Learning High-Order Dynamics of Real-World Networks

Da Eun Lee*
ddanable05@hanyang.ac.kr
Hanyang University
Seoul, Korea

Song Kyung Yu*
ssong915@hanyang.ac.kr
Hanyang University
Seoul, Korea

Yunyong Ko
yyko@cau.ac.kr
Chung-Ang University
Seoul, Korea

Sang-Wook Kim
wook@hanyang.ac.kr
Hanyang University
Seoul, Korea

Abstract

Real-world networks naturally have high-order relationships among objects and they evolve over time. To capture such crucial properties, dynamic hypergraph learning has been studied in a range of fields. Via an in-depth preliminary analysis, we observe two important but under-explored characteristics of high-order dynamics of real-world networks: high-order relations tend to **(O1)** *have a structural and temporal influence on other relations in a short term* and **(O2)** *periodically re-appear in a long term.* In this paper, we propose **Lincoln**, a method for **L**earning h**I**gh-order dy**N**ami**C**s **O**f rea**L**-world **N**etworks, that employs (1) bi-interactional hyperedge encoding for short-term patterns and (2) periodic time injection for long-term patterns. Extensive experiments on seven real-world datasets verify the superiority of Lincoln over nine state-of-the-art methods in the hyperedge prediction on dynamic networks.

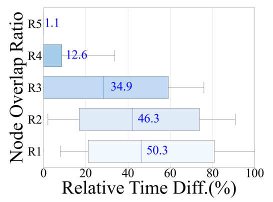
ACM Reference Format:
Da Eun Lee, Song Kyung Yu, Yunyong Ko, and Sang-Wook Kim. 2025. Learning High-Order Dynamics of Real-World Networks. In *Proceedings of (WWW '25 Companion).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXX

1 Introduction

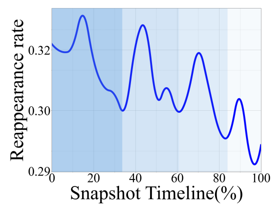
In real-world networks, high-order (i.e., *group-wise*) relations are prevalent, such as (1) a paper co-authored by a group of researchers in collaboration networks and (2) a chemical reaction co-induced by a group of proteins in protein-protein interaction (PPI) networks. A *hypergraph*, a generalized graph structure, can naturally model such group-wise relations as a *hyperedge* without any information loss unlike an ordinary graph. Due to its high expressive power, machine learning on hypergraphs has achieved breakthroughs in many tasks including node classification [6, 14, 15], clustering [9, 13], and link prediction [7, 20].

One important characteristic of real-world networks is that they are evolving dynamically, where some objects and relations can appear/disappear over time. For instance, a new researcher can join a research community (i.e., a new object appears), and a group of researchers may co-author a paper together (i.e., a new relation is formed). Although a number of graph learning methods [12, 16, 17] have been proposed to capture such dynamics, they inevitably suffer from information loss in modeling original group-wise relations [4, 6]. While, most of existing hypergraph learning methods have focused on *static* networks where the structures are fixed. Thus,

*Both authors contributed equally to this research.



(a) Observation 1 (short-term)



(b) Observation 2 (long-term)

Figure 1: Observations: high-order relations tend to (O1) influence the formation of other relations (short-term) and (O2) re-appear in the future periodically (long-term).

hypergraph learning for *dynamic* networks still remains under-explored only with a handful of existing studies [10, 18].

From this motivation, in this paper, we propose a novel snapshot-based hypergraph learning method, **Lincoln**, for effectively capturing high-order dynamics of real-world networks. The process of Lincoln is two-fold: (1) (**intra-snapshot learning**) it models objects and their high-order relations within a network snapshot as a hypergraph and represents them as embeddings by a hypergraph encoder and (2) (**inter-snapshot learning**) it updates the node embeddings based on newly observed nodes and hyperedges in the next snapshot to capture temporal patterns of the network structure evolving over time.

Furthermore, for better understanding high-order dynamics, we have analyzed the evolving patterns of real-world network structure. Specifically, we randomly select 50 high-order relations from the first 20% network snapshots and investigate how many the selected high-order relations *reappear* within the first 20% snapshots (short-term) and in the future snapshots (long-term). Figure 1(a) shows that **(O1)** *the more structurally-similar high-order relations tend to reappear in shorter time*, where the x-axis represents groups of hyperedge pairs according to the node overlap ratio, i.e., structural similarity (e.g., R1 represents a group with the lowest similarity and R5 represents a group with the highest similarity) and the y-axis represents the relative time difference from the formation time of the selected hyperedge pairs. The result implies that 'high-order relations within a snapshot have a structural and temporal influence on the formation of other relations'. Figure 1(b) shows that **(O2)** *high-order relations tend to periodically reappear in future snapshots*, which indicates that 'it is important to consider the periodic pattern of high-order relations across snapshots'.

Based on these observations, we propose two effective strategies, which are integrated into the process of Lincoln, for effectively capturing short-term and long-term patterns of high-order relations: **(1) Bi-interactional hyperedge encoding** that captures structural and temporal patterns of high-order relations within a snapshot for **(O1)** and **(2) Periodic time injection** that reflects long-term periodic time information into node embeddings for **(O2)**.

The main contributions of this work are as follows.

- **Observations**: We observe that high-order relations tend to **(O1)** *have a structural and temporal influence on other relations in a short term* and **(O2)** *periodically re-appear in a long term.*
- **Method**: We propose a novel dynamic hypergraph learning method, Lincoln, that effectively captures high-order dynamics of real-world networks by employing (1) bi-interactional hyperedge encoding and (2) periodic time injection.
- **Evaluation**: Via extensive experiments on seven real-world datasets, we demonstrate that (1) Lincoln *consistently* outperforms *all* nine competing methods in the hyperedge prediction on dynamic networks and (2) each of the proposed strategies is effective in improving the accuracy of Lincoln.

For reproducibility, we have released the code of Lincoln and datasets at: https://github.com/ssong915/LINCOLN.

2 Related Work

Graph learning methods. A number of graph learning methods, modeling a network as a graph, have achieved great success in many fields. TGAT [16] considers the time difference between relations in learning temporal node embeddings. TGN [12] learns temporal node embeddings based on nodes' neighbors and their long-term dependencies. ROLAND [17] extends static GNNs to a dynamic setting, while utilizing hierarchical node embeddings for better node representations. Theses methods, however, inevitably suffer from information loss in capturing high-order relations.

Hypergraph learning methods. To address the information loss, hypergraph learning methods, modeling a network as a *hypergraph*, have been widely studied. HGNN [6] learns high-order relations based on a clique expansion, which replaces hyperedges with cliques to transform a hypergraph into a graph. HNHN [5] represents a hypergraph as a bipartite graph, where one type corresponds to a node and the other corresponds to a hyperedge, and learns both node and hyperedge embeddings. AHP [8] employs an adversarial training-based model to generate realistic negative hyperedges. WHATsNET [4] learns within-hyperedge relationships among nodes via attention and positional encoding schemes for the edge-dependent node classification problem. Although these methods often outperform graph learning methods in many downstream tasks, they focus mainly on *static* networks where the structures are fixed. There have been only a handful of dynamic hypergraph learning methods. DHGNN [10] and TDHNN [18], define hyperedges and re-construct a hypergraph structure based on the defined hyperedges. These methods, however, often fail to capture temporal patterns of the network structure evolving over time.

3 Proposed Method: Lincoln

In this section, we present some preliminaries on dynamic hypergraph learning and describe our proposed method, **Lincoln**.

3.1 Preliminaries

Dynamic hypergraph. Consider a dynamic hypergraph as a sequence of hypergraph snapshots $H_D = \{H_1, H_2, \ldots, H_T\}$, where each snapshot $H_t = (V_t, E_t)$ consists of a set of nodes $V_t = \{v_{(1,t)}, v_{(2,t)}, ..., v_{(|V_t|,t)}\}$ and a set of hyperedges $E_t = \{e_{(1,t)}, e_{(2,t)}, ..., e_{(|E_t|,t)}\}$ with the time t. In each snapshot, node and hyperedge features are denoted as $\mathbf{P}_t \in \mathbb{R}^{|V_t| \times d}$ and $\mathbf{Q}_t \in \mathbb{R}^{|E_t| \times d}$, respectively.

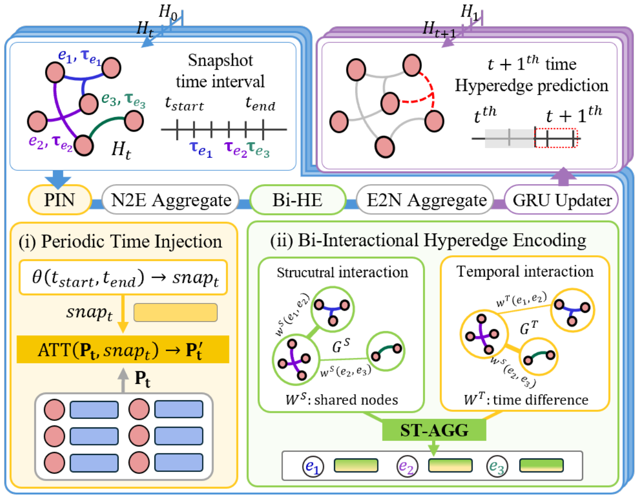


Figure 2: Overview: (1) intra-snapshot learning for the structural features within a snapshot and (2) inter-snapshot learning for the temporal features across snapshots.

Problem 1 (Dynamic Hyperedge Prediction). Given a dynamic hypergraph $H_D = \{H_1, H_2, ..., H_T\}$ and the initial node features $\mathbf{X} \in \mathbb{R}^{|V| \times F}$, we aim to learn the evolving dynamics of a network based on the previous t snapshots, thereby predicting whether a hyperedge candidate e' belongs to E_{t+1}.

3.2 Methodology

Overview. Figure 2 illustrates the overview of Lincoln, which generalizes static hypergraph learning into a dynamic setting. Lincoln aims to learn (1) the structural and temporal features within a snapshot and (2) the evolving networks across snapshots.

(1) Intra-Snapshot Learning. Given a hypergraph snapshot $H_t = (V_t, E_t)$, Lincoln produces node representations $\mathbf{P}_t \in \mathbb{R}^{|V_t| \times d}$ and hyperedge representations $\mathbf{Q}_t \in \mathbb{R}^{|E_t| \times d}$ by using a hyperedge encoder. Following [3, 5], Lincoln adopts a *2-stage aggregation* approach, which repeats (1) (*node-to-hyperedge*) producing the hyperedge embeddings by aggregating the node embeddings and (2) (*hyperedge-to-node*) producing the node embeddings by aggregating the hyperedge embeddings. Formally, the node and hyperedge embeddings at the k-th layer, $\mathbf{P}^{(k)}$ and $\mathbf{Q}^{(k)}$, are defined as:

$$\mathbf{Q_t}^{(k)} = \sigma(\mathbf{D}_{E_t}^{-1}\mathbf{H_t}^T\mathbf{P_t}^{(k-1)}\mathbf{W}_{E_t}^{(k)} + b_{E_t}^{(k)}), \quad (1)$$

$$\mathbf{P_t}^{(k)} = \sigma(\mathbf{D}_{V_t}^{-1}\mathbf{H_t}\mathbf{Q_t}^{(k)}\mathbf{W}_{V_t}^{(k)} + b_{V_t}^{(k)}), \quad (2)$$

where $\mathbf{P_t}^{(0)} = \mathbf{X}$, $\mathbf{W}_*^{(k)}$ and $b_*^{(k)}$ are trainable weight and bias matrices, respectively, $\mathbf{D}_*^{-1}$ is the normalization term, and σ is a non-linear activation function.

In addition, we integrate two effective strategies into the process of Lincoln for addressing the two important but under-explored characteristics of high-order relations: (1) *Bi-interactional hyperedge encoding* for **(O1)** and (2) *Periodic time injection* for **(O2)**.

(a) Bi-interactional hyperedge encoding: From the observation **(O1)**, the high-order relations within a snapshot play a structural and temporal role in forming other relations. To capture such an important feature, Lincoln (1) disentangles a high-order relation into structural and temporal embeddings and (2) learns structural and temporal correlations among high-order relations. Specifically, LINCOLN constructs two graphs (See G^S and G^T in Figure 2), where each node corresponds to a hyperedge and two hyperedge nodes are connected if the two hyperedges share the same nodes. The hyperedge nodes in each graph are initialized with $\mathbf{Q_t}$ (Eq. 1). We assign the edge weight $w^S(e_i, e_j)$ and $w^T(e_i, e_j)$ in G^S and G^T based on the following intuitions: two hyperedges are more likely to be closely connected if they share more nodes (i.e., structural proximity) or they occur in a shorter time period (i.e., temporal proximity). Then, it produces each hyperedge embedding into structural and temporal embeddings by applying a GNN model to G^S and G^T:

$$\mathbf{Q_t^S} = \text{GNN}^S(\mathbf{Q_t}, G^S), \quad \mathbf{Q_t^T} = \text{GNN}^T(\mathbf{Q_t}, G^T). \tag{3}$$

For a GNN model, we use a GCN model [19]. The two embeddings for each hyperedge are combined by an aggregation function:

$$\mathbf{Q_t'} = \text{ST-AGG}(\mathbf{Q_t^S}, \mathbf{Q_t^T}) = \sigma((\mathbf{Q_t^S} \oplus \mathbf{Q_t^T}) \cdot \mathbf{W}_{ST} + \mathbf{b}_{ST}), \tag{4}$$

where $\mathbf{W}_{ST}$ and $\mathbf{b}_{ST}$ are the trainable parameters. The resulting inter-relation-aware hyperedge embeddings ($\mathbf{Q_t'}$) are used in the E2N aggregation (Eq. 2), thereby reflecting the structural and temporal patterns of high-order relations into the learning process.

(b) Periodic time injection: Given the node embeddings $\mathbf{P_t}$ at time t, LINCOLN injects *period-specific* time information into the embeddings of nodes for reflecting the periodic patterns of high-order relations into them, before the N2E aggregation. Specifically, it (i) produces a snapshot-time embedding $snap_t$ by passing the start and end times of a snapshot t_{start} and t_{end} into a periodic function $\theta(\cdot)$ and (ii) aggregates $snap_t$ with $\mathbf{P_t}$ based on their attention weights computed by the snapshot-aware attention mechanism, to reflect different importance of each node in the context of the snapshot:

$$snap_t = \theta(t_{start}, t_{end}), \quad \mathbf{P_t'} = \text{ATT}(\mathbf{P_t}, snap_t). \tag{5}$$

For a periodic function $\theta(\cdot)$, we use a cosine function. The resulting snapshot-aware node embeddings ($\mathbf{P_t'}$) are used in the N2E aggregation (Eq.1), enabling LINCOLN to incorporate periodic patterns of high-order relations into its learning process.

(2) Inter-Snapshot Learning. Then, based on the node embedings learned from the previous snapshots, we aim to learn their temporal patterns across snapshots (i.e., inter-snapshot learning). Formally, LINCOLN updates the node embeddings at time t, $\mathbf{P_t}$, based on the node embeddings from the previous snapshot, $\mathbf{P_{t-1}^*}$, by using a temporal feature update module $g(\cdot)$ (e.g., GRU [1]):

$$\mathbf{P_t^*} = g(\mathbf{P_t}, \mathbf{P_{t-1}^*}). \tag{6}$$

Rather than using only the node embeddings from the final layer of intra-snapshot learning (i.e., $\mathbf{P_t^{(k)}}$) as the input of $g(\cdot)$, we take the intermediate node embeddings (i.e., $\mathbf{P_t^{(l)}}, 0 \leq l \leq k$) as the input and learns their temporal patterns for better understanding the complex network structure, by following [17]:

$$\mathbf{P_t^{*(l)}} = g(\mathbf{P_t^{(l)}}, \mathbf{P_{t-1}^{*(l)}}). \tag{7}$$

(3) Prediction and Training. Given the final node embeddings $\mathbf{P_t^*}$ at time t and a hyperedge candidate e' at time $t+1$, LINCOLN aggregates the embeddings of the nodes in e', $\mathbf{P_t}^*[e', :]$ and computes the probability of e' belonging to E_{t+1}, $\hat{y}_{e'}$, as:

$$\hat{y}_{e'} = pred(q_{e'}^*), \quad q_{e'}^* = \text{AGG}(\mathbf{P_t}^*[e', :]), \tag{8}$$

where $q_{e'}^* \in \mathbb{R}^d$ is the final hyperedge candidate embedding, $pred(\cdot)$ is a hyperedge predictor, a fully-connected layer ($d \times 1$) followed by a sigmoid function, and $\text{AGG}(\cdot)$ is an element-wise average pooling.

For the model training, we consider positive and negative hyperedges. For negative hyperedges, we use a heuristic negative sampling (NS) method, Motif NS (MNS) [11]. Thus, LINCOLN aims to train its model parameters so that positive examples obtain higher scores while negative examples obtain lower scores. Formally, the prediction loss is defined as:

$$\mathcal{L}_{pred} = -\frac{1}{|E'_{t+1}|} \sum_{e' \in E'_{t+1}} y_{e'} \cdot \log \hat{y}_{e'} + (1 - y_{e'}) \cdot \log (1 - \hat{y}_{e'}), \tag{9}$$

where $y_{e'}$ is the label of the hyperedge candidate e' (1 or 0).

Moreover, we use contrastive loss $\mathcal{L}_{con}$ between the structural and temporal hyperedge embeddings for better learning high-order relations, defined as $\mathcal{L}_{con} = -\log sim(\mathbf{Q_t^S}, \mathbf{Q_t^T})$, where $sim(\cdot)$ is the cosine similarity. Finally, the two losses are unified with a hyperparameter β to control the weight of the contrastive loss:

$$\mathcal{L} = \mathcal{L}_{pred} + \beta \mathcal{L}_{con}. \tag{10}$$

4 Experimental Validation

In this section, we comprehensively evaluate LINCOLN by answering the following questions.

- **EQ1**. To what extent does LINCOLN improve the existing methods in terms of the hyperedge prediction on dynamic networks?
- **EQ2**. Is each of our strategies beneficial for learning high-order dynamics of real-world relations?

4.1 Experimental Setup

Datasets and competitors.[1] We use seven real-world hypergraphs [2], which are classified into five categories: (1) email networks, (2) tagging networks, (3) thread participant networks, (4) contact networks, and (5) congress networks. We select 9 competitors, including 3 *dynamic graph-based* methods, 4 *static hypergraph-based* methods, and 2 *dynamic hypergraph-based* methods.

Evaluation protocol. We evaluate LINCOLN over the *hyperedge prediction* in a dynamic network. We use a 'live-update' evaluation setting, following [17]. It evaluates a model by utilizing *all* snapshots in a uniform way. It constructs training (70%), validation (20%), and test (10%) sets by splitting hyperedges 'within' each snapshot. As metrics, we use *area under ROC* (AUROC), and *average precision* (AP). We (1) measure AUROC on the test set at the epoch when the AUROC over the validation set is maximized, and (2) report the averaged AUROC and AP on the test set over five runs.

4.2 Experimental Results

Q1. Hyperedge prediction on dynamic networks. Table 1 shows the hyperedge prediction accuracy on seven dynamic hypergraphs. LINCOLN significantly outperforms competing methods in most

[1]More descriptions of datasets and competitors are provided in https://github.com/ssong915/LINCOLN.

Table 1: Hyperedge prediction accuracy (%) on seven dynamic hypergraphs. For each dataset, the best and the second-best results are highlighted in the bold font and underlined, respectively. OOM indicates 'out of memory' on a 11GB GPU.

Method	Email-enron		Email-eu		Tags		Thread		Contact-primary		Contact-high		Congress		Rank.
	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	
TGN	87.75	87.61	59.07	58.96	61.90	60.74	84.43	82.25	58.30	58.74	75.19	71.65	OOM	OOM	5.1
TGAT	**90.93**	89.72	44.18	47.10	48.49	50.97	83.91	80.12	44.10	50.09	41.79	49.26	OOM	OOM	7.0
ROLAND	80.37	82.42	64.20	66.12	67.37	67.04	80.76	85.23	74.47	76.60	70.96	75.59	94.01	92.90	3.0
HGNN	64.34	67.54	50.17	52.05	54.26	54.85	OOM	OOM	62.15	60.33	58.03	58.11	53.87	53.06	7.6
HNHN	74.29	70.79	55.55	53.78	57.53	55.56	50.48	50.68	56.44	54.77	59.04	59.21	58.57	57.10	7.2
AHP	81.99	81.38	61.97	59.73	62.89	62.85	75.23	74.54	72.55	68.22	83.68	80.27	68.06	75.71	4.0
WHATsNet	76.79	79.29	60.25	59.98	63.63	62.36	51.12	53.06	59.19	59.57	57.24	61.03	46.84	53.02	6.1
DHGNN	74.86	75.14	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	9.3
TDHNN	81.69	84.47	63.72	63.89	**76.65**	78.63	65.22	67.64	71.14	71.03	73.36	74.58	72.08	72.22	3.6
LINCOLN	84.35	**92.90**	**66.16**	**77.85**	72.19	**80.55**	**85.21**	**90.77**	**78.13**	**84.97**	**85.92**	**97.73**	**94.17**	**97.54**	**1.2**

Table 2: Ablation study: Each of our proposed strategies is beneficial to improving the accuracy of LINCOLN.

Method	Email-enron		Email-eu		Tags		Thread		Contact-primary		Contact-high		Congress	
	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP	AUROC	AP
LINCOLN	**84.35**	**92.90**	**66.16**	**77.85**	72.19	80.55	**85.21**	90.77	**77.84**	**84.32**	**85.92**	**97.73**	**94.17**	**97.54**
LINCOLN w/o PIN	75.84 ↓	88.60 ↓	59.95↓	70.69 ↓	70.82 ↓	78.99 ↓	80.03 ↓	94.17 ↑	75.99 ↓	83.91 ↓	82.23 ↓	96.90 ↓	92.03 ↓	96.90 ↓
LINCOLN w/o Bi-HE	72.01↓	86.54 ↓	65.13 ↓	77.00 ↓	**72.74** ↑	**81.31** ↑	81.78 ↓	**94.47** ↓	72.24↓	81.63↓	78.63↓	96.62↓	81.60↓	93.18↓

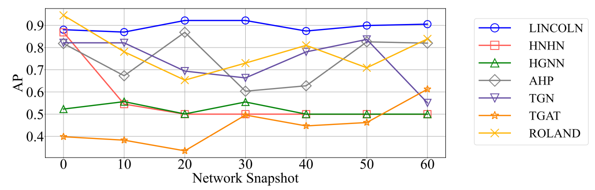


Figure 3: Hyperedge prediction accuracy on each snapshot.

cases, while achieving the best averaged rank. Via the t-tests with a 95% confidence level, we verify that the improvement of LINCOLN over all competing methods are statistically significant (i.e., the p-values ≤ 0.05). Figure 3 shows the accuracy (AP) of each method at every 10 snapshots. LINCOLN achieves the highest accuracies in most snapshots. Moreover, the accuracy of LINCOLN on each snapshot tends to be *stable*, while those of other methods tend to *fluctuate steeply*. These results verify that LINCOLN is able to effectively capture high-order dynamics than other competing methods.

Q2. Ablation study. We verify the effectiveness of our proposed strategies by ablating one of them: (i) periodic time injection (PI) and (ii) bi-interactional hyperedge encoding. Table 2 shows that the original version of LINCOLN consistently achieves the highest accuracy except for the Tags dataset, which indicates that ablating one of our proposed strategies could lead to performance degradation. Therefore, these results verify (1) the usefulness of our observations – (O1) their structural and temporal dependencies in a short term and (O2) periodic re-appearance of high-order relations in a long term – and (2) the effectiveness of our proposed strategies for capturing the short-term and long-term high-order dynamics.

5 Conclusion

In this paper, we have observed two important yet under-explored characteristics of real-world networks: high-order relations tend to **(O1)** *have a structural and temporal influence on others in a short term* and **(O2)** *periodically re-appear in a long term.* To address them, we propose LINCOLN for learning high-order dynamics of real-world networks, employing (1) bi-interactional hyperedge encoding and (2) periodic time injection. Via extensive experiments on seven real-world datasets, we verify the superiority of LINCOLN in capturing the high-order dynamics of real-world networks.

References

[1] Kyunghyun Cho. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078* (2014).
[2] Benson et al. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences* (2018).
[3] Chien et al. 2021. You are Allset: A Multiset Function Framework for Hypergraph Neural Networks. *arXiv:2106.13264* (2021).
[4] Choe et al. 2023. Classification of Edge-Dependent Labels of Nodes in Hypergraphs. In *SIGKDD*.
[5] Dong et al. 2020. Hnhn: Hypergraph Networks with Hyperedge Neurons. *arXiv:2006.12278* (2020).
[6] Feng et al. 2019. Hypergraph Neural Networks. In *AAAI*.
[7] Huo et al. 2018. Link Prediction with Personalized Social Influence. In *AAAI*.
[8] Hwang et al. 2022. Ahp: Learning to Negative Sample for Hyperedge Prediction. In *SIGIR*.
[9] Jure et al. 2007. Graph Evolution: Densification and Shrinking Diameters. *TKDD* (2007).
[10] Jiang et al. 2019. Dynamic Hypergraph Neural Networks. In *IJCAI*.
[11] Patil et al. 2020. Negative sampling for hyperlink prediction in networks. In *PAKDD*.
[12] Rossi et al. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv:2006.10637* (2020).
[13] Tsitsulin et al. 2023. Graph Clustering with Graph Neural Networks. *Journal of Machine Learning Research* (2023).
[14] Wu et al. 2019. Net: Degree-specific Graph Neural Networks for Node and Graph Classification. In *SIGKDD*.
[15] Wu et al. 2021. Enhancing Graph Neural Networks via Auxiliary Training for Semi-supervised Node Classification. *Knowledge-Based Systems* (2021).
[16] Xu et al. 2020. Inductive Representation Learning on Temporal Graphs. *arXiv:2002.07962* (2020).
[17] You et al. 2022. ROLAND: Graph Learning Framework for Dynamic Graphs. In *SIGKDD*.
[18] Zhou et al. 2023. Totally Dynamic Hypergraph Neural Network. In *IJCAI*.
[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907* (2016).
[20] Muhan Zhang and Yixin Chen. 2018. Link Prediction based on Graph Neural Networks. *In NeurIPS* (2018).