Berlin, 410–425.
[124] P. Ren, R. C. Wilson, and E. R. Hancock. 2008. Spectral embedding of feature hypergraphs. In *Structural, Syntactic, and Statistical Pattern Recognition*. 308–317.
[125] S. Rendle. 2010. Factorization machines. In *Proc. of the Int. Conf. on Data Mining*. 995–1000.
[126] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[127] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag.

[128] J. A. Rodrìguez. 2002. On the Laplacian eigenvalues and metric parameters of hypergraphs. *Linear and Multilinear Algebra* 50, 1 (2002), 1–14.
[129] P. Rosso, D. Yang, and P. Cudré-Mauroux. 2020. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proc. of the Web Conference 2020*. ACM, 1885–1896.
[130] R. Sahasrabuddhe, L. Neuhäuser, and R. Lambiotte. 2021. Modelling non-linear consensus dynamics on hypergraphs. *J. of Physics: Complexity* 2, 2 (2021), 025006.
[131] S. Saito, D. P. Mandic, and H. Suzuki. 2018. Hypergraph p-Laplacian: A differential geometry view. *Proc. of the AAAI Conf. on Artificial Intelligence* 32, 1 (2018).
[132] A. Sanchez-Gorostiaga, D. Bajić, M. L. Osborne, J. F. Poyatos, and A. Sanchez. 2019. High-order interactions distort the functional landscape of microbial consortia. *PLOS Biology* 17, 12 (12 2019), 1–34.
[133] N. Sasikaladevi. 2022. Robust and fast plant pathology prognostics (P3) tool based on deep convolutional neural network. *Multimedia Tools and Applications* 81, 5 (2022), 7271–7283.
[134] R. Sawhney, S. Agarwal, A. Wadhwa, T. Derr, and R. R. Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. *Proc. of the AAAI Conf. on Artificial Intelligence* 35, 1 (2021), 497–504.
[135] J. Shun. 2020. Practical parallel hypergraph algorithms. In *Proc. of the ACM Symposium on Principles and Practice of Parallel Programming*. 232–249.
[136] B. Srinivasan, D. Zheng, and G. Karypis. 2021. Learning over families of sets - Hypergraph representation learning for higher order tasks. In *Proc. of the 2021 SIAM Int. Conf. on Data Mining*. Siam Society, 756–764.
[137] L. Sun, S. Ji, and J. Ye. 2008. Hypergraph spectral learning for multi-label classification. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 668–676.
[138] X. Sun, H. Yin, B. Liu, H. Chen, J. Cao, Y. Shao, and N.Q. Viet Hung. 2021. Heterogeneous hypergraph embedding for graph classification. In *Proc. of the Int. Conf. on Web Search and Data Mining*. ACM, 725–733.
[139] X. Sun, H. Yin, B. Liu, H. Chen, Q. Meng, W. Han, and J. Cao. 2021. Multi-level hyperedge distillation for social linking prediction on sparsely observed networks. In *Proc. of the Web Conference 2021*. ACM, 2934–2945.
[140] X. Sun, H. Yin, B. Liu, Q. Meng, J. Cao, A. Zhou, and H. Chen. 2022. Structure learning via meta-hyperedge for dynamic rumor detection. *IEEE Trans. on Knowledge and Data Engineering* (2022), 1–12. https://ieeexplore.ieee.org/document/9946426.
[141] Y. Sun, Sujuan Wang, Qingshan Liu, Renlong Hang, and Guangcan Liu. 2017. Hypergraph embedding for spatial-spectral joint feature extraction in hyperspectral images. *Remote Sensing* 9, 5 (2017), 506. https://www.mdpi.com/2072-4292/9/5/506.
[142] Q. Suo, J. Guo, and A. Shen. 2018. Information spreading dynamics in hypernetworks. *Physica A: Statistical Mechanics and its Applications* 495 (2018), 475–487.
[143] J. Sybrandt and I. Safro. 2019. FOBE and HOBE: First- and High-Order Bipartite Embeddings. arXiv, abs/1905.10953.
[144] J. Sybrandt, R. Shaydulin, and I. Safro. 2020. Hypergraph partitioning with embeddings. *IEEE Trans. on Knowledge and Data Engineering* 34, 6 (2020), 2771–2782.
[145] T. Thonet, J.-M. Renders, M. Choi, and J. Kim. 2022. Joint personalized search and recommendation with hypergraph convolutional networks. In *Advances in Information Retrieval*. Springer-Verlag, 443–456.
[146] H. T. Trung, T. Van Vinh, N. T. Tam, J. Jo, H. Yin, and N. Q. V. Hung. 2022. Learning holistic interactions in LBSNs with high-order, dynamic, and multi-role contexts. *IEEE Trans. on Knowledge and Data Engineering* 35, 5 (2022), 5002–5016.
[147] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu. 2018. Structural deep embedding for hyper-networks. In *Proc. of the 32nd AAAI Conf. on Artificial Intelligence*. AAAI Press, Article 53.
[148] L. van der Maaten and G. Hinton. 2008. Visualizing data using t-SNE. *J. of Machine Learning Research* 9 (2008), 2579–2605.
[149] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.
[150] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. 2017. Graph attention networks. In *6th Int. Conf. on Learning Representations*.
[151] M. Vijaikumar, D. Hada, and S. Shevade. 2021. HyperTeNet: Hypergraph and transformer-based neural network for personalized list continuation. In *2021 IEEE Int. Conf. on Data Mining*. 1210–1215.
[152] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *Proc. of the 43rd Int. ACM Conf. on Research and Development in Information Retrieval*. ACM, 1101–1110.
[153] J. Wang, K. Ding, Z. Zhu, and J. Caverlee. 2021. Session-based recommendation with hypergraph attention networks. In *Proc. of the SIAM Int. Conf. on Data Mining*. Society for Industrial and Applied Mathematics, 82–90.
[154] J. Wang, Y. Zhang, L. Wang, Y. Hu, X. Piao, and B. Yin. 2022. Multitask hypergraph convolutional networks: A heterogeneous traffic prediction framework. *IEEE Trans. on Intelligent Transportation Systems* 23 (2022), 1–11.
[155] J. Wang, Y. Zhang, Y. Wei, Y. Hu, X. Piao, and B. Yin. 2021. Metro passenger flow prediction via dynamic hypergraph convolution networks. *IEEE Trans. on Intelligent Transportation Systems* 22, 12 (2021), 7891–7903.

[156] N . Wang, S. Wang, Y. Wang, Q. Z. Sheng, and M. A. Orgun. 2022. Exploiting intra- and inter-session dependencies for session-based recommendations. *World Wide Web* 25, 1 (2022), 425–443.
[157] R. Wang, Y. Li, S. Lin, H. Xie, Y. Xu, and J. C. S. Lui. 2021. On modeling influence maximization in social activity networks under general settings. *ACM Trans. on Knowledge Discovery from Data* 15, 6, Article 108 (2021), 28 pages.
[158] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Trans. on Big Data* 9 (2022), 415–436.
[159] L. Wu, D. Wang, K. Song, S. Feng, Y. Zhang, and G. Yu. 2021. Dual-view hypergraph neural networks for attributed graph learning. *Knowledge-Based Systems* 227 (2021), 107185.
[160] X. Wu, Q. Chen, W. Li, Y. Xiao, and B. Hu. 2020. AdaHGNN: Adaptive hypergraph neural networks for multi-label image classification. In *Proc. of the Int. Conf. on Multimedia*. ACM, 284–293.
[161] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Trans. on Neural Networks and Learning Systems* 32, 1 (2021), 4–24.
[162] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*. ACM, 70–79.
[163] L. Xia, C. Huang, and C. Zhang. 2022. Self-supervised hypergraph transformer for recommender systems. In *Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*. ACM, 2100–2109.
[164] L. Xia, P. Zheng, X. Huang, and C. Liu. 2021. A novel hypergraph convolution network-based approach for predicting the material removal rate in chemical mechanical planarization. *J. of Intelligent Manufacturing* 33, 8 (2021), 2295–2306.
[165] L. Xia, P. Zheng, and C. Liu. 2021. Predicting the material removal rate in chemical mechanical planarization process: A hypergraph neural network-based approach. *Proc. of the ASME Design Engineering Technical Conference* 85376 (2021).
[166] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. *Proc. of the AAAI Conf. on Artificial Intelligence* 35, 5 (2021), 4503–4511.
[167] B. Xu, N. Wang, T. Chen, and M. Li. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network.
[168] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong. 2022. Dynamic network embedding survey. *Neurocomputing* 472 (2022), 212–223. arxiv https://arxiv.org/abs/1505.00853
[169] H. Xue, L. Yang, V. Rajan, W. Jiang, Y. Wei, and Y. Lin. 2021. Multiplex bipartite network embedding using dual hypergraph convolutional networks. In *Proc. of the Web Conference 2021*. ACM, 1649–1660.
[170] N. Yadati. 2020. Neural message passing for multi-relational ordered and recursive hypergraphs. In *Proc. of the 34th Int. Conf. on Neural Information Processing Systems*. Curran Associates, Inc., Article 276.
[171] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar. 2019. HyperGCN: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.
[172] N. Yadati, V. Nitin, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar. 2020. NHP: Neural hypergraph link prediction. In *Proc. of the Int. Conf. on Information & Knowledge Management*. ACM, 1705–1714.
[173] S. Yan, Z. Zhang, X. Sun, G. Xu, L. Jin, and S. Li. 2022. HYPER2: Hyperbolic embedding for hyper-relational link prediction. *Neurocomputing* 492 (2022), 440–451.
[174] C. Yang, R. Wang, S. Yao, and T. Abdelzaher. 2022. Semi-supervised hypergraph node classification on hypergraph line expansion. In *Proc. of the 31st ACM Int. Conf. on Information & Knowledge Management*. ACM, 2352–2361.
[175] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in LBSNs: A hypergraph embedding approach. In *The World Wide Web Conference*. ACM, 2147–2157.
[176] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux. 2020. LBSN2Vec++: Heterogeneous hypergraph embedding for location-based social networks. *IEEE Trans. on Knowledge and Data Engineering* 34 (2020), 1843–1855.
[177] J. Yi and J. Park. 2020. Hypergraph convolutional recurrent neural network. In *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*. ACM, 3366–3376.
[178] S. Yoon, H. Song, K. Shin, and Y. Yi. 2020. How much and when do we need higher-order information in hypergraphs? A case study on hyperedge prediction. In *Proc. of the Web Conference 2020*. ACM, 2627–2633.
[179] C. Yu, C. Tai, T. Chan, and Y. Yang. 2018. Modeling multi-way relations with hypergraph embedding. In *Proc. of the ACM Int. Conf. on Information and Knowledge Management*. ACM, 1707–1710.
[180] Guihai Yu, Xiying Yuan, and Hui Qu. 2019. Signed k-uniform hypergraphs and tensors. *Linear Algebra Appl.* 580 (2019), 1–13.
[181] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proc. of the Web Conference 2021*. ACM, 413–424.
[182] H. Yuan and Y. Y. Tang. 2015. Learning with hypergraph for hyperspectral image feature extraction. *IEEE Geoscience and Remote Sensing Letters* 12, 8 (2015), 1695–1699.

[183] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 3391–3401.
[184] D. Zhang, J. Yin, X. Zhu, and C. Zhang. 2020. Network representation learning: A survey. *IEEE Trans. on Big Data* 6, 1 (2020), 3–28.
[185] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, and H. Yin. 2021. Double-scale self-supervised hypergraph learning for group recommendation. In *Proc. of the Int. Conf. on Information & Knowledge Management*. ACM, 2557–2567.
[186] L. Zhang, J. Guo, J. Wang, J. Wang, S. Li, and C. Zhang. 2022. Hypergraph and uncertain hypergraph representation learning theory and methods. *Mathematics* 10, 11 (2022), 1921. https://www.mdpi.com/2227-7390/10/11/1921.
[187] M. Zhang, H. Luo, W. Song, H. Mei, and C. Su. 2021. Spectral-spatial offset graph convolutional networks for hyperspectral image classification. *Remote Sensing* 13, 21 (2021), 4342. https://www.mdpi.com/2072-4292/13/21/4342.
[188] R. Zhang and J. Ma. 2020. MATCHA: Probing multi-way chromatin interaction with hypergraph representation learning. *Cell Systems* 10, 5 (2020), 397–407.e5.
[189] R. Zhang and J. Ma. 2020. Probing multi-way chromatin interaction with hypergraph representation learning. In *Research in Computational Molecular Biology*. Springer International Publishing, Cham, 276–277.
[190] R. Zhang, T. Zhou, and J. Ma. 2022. Multiscale and integrative single-cell hi-C analysis with Higashi. *Nature Biotechnology* 40, 2 (2022), 254–261.
[191] R. Zhang, Y. Zou, and J. Ma. 2020. Hyper-SAGNN: A self-attention based graph neural network for hypergraphs. In *Int. Conf. on Learning Representations*.
[192] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. 2019. Graph convolutional networks: A comprehensive review. *Computational Social Networks* 6, 1 (2019), 11.
[193] Q. Zheng and D. B. Skillicorn. 2015. Spectral embedding of directed networks. In *Proc. of the 2015 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining 2015*. ACM, 432–439.
[194] D. Zhou, J. Huang, and B. Schölkopf. 2007. Learning with hypergraphs: Clustering, classification, and embedding. In *Proc. of Neural Information Processing Systems*. 1601–1608.
[195] J. Zhu, S. Ghosh, and W. Wu. 2019. Group influence maximization problem in social networks. *IEEE Trans. on Computational Social Systems* 6, 6 (2019), 1156–1164.
[196] J. Zhu, J. Zhu, J. Ghosh, W. Wu, and J. Yuan. 2019. Social influence maximization in hypergraph in social networks. *IEEE Trans. on Network Science and Engineering* 6, 4 (2019), 801–811.
[197] Y. Zhu, Z. Guan, T. Tan, H. Liu, D. Cai, and X. He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.
[198] Y. Zhu and H. Zhao. 2022. Hypernetwork representation learning with the set constraint. *Applied Sciences* 12, 5 (2022), 2650. https://www.mdpi.com/2076-3417/12/5/2650.

Received 22 September 2022; revised 12 April 2023; accepted 31 May 2023