Directed Hypergraph Representation Learning for Link Prediction

Zitong Ma
School of Computer Science and Technology,
Soochow University
ztma99@stu.suda.edu.cn

Wenbo Zhao
School of Computer Science and Technology,
Soochow University
wbzhao@stu.suda.edu.cn

Zhe Yang*
School of Computer Science and Technology,
Soochow University
yangzhe@suda.edu.cn

Abstract

Link prediction is a critical problem in network structure processing. With the prevalence of deep learning, graph-based learning pattern in link prediction has been well-proven to successfully apply. However, existing representation-based computing paradigms retain some lack in processing complex networks: most methods only consider low-order pairwise information or eliminate the direction message, which tends to obtain a sub-optimal representation. To tackle the above challenges, we propose using directed hypergraph to model the real world and design a directed hypergraph neural network framework for data representation learning. Specifically, our work can be concluded into two sophisticated aspects: (1) We define the approximate Laplacian of the directed hypergraph, and further formulate the convolution operation on the directed hypergraph structure, solving the issue of the directed hypergraph structure representation learning. (2) By efficiently learning complex information from directed hypergraphs to obtain high-quality representations, we develop a framework DHGNN for link prediction on directed hypergraph structures. We empirically show that the merit of DHGNN lies in its ability to model complex correlations and encode information effectively of directed hypergraphs. Extensive experiments conducted on multi-field datasets demonstrate the superiority of the proposed DHGNN over various state-of-the-art approaches.

*Corresponding author. Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

1 INSTRUCTION

Link prediction is proposed for foreseeing the evolution of a graph (network) by extracting potential information from existing structures and features(Liben-Nowell and Kleinberg, 2003). The advantages of graph learning and deep learning methods in processing Non-Euclidean Structure Data and extracting relevant information have driven the development and deployment of link prediction in various fields(Zhang et al., 2018). Among them, the development of the neural network method represented by GCNs has served a wider application.

The link prediction task based on GCNs can be decomposed into two parts, 1) embedding learning: which models the realistic data and transforms it into vectorized representations, and 2) link prediction: to predict the link existence in pairs of nodes. To improve the quality of representation, a rising trend of leveraging more general data modeling and learning methods has been observed recently, such as hypergraphs, which are more suitable for the real world.

With the advent of Hypergraph Learning(Zhou et al., 2006) and its deepening research, numerous researchers have attracted considerable attention by modeling real-world data as hypergraph structures to capture high-order interactions and utilizing deep learning methods to obtain semantically sufficient representations(Gao et al., 2022). However, within modeling hypergraph is missing significant directionality in many practical applications. We can generalize directed graphs to directed hypergraphs analogously that hypergraphs are a generalization of undirected graphs(Ausiello and Laura, 2017).

Directed hypergraphs are different from directed graphs and hypergraphs, which summarize the high-order directed information of the network and perform higher-dimensional modeling of real data. It is common for users to purchase a bundle of items together while shopping online. As shown in Figure 5

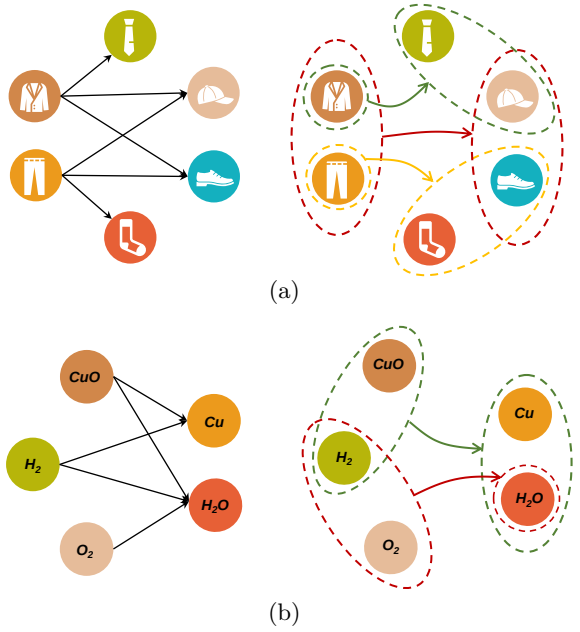

Figure 1: Examples of directed interactions on (a) co-purchase network and (b) chemical reaction.

(a), within directed graphs, edges can only point to a single vertex, which restricts the ability to model the notion that certain items are purchased together. However, grouping items by a hyperarc not only indicates that the bundled items are purchased simultaneously and contains the temporal ordering of purchased items. In chemical reaction networks(Figure 5 (b)), using directed graphs to represent reactions by pointing from reactants to products, omits the cooperative interaction between multiple reactants and makes it difficult to distinguish multiple distinct reaction processes, leading to the erroneous representation of reaction equations. This limitation can be overcome by using directed hypergraphs, where a single hyperarc can connect multiple reactants and multiple products simultaneously, enabling a more accurate representation of chemical reaction equations.

Despite the potential of directed hypergraphs, only a few works shift attention to directed hypergraph representation learning(Tran and Tran, 2020; Xiao et al., 2022). The Laplacian matrices are transformed to asymmetric due to the direction of the structure, which leads to the uncertainty of eigenvalue decomposition, whereas the hypergraph spectrum convolution requires a decomposable Laplacian. Thus numerical works simplify the problem by relaxing the directed limit to a directed to undirected transformation where both the set of tail and the head nodes are integrated into an ordinary hyperedge. In this approach, the original structure is corrupted, leading to misleading message passing. To tackle the aforementioned issue, we focus on developing efficient convolution approaches to learning complex directed hypergraph structures.

In this paper, we aim to express high-level information and obtain vertex embeddings through directed hypergraphs and spectral-based methods. We propose a representation learning framework on directed hypergraphs named Directed Hypergraph Neural Network (DHGNN), in which the high-order and direction semantics are preserved in the latent space. First, motivated by the spectral method for directed graphs and personalized PageRank algorithm(Tong et al., 2020; Bahmani et al., 2010), we formulate the regularization framework and derive Laplacian for directed hypergraphs by examining the underlying bonds between Markov chain stationary distribution and certain matrices associated with directed hypergraph and further extend the spectral convolution on the hypergraph to the directed hypergraph. Second, we design a multiple embedding fusion that integrates multi-granularity nodes and hyperarcs transitional signals into a common latent representation space and infers missing directed links by joint encoding source and target nodes.

In summary, the contributions of our work are:

- We propose a novel framework DHGNN for directed link prediction, which encodes and reveals the underlying interaction patterns in directed hypergraph structure.
- By introducing the personalized PageRank, we develop algorithms for directed hypergraph embedding and transductive inference based on the directed hypergraph. Moreover, we completed the generalization of hypergraph convolution to directed hypergraphs.
- In addition, we inject multiple embedding fusion for learning precise representation of the feature space under various levels of granularity, to foresee the link between a pair of nodes.
- We conduct extensive experiments on diverse domain datasets with explicit features and latent features respectively, demonstrating the effectiveness of the proposed DHGNN framework over state-of-the-art methods.

2 RELATED WORKS

In this section we review prior works related to DHGNN from three aspects: 1) Directed graph Embedding Learning; 2) Hypergraph Representation Learning; 3) Link Prediction.

2.1 Directed Graph Embedding Learning

Existing solutions generally be divided into the following two categories: (1) Model directed graphs and exploit GNN-derived methods. DIGCN(Tong et al., 2020) gives a definition of the Laplacian of directed graphs, extending spectrum-based undirected graph convolutions to directed graphs. D-HYPR(Zhou et al., 2022) introduces four canonical types of neighborhoods in hyperbolic space and captures graph semantics and structural information through a message-passing-based GNN method. (2) Preserve the asymmetry by generating embeddings for the nodes at both ends of the directed edge respectively (Khosla et al., 2020; Ou et al., 2016). For example, DiGAE(Kollias et al., 2022) encodes the source and target nodes separately and decodes them to reconstruct the directed edges between pairwise nodes.

2.2 Hypergraph Representation Learning

As a generalization of graphs, hypergraphs describe real-world data with more appropriate and delicate representations. Since the introduction of linear Laplacian in the seminal work(Zhou et al., 2006), many works have applied it to the processing of hypergraph structure data(Ji et al., 2020; Zu et al., 2016; Wang et al., 2015).Feng(Feng et al., 2019) proposed the HGNN framework and defined the spectral convolution on the hypergraph to obtain the representation of the hypergraph structure. Dong(Dong et al., 2020) focused on hyperedge representation and formulated a framework that convolves and normalizes hypernodes and hyperedges respectively. Gao(Gao et al., 2022) extended the HGNN framework and proposed a spatial hypergraph convolution pattern to achieve more powerful expressions from different hypergraph structures. The methods mentioned above are all carried out on undirected hypergraphs making it inevitable to lose critical information in real-world modeling.

2.3 Link Prediction

Link prediction is an important task in network structure analysis, there are usually three link prediction methods: 1) Similarity-based methods: scoring by measuring pairs of nodes' similarity(Liben-Nowell and Kleinberg, 2003; Liu and Lü, 2010); 2) Statistics-based methods: common methods include maximum likelihood method and probability relationship model(Clauset et al., 2008); 3) Graph representation learning methods: learning edge representations and node representations are widely used method currently. The Node2vec(Grover and Leskovec, 2016) takes the lead in applying feature learning algorithms to link prediction and examining that node embeddings outperform previous heuristic scores. GCNs and their faction methods thriving recently makes the representation of the node more comprehensive and achieves excellent results in link prediction(Zhang and Chen, 2018).

3 PRELIMINARIES

A directed hypergraph(Gallo et al., 1993) $G = (V, E, W)$ consists of a finite set of vertices V, together with a set of hyperarcs E. For each hyperarc $e = (e^{tail}, e^{head})$ in E, e^{tail} and e^{head} are the tail of hyperarc e and the head of hyperarc e, respectively. We note that e^{tail} can contain more than one vertices, where $e^{tail} = \{v_1, v_2, ..., v_i\}$ for $i \geq 1$, as well as $e^{head} = \{v_1, v_2, ..., v_j\}$ for $j \geq 1$. $W \in \mathbb{R}^{|E|\times|E|}$ denotes the diagonal matrix which contains the weights of hyperarcs. Each hyperarc e can associate a positive value $w(e)$. A directed hypergraph is a generalization of a directed graph, such that both ends of an arc can contain an arbitrary nonzero number of vertices.

The directed hypergraph G can be represented by two incidence matrices $H^{tail} \in \{0,1\}^{|V|\times|E|}$ and $H^{head} \in \{0,1\}^{|V|\times|E|}$, with entries defined as:

$$h^{tail}(v, e^{tail}) = \begin{cases} 1, & \text{if } v \in e^{tail} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$h^{head}(v, e^{head}) = \begin{cases} 1, & \text{if } v \in e^{head} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

As previously defined, we can define the out-degree $d^{tail}(v)$ and in-degree $d^{head}(v)$ as: $d^{tail}(v) = \sum_{e\in E} w(e)h^{tail}(v, e^{tail})$, $d^{head}(v) = \sum_{e\in E} w(e)h^{head}(v, e^{head})$. Similarly, for an hyperarc $e \in E$, it has out-degree $d^{tail}(e)$ and in-degree $d^{head}(e)$, which is defined by $d^{tail}(e) = \sum_{v\in V} h^{tail}(v, e^{tail})$ and $d^{head}(e) = \sum_{v\in V} h^{head}(v, e^{head})$. The diagonal matrix $D_v^{tail} \in \mathbb{R}^{|V|\times|V|}$ and $D_v^{head} \in \mathbb{R}^{|V|\times|V|}$, $D_e^{tail} \in \mathbb{R}^{|E|\times|E|}$ and $D_e^{head} \in \mathbb{R}^{|E|\times|E|}$ represents the vertex degree and hyperarc degree respectively.

4 DIRECTED HYPERGRAPH CONVOLUTION

In this section, we introduce the random walk to the directed hypergraph and formulate the regularization framework. For the purpose of reducing the computational complexity of downstream tasks, we further give the transductive inference of directed hypergraph Laplacian by analyzing the approximate digraph Laplacian properties. Finally, we provide the definition of directed hypergraph convolution based on the hypergraph convolution.

4.1 Random Walk On Directed Hypergraph

We can consider the problem from the perspective of a random walk on a directed hypergraph. Similar to (Ducournau and Bretto, 2014), given the current location $u \in V$, first select a hyperarc $e = (e^{tail}, e^{head})$ incident with vertex $u \in e^{tail}$ randomly, and then choose a vertex $v \in e^{head}$. From this, the probability $p(u, v)$ can be formulated as

$$p(u,v) = \sum_{e \in E} w(e) \frac{h^{tail}(u, e^{tail})}{d^{tail}(u)} \frac{h^{head}(v, e^{head})}{d^{head}(e)} \quad (3)$$

Let $P \in \mathbb{R}^{|V| \times |V|}$ denote the transition probability matrix, which is represented as $P = {D_v^{tail}}^{-1} H^{tail} W {D_e^{head}}^{-1} H^{head^T}$.

When a directed graph is strongly connected and aperiodic, the random walk converges to a unique stationary distribution(Zhou et al., 2005). Generalizing the definition of strongly connected digraphs to directed hypergraphs, for any pair of vertices $i, j \in V$, if there has at least one directed path from vertex i to j, the directed hypergraph is strongly connected(Allamigeon, 2011). Directed hypergraphs modeled from real-world data are not necessarily strongly connected, in order to satisfy this property, we consider adding an auxiliary vertex ψ(Tong et al., 2020), as the personalized PageRank teleport set, which incident with two hyperarcs, in-hyperarc e_{ψ_1} and out-hyperarc e_{ψ_2}, respectively. For the hyperarc $e_{\psi_1} = (e_{\psi_1}^{tail}, e_{\psi_1}^{head})$, its head contains one vertex ψ and the tail contains all vertices in V. And for $e_{\psi_2} = (e_{\psi_2}^{tail}, e_{\psi_2}^{head})$, its tail contains one vertex ψ and the head contains all vertices in V. In addition, we add self-loops to the original directed hypergraph to ensure aperiodic. Similar to directed graphs(Zhou et al., 2005) and hypergraphs(Zhou et al., 2006), the transition probability p of a strongly connected and aperiodic directed hypergraph $G = (V, E, W)$ has a unique stationary distribution:

$$\pi(v) = \sum_{u \to v} \pi(u) p(u, v) \quad (4)$$

Let S denote a vertex subset of V and S^c denote the complement of S. Define the volume of S by $volS = \sum_{v \in S} \pi(v)$, obviously, $volS$ is the probability that random walk occupies vertices in S, and $volS + volS^c = volV = 1$. Further, define the out-hyperarc boundary $\partial S = \left\{ e \in E | e^{tail} \subset S, e^{head} \subset S^c \right\}$ of S, the volume of ∂S is define as $vol\partial S = \sum_{e \in \partial S, [u,v] \in e} \pi(u) p(u, v)$, and the in-hyperarc boundary of S is represented by $\partial S^c = \left\{ e \in E | e^{tail} \subset S^c, e^{head} \subset S \right\}$. Clearly, the volume of ∂S is the probability with which a step from S to ∂S, similar to $vol\partial S^c$.

A hyperarc can be transformed into a set of directed edges by star expansion, where each vertex in the e^{tail} is connected with all vertices in the e^{head} through directed edges. This transformation yields an equivalent directed graph for the random walk on the directed hypergraph, and consequently, $vol\partial S = vol\partial S^c$ holds in the directed hypergraph, as proved in the directed graph(Zhou et al., 2005). With that, we can formalize the cut as

$$\arg\min_{x \in S} c(S) = vol\partial S \left(\frac{1}{volS} + \frac{1}{volS^c} \right) \quad (5)$$

where the fraction $\frac{vol\partial S}{volS}$ is the probability of a random walk leaving S along a hyperarc in ∂S, similar to the ratio $\frac{vol\partial S^c}{volS^c}$. Note that equation(5) is a generalization of the normalized cut criterion for directed graphs(Zhou et al., 2005). The Optimization objective aims to obtain a partition that maximizes the probability of staying in the same cluster and minimizes the probability of crossing different clusters as possible.

We can relax equation (5) into a real-valued optimization problem as follows:

$$\begin{aligned} &\arg\min_f \Omega(f) = \sum_{e \in E} \sum_{[u,v] \in e} \frac{\pi(u) p(u,v) \left(\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}} \right)^2}{2} \\ &subject\ to \sum_{v \in V} f^2(v) = 1, \sum_{v \in V} f(v) \sqrt{\pi(v)} = 0 \end{aligned} \quad (6)$$

we can formulate the regularization framework which is in keeping with Tran et al.(Tran and Tran, 2020):

$$\arg\min_f \Omega(f) + R_{srm}(\theta) \quad (7)$$

where the $\Omega(f)$ denotes the regularization term with a classification function f, and R_{srm} is defined as the empirical loss and parameter regularization. Let $\Theta = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}}{2}$,$\Delta = I - \Theta$, where I denotes the identity matrix, the numerator of the optimization problem (6) can be verified to $\Omega(f) = 2 f^T \Delta f$.

Therefore, the solution to the optimization problem can be solved as a general eigendecomposition. Similar to Zhou(Zhou et al., 2006), We can define the Laplacian for a directed hypergraph as follows

$$\Delta = I - \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}}{2} \quad (8)$$

where $\Pi \in \mathbb{R}^{|V| \times |V|}$ is a diagonal matrix with diagonal elements π.

4.2 Directed Hypergraph Convolution

From section 4.1, we need to ensure that the directed hypergraph is strongly connected and aperiodic. As previously mentioned, we can add an auxiliary vertex

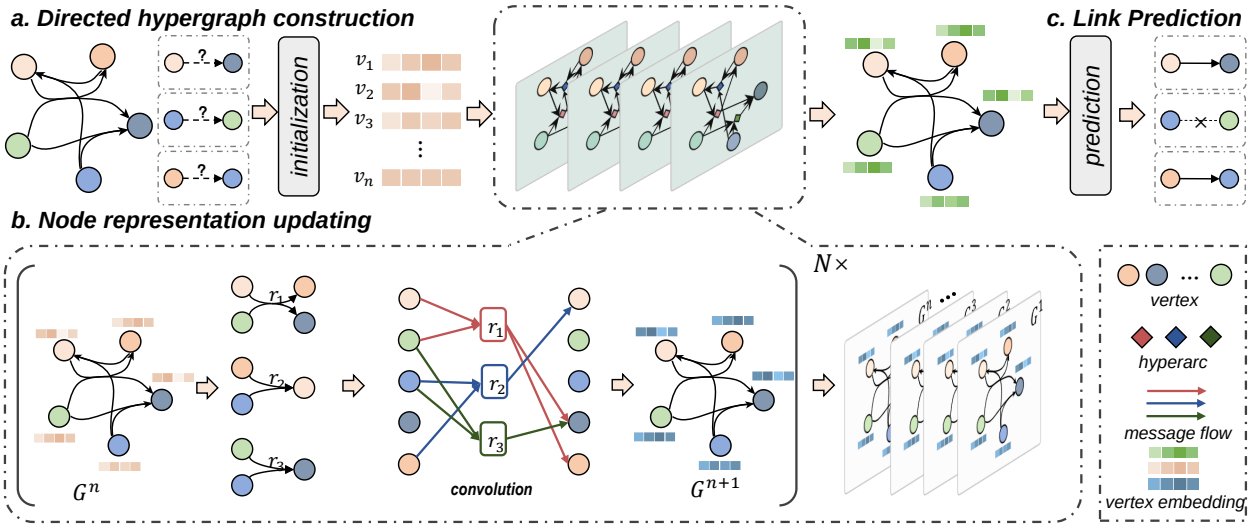

Figure 2: The architecture of the proposed DHGNN framework. (a) Directed hypergraph construction from data correlations, and latent features generation by an initialization function. The goal is to foresee the links between the given nodes. (b) Node representation learning. The node representations are updated by message flow and aggregation and are further fused through the fusion function. (c) Predict if there is a link between pairwise nodes from their representations.

ψ with two hyperarcs e_{ψ_1} and e_{ψ_2}, and add self-loops for all vertices to satisfy the above two properties. Let α denote the probability that steps to the auxiliary vertex(Tong et al., 2020). Then the transition matrix is defined as

$$\hat{P} = \begin{bmatrix} (1-\alpha)\tilde{P} & \alpha \\ \frac{1}{|V|} & 0 \end{bmatrix} \tag{9}$$

where $\hat{P} \in \mathbb{R}^{(|V|+1)\times(|V|+1)}$, and $\tilde{P}$ is the transition matrix of the original directed hypergraph with added self-loops. We divide the random walk into two events: walking on the original directed hypergraph with a probability of $1-\alpha$, and stepping to the auxiliary node with a probability of α. $\frac{1}{|V|}$ indicates that the probability of teleporting to each vertex on the directed hypergraph is equal. The matrix $\hat{P}$ is aperiodic and irreducible, which has a unique stationary distribution $\hat{\pi}$. Thus, we have $\hat{\pi} = (\pi_{\tilde{P}}, \pi_{\psi})$. Where $\pi_{\tilde{P}}$ represent the stationary distribution of the original n vertices, and the second part π_{ψ} is the stationary distribution of the auxiliary vertex ψ. With this, we can adjust the value of the transition probability to control the effect of the auxiliary vertex. By adjusting the probability $\alpha \to 0$, the first part $\pi_{\tilde{P}}$ approximates the stationary distribution of the original directed hypergraph.

In conclusion, the directed hypergraph Laplacian can be developed as

$$\begin{aligned} L \approx I - \frac{1}{2}(&\Pi_{\tilde{P}}^{1/2} D_v^{tail^{-1}} H^{tail} W D_e^{head^{-1}} H^{head^T} \Pi_{\tilde{P}}^{-1/2} \\ &+ \Pi_{\tilde{P}}^{-1/2} (D_v^{tail^{-1}} H^{tail} W D_e^{head^{-1}} H^{head^T})^T \Pi_{\tilde{P}}^{1/2}) \end{aligned} \tag{10}$$

where $\Pi_{\tilde{P}} \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix. With the spectral convolution analysis on hypergraph(Feng et al., 2019), we can follow the derivation to derive the convolution of directed hypergraph:

$$Y = \sigma(L\mathcal{E}\Theta) \tag{11}$$

where $\mathcal{E} \in \mathbb{R}^{|V|\times C_1}$ denotes the vertex embedding and $\Theta \in \mathbb{R}^{C_1\times C_2}$ is the learnable weight matrix. Y is the representation generated from the convolution operation. $\sigma(\cdot)$ is the alternative non-linear activation function.

The above convolution can be explained in a more intuitive way (as shown in Figure 2(b)). Specifically, the information of vertices in the tail set is aggregated according to the connection method of hyperarcs and then propagated to the vertices pointed to in the head set.

5 FRAMEWORK FOR LINK PREDICTION

In this section, we scheme the architecture for directed hypergraph structure representation learning. In accordance with what we have already mentioned, we

develop a novel model DHGNN which consists of three key modules: 1) Embedding learning on directed hypergraph; 2) Multi-scale embedding fusion; 3) Negative sampling and prediction. The framework is illustrated in Figure 2.

5.1 Embedding Learning

Initialization. Given a directed hypergraph $G = (V, E, W)$ with N vertices and M hyperarcs modeled from observed data. Unless otherwise specified, the weights of hyperarcs below are set to $w = 1$. Formally, we use $Z \in \mathbb{R}^{N \times d}$ to represent the initial node embedding, containing the explicit features and latent features. Following the mainstream latent features generate methods, we draw the embedding table from a prescribed distribution $g(\cdot)$, which could jointly optimize with other parameters. Then the initial node embedding can be obtained by:

$$z_i^{(0)} \sim g(z_i^{(0)}) \tag{12}$$

where $z_i^{(0)} \in \mathbb{R}^d$ denotes the initialized node embedding, which depicts nodes in a low-dimension latent representation space. Based on this, we model the complex dependencies among each node and work out a strategy to aggregate information through convolution on the directed hypergraph structure, which obtains informative and accurate representations for nodes.

High-order message flow. To capture the message passing direction, we inject the directed hypergraph convolution into directed high-order relation encoding, which is defined by

$$Z^{(l)} = LZ^{(l-1)}\Theta^{(l-1)} \tag{13}$$

where $\Theta^{(l-1)}$ is trainable weight parameter, $Z^{(l)}$ is node embeddings, L is the approximate Laplacian defined in section 4. It is worth noticing that we remove the nonlinear activation, since the nonlinearity in the convolution layer is non-essential and even causes a negative effect(Wu et al., 2019; He et al., 2020).

5.2 Multi-scale Embedding Fusion

The embedding may exhibit diverse transitional patterns over different message-passing layers, and representation transformation between different layers may lead to information loss. To tackle the aforementioned issue, we integrate the diverse transitional patterns into a common latent space and aggregate the multi-granularity representation, so as to endow the model with the capability of encoding multi-dimensional semantics expression. The multi-scale embedding fusion presents as follows:

$$Z^* = h(Z^{(0)}, Z^{(1)}, ..., Z^{(l)}) \tag{14}$$

where $h(\cdot)$ is the fusion function such as concatenate,point-wise addition, etc. It is evident that deep embeddings have finer granularity than shallow embeddings. Aggregating multiple layers of embeddings enables multi-grained representations.

5.3 Prediction and Optimization

Prediction. Given the obtained embedding z_i^* and z_j^*, the predicted score is computed by the inner product between the representation of the source and target entities:

$$r_{i \to j} = {z_i^*}^T z_j^* \tag{15}$$

Optimization. Typically, under the condition that negative interactions in link prediction tasks are invisible, representations are generated based on existing revealed relations. In the absence of labeled negative samples, the embedding neglects the commonality of negative samples, leading to prediction deviations. For example, in social networks, users may distrust or dislike others, and in chemical reactions, certain substances cannot react. Thus, a heuristic negative sampling strategy is adopted in this paper. We sample negatives as latent feedback to assist model learning and introduce BPR(Rendle et al., 2012) loss as an optimization criterion:

$$\mathcal{L} = -\sum_{(i,j^+,j^-) \in T} -ln\sigma(r_{i \to j^+} - r_{i \to j^-}) + \lambda ||\Theta||_2^2 \tag{16}$$

where T indicates the pairwise training data, which is assumed that node i to j^+ is positive feedback, and i to j^- is negative feedback.

6 EXPERIMENTS

In our evaluation, we examine the performance of our proposed DHGNN in multi-tasks and datasets[1].

6.1 Experimental Settings

6.1.1 Datasets

To demonstrate the effectiveness of our proposed framework, we conduct experiments on several open-access digraph datasets of varied fields. We construct forward hyperarcs(Gallo et al., 1993) based on original directed edges and related semantic information. The dataset information is shown in Table 1.

6.1.2 Baselines

We compare our model with state-of-the-art models in three different categories, including: Undirected

[1]https://github.com/mazitong/DHGNN

Table 1: Datasets statistics.

Dataset	Category	Nodes	Edges
Air	Preferred Routes	1226	2615
Survey	Social network	2539	12969
Cora	Citation network	2708	5429
Citeseer	Citation network	3312	4715
AM-computer	Co-purchase	13752	287209

graph network method, GCN(Kipf and Welling, 2016), GAT(Veličković et al., 2017), GraphSAGE(Hamilton et al., 2017), PNA(Corso et al., 2020); Undirected hypergraph learning method, HGNN(Feng et al., 2019), HNHN(Dong et al., 2020), HGNN$^+$(Gao et al., 2022); Digraph network method including DiGCN(Tong et al., 2020), DiGAE(Kollias et al., 2022), D-HYPR(Zhou et al., 2022).

6.1.3 Task and Evaluation Metrics.

We conduct our model with the following tasks: Link Prediction with explicit features and latent features.

We construct a directed hypergraph to learn node feature representations and examine the performance of our proposed model by predicting whether there is a relationship between pairwise nodes. We randomly mask 20% of the links from the original dataset as the test set and retain 80% of the links for training which is a basis of directed hyperarc construction. Furthermore, we use a uniform distribution to generate initial features for latent feature learning.

Evaluation metrics are the Area under the Curve of ROC (AUC) and Average Precision(AP) for link prediction.

6.1.4 Parameter Settings

To ensure the fairness and validity of the comparative experiments, we tuned the hyperparameters of all models to achieve the best baseline results. The hyperparameters are tuned by grid search. Since the dataset with different distributions will affect the performance of the model, we randomly divide the dataset into diverse splits and obtain the average of the multiple experimental results. In all cases, models are performed by full-batch gradient descent, optimized with Adam.

6.2 Performance Comparison

In this subsection, we discuss the evaluation results for compared approaches.

Link prediction with explicit features. The experimental results of link prediction with node explicit features are summarized in Table 2, we conduct experiments on two citation networks, and the DHGNN consistently outperforms the baselines. It can be found that the results of hypergraph learning methods and directed graph methods are improved compared to undirected graph methods, due to the rich and accurate semantics in complicated structures. Based on experimental results, it can be demonstrated that the utilization of directed hypergraph methodology amalgamates the strengths of both hypergraphs and directed graphs, leading to a significant performance enhancement.

Table 2: Performance comparison on Link Prediction task with explicit features, in terms of AUC and AP averaged over 100 runs. The best results are in **bold** and the second are in underline. The data in the table are presented as percentages(e.g., 91.92%).

	Cora		Citeseer	
Model	AUC	AP	AUC	AP
GCN	91.92	90.23	88.29	89.18
GAT	91.29	91.10	82.23	84.37
GraphSAGE	93.03	93.01	87.42	88.77
PNA	93.43	93.15	88.26	89.63
HGNN	92.89	92.28	90.93	89.54
HNHN	92.22	92.32	88.44	89.45
HGNN$^+$	93.62	93.11	92.04	92.49
DiGCN	93.28	93.22	92.12	92.46
DiGAE	93.68	93.27	90.79	89.88
Ours	**94.27**	**93.40**	**93.87**	**93.51**
Improve	0.63%	0.14%	1.90%	1.10%

Link prediction with latent features. In this experiment, we take 64-dimensional embeddings as input, and as shown in Table 3, our model outperforms all baselines in both AUC and AP metrics on five datasets, with the highest improvement reaching 5.6854%, demonstrating its impressive effectiveness. Without relying on explicit features, our method effectively learns latent node features with rich structural information and achieves strong predictive performance.

It can be observed that the undirected graph methods lack effectiveness on directed datasets because of aiming at undirected graphs. Although hypergraph approaches are designed to extract hypergraph high-order features, it has inherent drawbacks to directional information learning. Moreover, the directed graph methods preserve the direction signals, however, these models have limited ability to capture higher-order structural information. Directed graph and hypergraph methods outperform undirected graph methods

Table 3: Performance comparison on Link Prediction task with 64-dimensional latent features, in terms of AUC and AP averaged over 100 runs. The best results are in **bold** and the second are in underline. The data in the table are presented as percentages(e.g., 74.62%).

	Cora		Citeseer		Air		Survey		AM-Computers	
Model	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
GCN	74.62	76.27	67.94	70.30	71.43	73.11	86.17	87.23	90.38	89.56
GAT	82.58	84.60	74.01	77.51	80.24	81.46	91.05	91.65	91.14	90.41
GraphSAGE	85.72	83.97	75.65	76.62	82.03	79.19	87.35	85.38	93.17	93.27
PNA	86.24	86.73	75.48	75.92	84.38	83.85	89.97	88.91	93.19	93.02
HGNN	87.29	86.25	77.42	79.08	85.14	84.58	87.60	86.41	92.39	91.47
HNHN	85.81	84.70	77.83	78.56	83.57	80.50	87.47	85.92	93.18	93.39
HGNN$^+$	90.12	90.70	79.54	81.91	87.14	85.87	88.94	88.03	90.97	89.38
DiGCN	87.67	87.57	77.07	80.22	87.30	88.32	90.18	89.86	93.20	92.18
D-HYPR	88.10	90.70	76.85	82.65	86.46	87.43	91.95	92.96	93.10	92.73
DiGAE	82.67	83.28	72.53	75.15	80.21	79.97	86.17	83.96	91.85	84.96
Ours	**91.95**	**93.38**	**84.06**	**86.37**	**89.13**	**89.15**	**93.20**	**93.65**	**95.81**	**95.18**
Improve	2.09%	0.95%	2.04%	2.94%	1.37%	0.73%	2.81%	1.92%	5.68%	4.51%

in most evaluation cases. These observations indicate that learning by incorporating both orientation and higher-order information is compelling. Our method is more effective than other methods for efficient modeling and embedding learning on real-world data.

Fusion method. We conducted experiments on the fusion function of the fusion module to compare the effectiveness of different fusion methods. Following the approach in GIN(Xu et al., 2018), we employed mean pooling, max pooling, sum pooling, and concatenation to integrate embeddings from different convolution layers. The experimental results are reported in Table 4, where it can be observed that the concatenation method yielded the best performance. Concatenation preserves the original embedding information to a greater extent compared to the other fusion methods, making it the most informative. Mean pooling may lose some specific, non-average features as it treats all features equally important. Max pooling only retains the maximum values, which could be effective in certain scenarios, particularly when only the strongest or most significant features are of interest, however, it inevitably entails information loss. Sum pooling aggregates all embedding values to obtain a new embedding vector. While this method preserves information from all features, it overlooks variations in importance across different features. Hence, the concatenation function serves as the embedding fusion method in the experiments conducted within this paper.

6.3 Embedding Size Sensitivity

Figure 3 shows the link prediction results of DHGNN in 4 to 64-dimensional initial node embeddings. We can draw a conclusion that the overall performance is positively correlated to the size of the embedding dimension. As the embedding dimension and the amount of information contained increase, the prediction performance improves. However, At high dimensional embedding sizes such as 32 and 64, the performance improvement rate is reduced and even shows negative growth on some datasets due to the model overfitting from a large amount of information. In general, our method keeps stable when in most dimension embeddings.

Table 4: Performance comparison of different fusion functions in the link prediction task of implicit features(Concat represents the concatenate fusion function). The best results are in **bold** and the second are in underlined. The data in the table are presented as percentages(e.g., 89.44%).

	Cora		Citeseer	
Model	AUC	AP	AUC	AP
Mean	89.44	91.65	82.20	84.38
Max	89.32	89.57	82.93	83.52
Sum	90.21	91.34	83.47	85.11
Concat	**91.95**	**93.38**	**84.06**	**86.37**

6.4 Ablation Study

We analyze the effectiveness of the embedding fusion component by conducting experiments on link prediction with latent features. As shown in Figure 4(a), the model with fusion operation outperforms the model

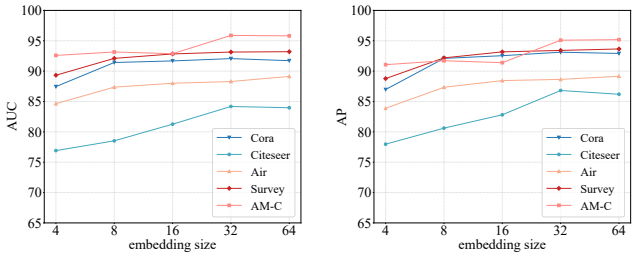

Figure 3: Results of different embedding dimensions.

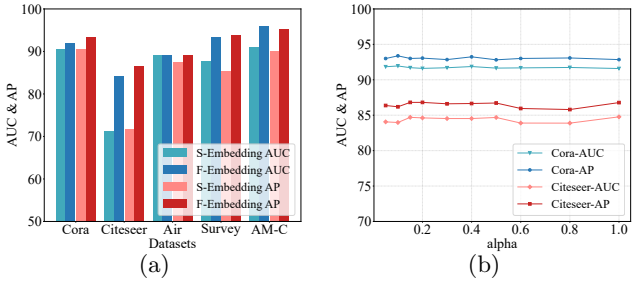

Figure 4: (a) The result of examining the fusion module. F-embedding is to keep the fusion module, and S-Embedding is to remove the fusion module. (b) Impact of α to AUC and AP.

only with deep layer embedding over all datasets, especially the improvement on Citeseer is high to 12.96%. We attribute the improvement to the multi-scale information aggregation during convolution, which the single deep embedding does not consider. Such fusion can prevent the current information from being diluted during the message-passing process, leveraging different granular information thus achieving effective learning.

6.5 Parameter Analysis

We investigate the sensitivity of the transition probability α. As Figure 4(b) shows, the experimental performance on the citation datasets is most prominent and relatively stable when α ranges from $[0.1, 0.5]$. Referring to the previously mentioned in section 4, the probability of a node transferring to the auxiliary node will grow with the increase of α, and the approximation to the original directed hypergraph will be reduced. Therefore, we recommend keeping lower α in the range of $[0.1, 0.2]$ to retain structural information and ensure performance.

7 CONCLUSION

In this study, we develop a framework DHGNN for link prediction on directed hypergraph structure, which provides a solution for directed hypergraph representation learning. DHGNN is capable to handle more abundant and complex correlations compared with normal graphs and hypergraphs. We demonstrate the effectiveness of the proposed DHGNN in multiple experiments, which is significantly competitive with state-of-the-art models.

In the future, we will explore spatial convolution patterns for directed hypergraphs. Furthermore, we will investigate combining our model with existing models to apply it in more complex scenarios.

Acknowledgements

This research was partially supported by the NSFC (62376180, 62176175), Industry-University Cooperation Collaborative Education Project of the Ministry of Education of China (220606363154256), the Major Project of Natural Science Research in Universities of Jiangsu Province (21KJA520004), Suzhou Science and Technology Development Program(SYG202328), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

Allamigeon, X. (2011). Strongly connected components of directed hypergraphs. *CoRR, abs/1112.1444*.

Ausiello, G. and Laura, L. (2017). Directed hypergraphs: Introduction and fundamental algorithms—a survey. *Theoretical Computer Science*, 658:293–306.

Bahmani, B., Chowdhury, A., and Goel, A. (2010). Fast incremental and personalized pagerank. *arXiv preprint arXiv:1006.2880*.

Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Velickovic, P. (2020). Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 13260–13271.

Dong, Y., Sawin, W., and Bengio, Y. (2020). Hnhn: hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*.

Ducournau, A. and Bretto, A. (2014). Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding*, 120:91–102.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. (2019). Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565.

Gallo, G., Longo, G., Pallottino, S., and Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201.

Gao, Y., Feng, Y., Ji, S., and Ji, R. (2022). Hgnn $^+$: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Ji, S., Feng, Y., Ji, R., Zhao, X., Tang, W., and Gao, Y. (2020). Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2020–2029.

Khosla, M., Leonhardt, J., Nejdl, W., and Anand, A. (2020). Node representation learning for directed graphs. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 395–411. Springer.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kollias, G., Kalantzis, V., Idé, T., Lozano, A., and Abe, N. (2022). Directed graph auto-encoders. In *AAAI Conference on Artificial Intelligence*.

Kunegis, J. (2013). Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*, pages 1343–1350.

Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559.

Liu, W. and Lü, L. (2010). Link prediction based on local random walk. *EPL (europhysics Letters)*, 89(5):58007.

Moody, J. (2001). Peer influence groups: identifying dense clusters in large networks. *Social networks*, 23(4):261–283.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Rossi, R. and Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Twenty-ninth AAAI conference on artificial intelligence*.

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

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., and Lim, A. (2020). Digraph inception convolutional networks. *Advances in neural information processing systems*, 33:17907–17918.

Tran, L. H. and Tran, L. H. (2020). Directed hypergraph neural network. *arXiv preprint arXiv:2008.03626*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, M., Liu, X., and Wu, X. (2015). Visual classification by ℓ_1-hypergraph modeling. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2564–2574.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Xiao, G., Liao, J., Tan, Z., Yu, Y., and Ge, B. (2022). Hyperbolic directed hypergraph-based reasoning for multi-hop kbqa. *Mathematics*, 10(20):3905.

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

Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018). Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28.

Zhang, M. and Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhou, D., Huang, J., and Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, pages 1036–1043.

Zhou, D., Huang, J., and Schölkopf, B. (2006). Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19.

Zhou, H., Chegu, A., Sohn, S. S., Fu, Z., De Melo, G., and Kapadia, M. (2022). D-hypr: Harnessing neighborhood modeling and asymmetry preservation for digraph representation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2732–2742.

Zu, C., Gao, Y., Munsell, B., Kim, M., Peng, Z., Zhu, Y., Gao, W., Zhang, D., Shen, D., and Wu, G. (2016). Identifying high order brain connectome biomarkers via learning on hypergraph. In *International Workshop on Machine Learning in Medical Imaging*, pages 1–9. Springer.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable] Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable] Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Yes
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Yes
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable] Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable] Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable] Yes
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Yes
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Yes

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable] Not Applicable

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not Applicable

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not Applicable

Directed Hypergraph Representation Learning for Link Prediction

A SUPPLEMENTARY EXPERIMENTS

Semi-supervised Node Classification. Table 5 shows the node classification results on Citation networks. We report the average ACC(accuracy) of 100 runs on each dataset. Our method is determined to surpass other methods on two datasets. As we can see, the hypergraph and digraph methods perform better compared to undirected graphs since the datasets contain direction information and high-order information. DHGNN is prone to transmit this information, which shows satisfactory suitability for complex datasets.

Table 5: Performance comparison on Node Classification in terms of ACC averaged over 10 random dataset splits. The best results are highlighted with **bold** and the second are in underline.

Model	Cora	Citeseer
GCN	79.34%	70.38%
GAT	80.12%	70.76%
GraphSAGE	78.88%	68.26%
HGNN	81.20%	71.20%
HNHN	79.40%	68.86%
HGNN$^+$	80.62%	70.30%
DiGCN	80.58%	70.78%
Ours	**83.08%**	**75.56%**
Improve	2.32%	6.12%

Training Time. We compare the training speed of each model under the same training conditions and report the time cost of training. Table 6 summarizes the training time per epoch. Our method performs well compared to most methods, our model training time is insignificantly different from most methods on small datasets, but shows advantages on large datasets. Notice that the DiGAE is much faster than other methods because the encoder and decoder are designed efficiently, there are fewer elements in the matrix multiplication, and the operation time is considerably shortened. Despite low training time cost, DiGAE is barely satisfactory in prediction with the latent feature.

Table 6: Average training time cost per epoch in second(s).

Model	Cora	Citesser	Air	Survey	AM-Computers
GCN	0.42	0.43	0.17	0.78	16.33
GAT	0.53	0.57	0.22	0.91	18.98
GraphSAGE	0.41	0.44	0.17	0.76	16.75
HGNN	0.44	0.46	0.17	0.76	16.25
HNHN	0.43	0.45	0.18	0.77	16.45
HGNN$^+$	0.42	0.46	0.18	0.80	17.09
DiGCN	0.53	0.55	0.25	0.91	20.65
D-HYPR	1.79	2.02	0.54	1.70	34.50
DiGAE	0.04	0.06	0.02	0.08	1.70
Ours	0.37	0.44	0.13	0.51	10.34

Convergence. Figure 5 shows the convergence curve of DHGNN on all datasets. From the results, we can observe that the loss function of DHGNN achieves stability requires about 60 epochs.

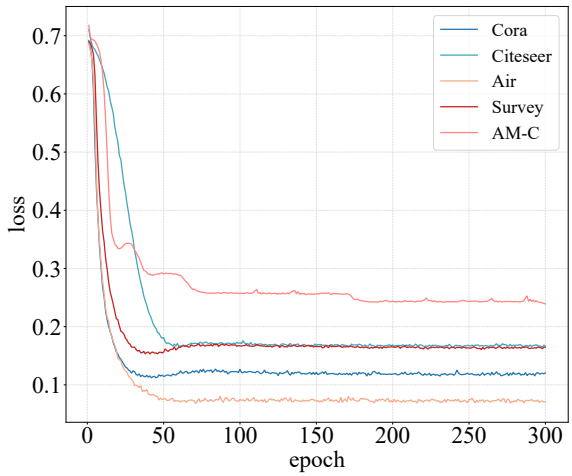

Figure 5: Convergence speed curves of loss function on each dataset.

B HYPERPARAMETERS SETTINGS

The hyperparameters is tuned by grid search: learning rates $lr \in \{0.1, 0.01, 0.001\}$, weight decays $\omega \in \{0.0001, 0.0003, 0.0005\}$, dropout rates $dp \in \{0.1, 0.3, 0.5, 0.9\}$. Moreover, model-specific critical hyperparameters are taken into account as well. For GAT, the number of attention heads is from $h \in \{4, 8\}$; for D-HYPR, the k-order proximity is tuned from $\{1, 2, 3\}$ and λ is from $\{0.5, 1, 1.5\}$; for DiGCN, the teleport probability in $\{0.05, 0.1, 0.15\}$ and the k^{th}-order proximity is in $\{1, 2, 3\}$; for DiGAE, the degrees in message passing from $\{0.0, 0.2, 0.4\}$.

C DIRECTED HYPERGRAPH CONSTRUCTION

We provide the reference of the dataset and the construction approach of the directed hypergraph. Air(Kunegis, 2013) is aviation routes. We define the airport as a node, the flight route as a hyperarc, e^{tail} contains a departure airport, and the e^{head} contains multiple arrival airports. Survey(Moody, 2001) is a social network. We take the user as the node, the trust relationship as the hyperarc, e^{tail} as a user, and e^{tail} as all other users set trusted by the user. Cora(Sen et al., 2008) and Citeseer(Rossi and Ahmed, 2015) are citation network datasets. We portray articles as nodes, citation relationships as hyperarcs, e^{tail} as one article, and e^{head} as all the articles cited by it. AM-computer(Shchur et al., 2018) is the purchase network. The items are defined as nodes, the purchase orders are defined as hyperarcs, e^{tail} contains a currently purchased product, and e^{head} contains the products purchased immediately after it.

D DERIVATION

In this section, we present the detailed proof of Section 4.1.

In Section 4.1, we provide the equation $vol\partial S = vol\partial S^c$, the following is the proof process:

We know that $vol\partial S = \sum_{e \in \partial S, [u,v] \in e} \pi(u) p(u, v)$ and $vol\partial S^c = \sum_{e \in \partial S^c, [u,v] \in e} \pi(u) p(u, v)$, thus the problem is to verify that the two formulas are equal. First, assume a vertex $v \in S$, the probability of one step random walk to S^c is

$$\sum_{v \to u} \pi(v) p(v, u) = \pi(v) \sum_{v \to u} p(v, u) = \pi(v),$$

and the probability that from S^c to vertex v is

$$\sum_{u\to v}\pi(u)p(u,v)=\pi(v).$$

It is easy to see that $\sum_{v\to u}\pi(v)p(v,u)=\sum_{u\to v}\pi(u)p(u,v)=\pi(v)$, Then for all vertices in S we have that

$$\sum_{v\in S}(\sum_{v\to u}\pi(v)p(v,u))=\sum_{[u,v]\in\partial S}\pi(u)p(u,v),$$

$$\sum_{v\in S}(\sum_{u\to v}\pi(u)p(u,v))=\sum_{[u,v]\in\partial S^c}\pi(u)p(u,v),$$

$$\sum_{v\in S}(\sum_{v\to u}\pi(v)p(v,u)-\sum_{u\to v}\pi(u)p(u,v))=\sum_{[u,v]\in\partial S}\pi(u)p(u,v)-\sum_{[u,v]\in\partial S^c}\pi(u)p(u,v)=0.$$

As previously mentioned, from (Zhou et al., 2005), we can formalize the cut as

$$\arg\min_{x\in S}c(S)=vol\partial S(\frac{1}{volS}+\frac{1}{volS^c}). \tag{5}$$

which is a generalization of the normalized cut criterion for directed graphs. The Optimization objective aims to obtain a partition that maximizes the probability of staying in the same cluster and minimizes the probability of crossing different clusters as possible.

Define an indicator function h with that $h(v)=1$ if $v\in S$ and -1 otherwise. Let t denote the volume of S, we can written (5) as

$$\begin{aligned}c(S)&=\sum_{e\in E}\sum_{[u,v]\in e}\pi(u)p(u,v)\frac{(h(u)-h(v))^2}{4}(\frac{1}{t}+\frac{1}{1-t})\\&=\frac{\sum_{e\in E}\sum_{[u,v]\in e}\pi(u)p(u,v)(h(u)-h(v))^2}{4t(1-t)}.\end{aligned}$$

Then define another function g by $g(v)=2(1-t)$ if $v\in S$ and $-2t$ otherwise. We can find that $h(u)-h(v)=g(u)-g(v)$, and it is not hard to know that

$$\begin{aligned}\sum_{v\in V}\pi(v)g(v)&=\sum_{v\in S}\pi(v)g(v)+\sum_{v\in S^c}\pi(v)g(v)\\&=t\cdot 2(1-t)+(1-t)\cdot(-2t)\\&=0,\end{aligned}$$

$$\begin{aligned}\sum_{v\in V}\pi(v)g^2(v)&=\sum_{v\in S}\pi(v)g^2(v)+\sum_{v\in S^c}\pi(v)g^2(v)\\&=t\cdot 4(1-t)^2+(1-t)\cdot 4t^2\\&=4t(1-t).\end{aligned}$$

Moreover, the $c(S)$ may be written

$$c(S)=\frac{\sum_{e\in E}\sum_{[u,v]\in e}\pi(u)p(u,v)(g(u)-g(v))^2}{\sum_{v\in V}\pi(v)g^2(v)}.$$

Then we define function $f=\sqrt{\pi}g$, thus the above can be concluded that

$$\begin{aligned}&\arg\min_f\Omega(f)=\frac{\sum_{e\in E}\sum_{[u,v]\in e}\pi(u)p(u,v)(\frac{f(u)}{\sqrt{\pi(u)}}-\frac{f(v)}{\sqrt{\pi(v)}})^2}{2f(u)f(v)}\\&subject\ to\sum_{v\in V}f^2(v)=1,\sum_{v\in V}f(v)\sqrt{\pi(v)}=0.\end{aligned} \tag{6}$$

Let $\Theta = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P \Pi^{1/2}}{2}$,$\Delta = I - \Theta$, where I denotes the identity matrix, the numerator of optimization problem (6) can be verified to $\Omega(f) = 2f^T \Delta f$.

Divide the Formula into two parts, each part describes different directions of hyperarc.

$$\begin{aligned}
&\frac{1}{2} \sum_{e \in E} \sum_{[u,v] \in e} \pi(u) p(u,v) (\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}})^2 \\
=&\frac{1}{2} \sum_{v \in V} \sum_{e \in E} [\sum_{u \to v} \pi(u) p(u,v) (\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}})^2 \\
&+ \sum_{v \to u} \pi(v) p(v,u) (\frac{f(v)}{\sqrt{\pi(v)}} - \frac{f(u)}{\sqrt{\pi(u)}})^2] \\
=&\frac{1}{2} \sum_{v \in V} \sum_{e \in E} \{[\sum_{u \to v} p(u,v) f^2(u) + \sum_{u \to v} \frac{\pi(u) p(u,v)}{\pi(v)} f^2(v) - 2 \sum_{u \to v} \frac{\pi(u) p(u,v)}{\sqrt{\pi(u)\pi(v)}} f(u) f(v)] \\
&+ [\sum_{v \to u} p(v,u) f^2(v) + \sum_{v \to u} \frac{\pi(v) p(v,u)}{\pi(u)} f^2(u) - 2 \sum_{v \to u} \frac{\pi(v) p(v,u)}{\sqrt{\pi(v)\pi(u)}} f(u) f(v)]\}.
\end{aligned}$$

Moreover, for the first term we can see that

$$\begin{aligned}
&\sum_{u \in V} \sum_{u \to v} p(u,v) f^2(u) = \sum_{u \in V} (\sum_{u \to v} p(u,v)) f^2(u) \\
=&\sum_{u \in V} f^2(u) = \sum_{v \in V} f^2(v),
\end{aligned}$$

$$\sum_{v \in V} \sum_{u \to v} \frac{\pi(u) p(u,v)}{\pi(v)} f^2(v) = \sum_{u \to v} \frac{\pi(v)}{\pi(v)} f^2(v) = \sum_{v \in V} f^2(v).$$

Similarly, for the second term, we have that

$$\sum_{v \in V} \sum_{v \to u} p(v,u) f^2(v) = \sum_{v \in V} f^(v),$$

$$\sum_{v \in V} \sum_{v \to u} \frac{\pi(v) p(v,u)}{\pi(u)} f^2(u) = \sum_{v \in V} f^2(v).$$

Thus, we rewrite the optimization function as

$$\begin{aligned}
\Omega(f) &= \frac{1}{2} (4f^2(v) - 2 \sum_{v \to u} \frac{\pi(v) p(v,u)}{\sqrt{\pi(v)\pi(u)}} f(u) f(v) - 2 \sum_{u \to v} \frac{\pi(u) p(u,v)}{\sqrt{\pi(u)\pi(v)}} f(u) f(v)) \\
&= 2 \sum_{v \in V} \left\{ f^2(v) - \frac{f(u)\pi(u)^{1/2} p(u,v) \pi(v)^{-1/2} f(v) + f^2(v) - f(u)\pi(u)^{-1/2} p(v,u) \pi(v)^{1/2} f(v)}{2} \right\}.
\end{aligned}$$

By denoting that with matrix notation, the problem reduces to

$$\Omega(f) = 2f^T \Delta f,$$

as desired.