DHMConv: Directed Hypergraph Momentum Convolution Framework

Wen-Bo Zhao
Soochow University
Suzhou, P. R. China
wbzhao@stu.suda.edu.cn

Zi-Tong Ma
Soochow University
Suzhou, P. R. China
ztma99@stu.suda.edu.cn

Zhe Yang*
Soochow University
Suzhou, P. R. China
yangzhe@suda.edu.cn

Abstract

Due to its capability to capture high-order information, the hypergraph model has shown greater potential than the graph model in various scenarios. Real-world entity relations frequently involve directionality, in order to express high-order information while capturing directional information in relationships, we present a directed hypergraph spatial convolution framework that is designed to acquire vertex embeddings of directed hypergraphs. The framework characterizes the information propagation of directed hypergraphs through two stages: hyperedge information aggregation and hyperedge information broadcasting. During the hyperedge information aggregation stage, we optimize the acquisition of hyperedge information using attention mechanisms. In the hyperedge information broadcasting stage, we leverage a directed hypergraph momentum encoder to capture the directional information of directed hyperedges. Experimental results on five publicly available directed graph datasets of three different categories demonstrate that our proposed DHMConv outperforms various commonly used graph and hypergraph models.

1 Introduction

Graph structures, owing to their capacity to intuitively represent the relationships among entities, have found extensive applications across diverse fields. In practical scenarios, it is not prudent to solely rely on isolated vertex feature vectors as the input for algorithms, as it disregards the interdependence among vertices. This is why, in the process of acquiring vertex embeddings, it is imperative to integrate the features of both the vertices and their neighboring vertices. By doing so, the resulting embeddings encompass the vertex features as well as the graph structure information, thereby facilitating subsequent tasks.

Graph Convolutional Neural Networks (GCNs) are among the most widely used techniques for generating vertex embeddings and can be broadly categorized into two groups: spatial-based and spectral-based GCNs. Spatial-based GCNs can be conceptualized as an information propagation process that achieves global information transfer via local information updates focused on a vertex and its neighboring vertices (Gilmer et al., 2017; Micheli, 2009; Niepert et al., 2016). Spectral-based GCNs are inspired by the Fourier transform in signal processing and achieve global information updates through the Laplacian matrix (Bruna et al., 2013; Defferrard et al., 2016; Kipf and Welling, 2016).

While the graph structure is an intuitive representation of binary relationships between vertices, its modeling capacity for relationships involving multiple entities is somewhat limited. Hypergraphs can be employed to model such relationships. A hyperedge in a hypergraph can contain multiple vertices, providing a lossless means of modeling higher-order relationships (Bretto, 2013). In recent years, there has been extensive research on machine learning methods based on hypergraphs, such as hypergraph transductive learning (Zhou et al., 2006), spectral hypergraph neural networks (Feng et al., 2019), and spatial hypergraph neural networks (Gao et al., 2022). These models have significantly enriched the theoretical foundations and practical applications of hypergraphs in machine learning (Fan et al., 2021; Chang et al., 2021; Jiang and Luo, 2022; Mercado et al., 2021).

In practical settings, entity relationships are often asymmetric, with directional attributes prevalent in

*Corresponding author. Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

many relationships such as sequential connections and causal connections (Xie et al., 2018; Chen et al., 2019; Liao et al., 2018). Traditional graph or hypergraph models are inadequate for capturing the sequence of events. To address this issue, scholars have proposed machine learning methods based on directed graph structures (Zhou et al., 2005; Tong et al., 2020; Kollias et al., 2022). These methods leverage the directionality of edges to guide the information flow during directed graph convolution, enabling the resulting vertex embeddings to encode both graph structure and edge directionality (Ma et al., 2019; Kampffmeyer et al., 2019). In order to obtain possible higher-order information and implicit directional information on graph structures, we have designed a directed hypergraph spatial convolution framework.

The main contributions of this paper are as follows:

- We have proposed a directed hypergraph convolution framework, DHMConv, which is capable of encoding high-order information of directed hypergraphs and flexibly encoding directional information of directed hyperedges.
- We optimized the process of hyperedge convolution using an attention mechanism, which enables the model to focus on specific hyperedges and vertices during the convolution operation.
- We propose a directed hypergraph momentum encoder to effectively capture the directionality information in directed hyperedges. This encoder further enhances the accuracy and stability of vertex information updates by modeling directional information in a refined manner.

Our study involved conducting link prediction, vertex classification, and ablation experiments on five datasets from different domains. The results conclusively demonstrate that the DHMConv framework exhibits exceptional potential for effectively handling directed hypergraphs.

2 Preliminaries

For any arbitrary unweighted directed hypergraph G, it can be expressed as $G = (V, E)$ (Gallo et al., 1993), where V denotes the set of vertices in the directed hypergraph $V = \{v_1, v_2, v_3, \cdots, v_n\}$, and E represents the set of directed hyperedges in the directed hypergraph $E = \{e_1, e_2, e_3, \cdots, e_m\}$.

Given a directed hyperedge e in a hypergraph, its directional information can be represented as $e = \{e^{tail}, e^{head}\}$, where e^{tail} is the set of tail vertices of the hyperedge, and e^{head} is the set of head vertices of the hyperedge. Information propagation occurs within the same directed hyperedge, propagating from e^{tail} to e^{head}.

Similar to the definition of a directed hyperedge e, we define the adjacency matrix $H = \{H^{tail}, H^{head}\}$ for a directed hypergraph, it denotes the membership of a vertex in a hyperedge. The adjacency matrix H is defined based on the combination of H^{tail} and H^{head}, as shown in formulas (1) and (2), respectively.

$$H^{tail}(v,e) = \begin{cases} 1, & v \in e^{tail} \\ 0, & v \notin e^{tail} \end{cases} \quad (1)$$

$$H^{head}(v,e) = \begin{cases} 1, & v \in e^{head} \\ 0, & v \notin e^{head} \end{cases} \quad (2)$$

Directed hyperedges and vertex degrees are crucial factors in spatial convolutions. By utilizing the adjacency matrix, we can define and compute the degrees of directed hyperedges and vertices, as outlined in formulas (3) and (4), respectively.

$$dig(e) = \sum_{v \in V} H^{tail}(v,e) + \sum_{v \in V} H^{head}(v,e) \quad (3)$$

$$dig(v) = \sum_{e \in E} H^{tail}(v,e) + \sum_{e \in E} H^{head}(v,e) \quad (4)$$

3 Directed Hypergraph Momentum Convolution

In this section, we provide a detailed exposition of the information propagation process of the DHMConv model proposed in this paper from a spatial perspective, which is analogous to the message-passing process of vertex-hyperedge-vertex in hypergraph convolution. It is worth noting that the directedness of hypergraphs in the DHMConv model guides the entire information propagation process, distinguishing it from the message-passing process in conventional hypergraphs (Gao et al., 2022; Bai et al., 2021).

3.1 Directed Hyperedge Information Aggregation

In prior hypergraph convolution works (Ausiello and Laura, 2017), the process of aggregating vertex information through hyperedges did not explicitly consider the individual weights associated with each vertex/hyperedge pair. Instead, weight matrices were utilized to determine the significance of vertices, which overlooked the heterogeneity of vertex impacts on different hyperedges. Inspired by the work of Veličković et al. (2017), this paper proposes an attention-based hyperedge information aggregation process to account for this variation.

In this paper, attention coefficients for the vertex-hyperedge pairs are computed using the following method:

$$\varepsilon_{ve} = LeakyReLU(a^{\mathrm{T}}[Wh_v^l || Wh_e^l]), \quad v \in e \tag{5}$$

$$\lambda_{ve} = \frac{exp(\varepsilon_{ve})}{\sum_{i \in e^{head}} exp(\varepsilon_{ie^{head}}) + \sum_{j \in e^{tail}} exp(\varepsilon_{je^{tail}})} \tag{6}$$

Where a and W are trainable parameters, h_v and h_e represent feature vectors of vertices and hyperedges. $||$ is the concatenation operation. Specifically, ε_{ve} and λ_{ve} denote attention value and attention score between vertices and hyperedges, respectively.

The information aggregation process of hyperedges is modeled by the equation (7), which comprises two distinct cases: directional aggregation and bidirectional aggregation.

$$h_e^{l+1} = Aggregate(h_v^l), \quad v \in \begin{cases} e^{tail}, & Directional \\ e^{head} \cup e^{tail}, & Bidirectional \end{cases} \tag{7}$$

Building on the aforementioned discussion, the information flow from the head and tail vertices of a directed hyperedge to the hyperedge itself can be mathematically formulated as follows:

$$\hat{h}_{e^{head}} = \sum_{v \in e^{head}} \frac{\lambda_{ve}}{\sqrt{dig(v)} \cdot \sqrt{dig(e)}} h_v^l \tag{8}$$

$$\hat{h}_{e^{tail}} = \sum_{v \in e^{tail}} \frac{\lambda_{ve}}{\sqrt{dig(v)} \cdot \sqrt{dig(e)}} h_v^l \tag{9}$$

Where $dig(v)$ and $dig(e)$ denote the degree of a vertex and a hyperedge, respectively, and the function of $\sqrt{dig(v)} \cdot \sqrt{dig(e)}$ is to normalize the data.

Representation of hyperedge embedding in layer $l+1$ is given by:

$$h_e^{l+1} = f(\hat{h}_{e^{head}}, \hat{h}_{e^{tail}}) + h_e^l \tag{10}$$

The function $f(\cdot)$ in formula (10) is influenced by the direction of the aggregation. If it is directional aggregation, the value is calculated as $\hat{h}_{e^{tail}}$. If it is bidirectional aggregation, the value is calculated as the sum of $\hat{h}_{e^{head}}$ and $\hat{h}_{e^{tail}}$. The updated values from the vertices and the hyperedge information from layer l are summed to represent the hyperedge embedding for layer $l+1$.

3.2 Directed Hyperedge Information Broadcasting

As previously discussed, the conventional approach to information propagation in directed graphs typically relies on the directionality of edges, which may lead to the loss of crucial information from the head vertex when propagating to the tail vertex during the convolution process. To address this issue, we propose a novel design where the direction of directed edges is leveraged as the weight for information propagation. This design allows for the simultaneous transmission of information along the directed edges while retaining a proportion of it to be conveyed to the tail vertex.

Similar to the process of information aggregation, directed hyperedge information broadcasting can also be divided into directional and bidirectional cases. It can be described as the formula (11):

$$h_v^{l+1} = Broadcast(h_e^{l+1}), \quad v \in \begin{cases} e^{head}, & Directional \\ e^{head} \cup e^{tail}, & Bidirectional \end{cases} \tag{11}$$

More specifically, the amount of information that a vertex receives from a hyperedge can be quantified using the following formula (12) and (13):

$$\hat{h}_{v \in e^{head}} = \sum_{v \in e^{head}} h_e^{l+1} \tag{12}$$

$$\hat{h}_{v \in e^{tail}} = \sum_{v \in e^{tail}} h_e^{l+1} \tag{13}$$

The vertex embedding at layer $l+1$ can be represented as the formule (14):

$$h_v^{l+1} = g(\hat{h}_{v \in e^{head}}, \hat{h}_{v \in e^{tail}}) + h_v^l \tag{14}$$

The function $g(\cdot)$ in formula (14) is also influenced by the directionality of the information broadcasting. When the transmission is directional, it is calculated as $\hat{h}_{v \in e^{head}}$. When the transmission is bidirectional, a directed hypergraph momentum encoder is used to encode it. The resulting output is then added to the information from layer l to obtain the vertex information at layer $l+1$.

3.3 Model construction and momentum encoder

The matrix representation of information update during the hyperedge information aggregation process can be expressed as the formula (15) and (16):

$$X_e^{l+1}(D) = \hat{X}_{e^{tail}} + X_e^l \tag{15}$$

$$X_e^{l+1}(B) = \hat{X}_{e^{tail}} + \hat{X}_{e^{head}} + X_e^l \tag{16}$$

Where X represents a matrix composed of vectors, and D and B respectively represent Directional and Bidirectional. The above equation indicates adding the corresponding elements of the matrices.

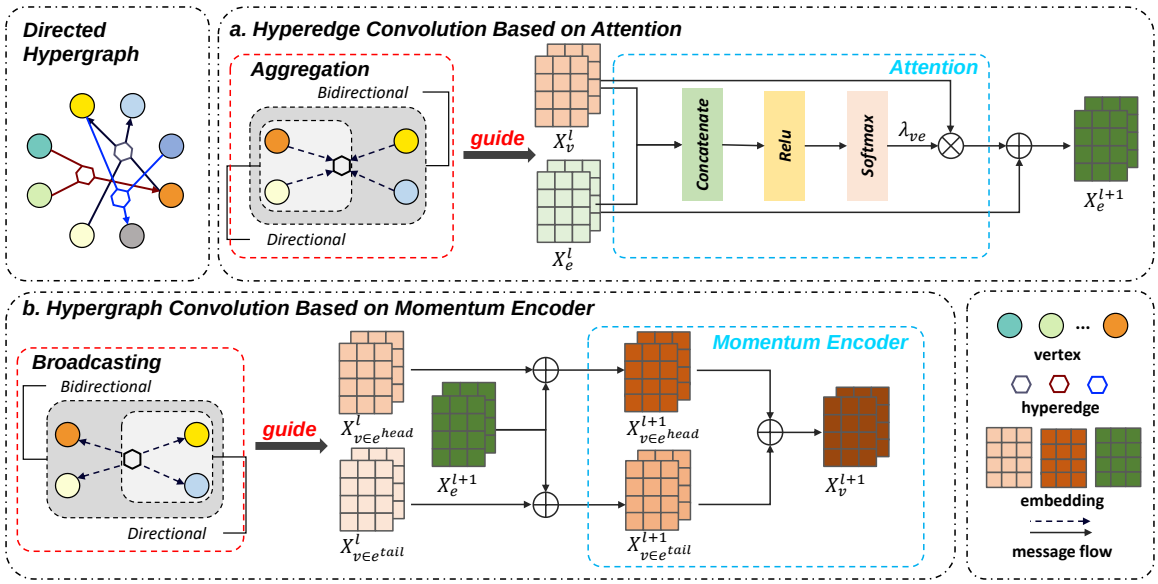


Figure 1: The basic structure of the proposed DHMConv framework in this paper. **(a)** Attention-based directed hyperedge convolution. The attention mechanism allows the model to focus on specific hyperedges and vertices during the convolution operation, resulting in more effective and accurate updates to the hyperedge information. **(b)** Hypergraph convolution based on directed hypergraph momentum encoder. The momentum encoder enhances the accuracy of directional information capture, thereby improving the precision of updates to vertex information.

Similarly, the matrix form of information update in the hyperedge information broadcasting process can be expressed as the formula (17) and (18):

$$X_v^{l+1}(D) = H^{head} \cdot X_e^{l+1} + X_v^l \tag{17}$$

$$X_v^{l+1}(B) = \alpha \cdot (H^{head} \cdot X_e^{l+1} + X_v^l) + \beta \cdot (H^{tail} \cdot X_e^{l+1} + X_v^l) \tag{18}$$

In order to reflect the effect of directionality in the broadcast process, we use weights α and β for different propagation directions to control the direction of information transmission. α and β are trainable parameters, by adjusting their values, control over the direction of information propagation in the broadcast process can be achieved.

We can add self-loops to vertices to obtain the $l+1$ layer vertex information by the same formula as above. By normalizing the weights with $\eta = \alpha/(\alpha+\beta)$, the update during hyperedge broadcasting can be represented as:

$$X_v^{l+1}(B) = \eta \cdot H^{head} \cdot X_e^{l+1} + (1-\eta) \cdot H^{tail} \cdot X_e^{l+1} \tag{19}$$

In this paper, the equation (19) is referred to as the directed hypergraph momentum encoder. The complete process of hyperedge information aggregation and hyperedge information broadcasting based on momentum encoder is referred to as directed hypergraph momentum convolution.

In the process of directed hypergraph momentum convolution, we drew inspiration from k-GNN (Morris et al., 2019) and employed the summation function as the aggregation scheme. The scientific validity of this approach was confirmed by Xu et al. (2018).

The proposed framework is capable of processing the convergence and broadcasting processes in both unidirectional and bidirectional ways, depending on the directionality of the directed hypergraph, as illustrated in figure 1. As a result, four types of directed hypergraph convolutional models are obtained, namely DHMConv-DD, DHMConv-DB, DHMConv-BD, and DHMConv-BB, each of which captures different aspects of the information flow within the directed hypergraph.

4 Experiment

4.1 Evaluation Metrics and Datasets

In link prediction tasks with explicit features, we use vertex features provided in the dataset as initial features. For link prediction tasks with implicit features, we generate initial features for potential feature learning using a uniform distribution.* The evaluation metrics for link prediction tasks are the area under the ROC curve (AUC) and the average precision (AP).

*https://github.com/WBZhao98/DHMConv

Vertex classification tasks are a fundamental benchmark for evaluating the ability of machine learning models to learn effective feature representations. In this study, we conduct node classification experiments on the widely used Citeseer and Cora datasets. Following the experimental protocol of prior research Kipf and Welling (2016), we limit the training dataset to 20 labels per class and reserve 500 samples for testing purposes. The evaluation of our proposed model's performance is conducted using accuracy (ACC) as the evaluation metric.

We conduct experiments on several open-access directed graph datasets, Air (Kunegis, 2013), Citeseer (Rossi and Ahmed, 2015), Cora (Sen et al., 2008), DBLP (Ley, 2002), and Survey (Moody, 2001). As shown in algorithm 1, we construct directed hyperedges based on originally directed edges and related semantic information.

Algorithm 1 Directed Hypergraph Construction

Input: directed graph adjacency matrix A
Initialization: $H^{tail} = 0 \in \mathbb{R}^{n\times n}, H^{head} = 0 \in \mathbb{R}^{n\times n}$, number of directed hyperedges $m = 1$
for $i, j = 1, 2, \ldots, n$ **do**
  Iterate over all directed edges
  if $A[j][i] = 1$ **then**
    $H^{tail}[i][j] = 1$;
    $H^{head}[j][i] = 1$;
  end if
  Construct adjacency matrix from the incoming edges
end for
Delete all-zero columns of the matrix H^{tail}, H^{head}
Output: directed hypergraph adjacency matrix $H = \{H^{tail} \in \mathbb{R}^{n\times m}, H^{head} \in \mathbb{R}^{n\times m}\}$

Algorithm 1 constructs a directed hypergraph based on the incoming edges of a directed graph, which is designed to ensure that all vertices that simultaneously point to a vertex are included within the same directed hyperedge. The advantage of this approach is to maximize the capture of vertices with higher-order relationships into a single hyperedge.

Table 1 shows the data information. All the datasets listed in Table 1 will be utilized for the link prediction with implicit features task. Citeseer and Cora, which contain vertex features and category information, will additionally be employed for both the link prediction with explicit features and vertex classification tasks.

Table 1: Statistical information of datasets.

Datasets	Nodes	Edges	Vertex	Edge
Air	1,226	2,615	Airport	Route
Citeseer	3,312	4,715	Paper	Citation
Cora	2,708	5,429	Paper	Citation
DBLP	12,590	49,759	Paper	Citation
Survey	2,539	12,969	User	Friendship

4.2 Baseline

We have selected the most commonly used graph models as our baselines, including GCN (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2017). Moreover, we have also incorporated advanced hypergraph learning models, such as HGNN (Feng et al., 2019), HNHN (Dong et al., 2020), HGNN+ (Gao et al., 2022), and directed graph learning models, such as DiGCN (Tong et al., 2020) and DiGAE (Kollias et al., 2022).

The baseline models employed in this study provide comprehensive evidence of the divergent performance exhibited by graph convolutional networks operating in the spatial and spectral domains, as well as their augmented counterparts that integrate higher-order or directional information. These models are systematically evaluated on common datasets, highlighting their distinct performances.

4.3 Parameter Settings and Environment

To train all models, we optimized them with the Adam algorithm and employed full-batch gradient descent. In order to ensure fairness in our comparative experiments, we conducted hyperparameter tuning for all models to obtain the best baseline results. We employed grid search to tune hyperparameters, where the learning rate lr was selected from {0.001, 0.005, 0.01, 0.05, 0.1}, weight decay w was selected from {0.0001, 0.0002, 0.0003}, and dropout rate (Srivastava et al., 2014) dp was selected from {0.1, 0.2, 0.3}. All other parameters, as well as some model-specific parameters, were set to the best values given in the original paper's corresponding experiments (Tong et al., 2020; Kollias et al., 2022). In the experiments, DHMConv utilized a single-layer network structure to obtain vertex embeddings.

All experiments were conducted on a 64-bit Windows 10 operating system with an Intel Core i7-10700 CPU @ 2.9GHz and 32GB of memory. The algorithm implementation utilized Python 3.6 as the underlying programming language environment, with NumPy 1.19.5 used for data format conversion and matrix operations. The development of the relevant deep learning model framework was based on pytorch version 1.10.0.

Table 2: The results of the Link Prediction task using 64-dimensional implicit features.

Model	Air		Citeseer		Cora		DBLP		Survey	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
GCN	71.56	73.05	67.93	70.35	74.63	76.30	89.28	87.77	86.16	87.25
GraphSAGE	81.97	79.23	75.68	76.59	85.70	83.94	91.32	89.08	87.34	85.36
GAT	80.33	81.52	73.97	77.56	82.59	84.62	95.34	95.82	91.09	91.62
DiGCN	87.27	88.18	77.09	80.24	87.66	87.58	86.92	84.02	90.21	89.91
DiGAE	80.15	79.93	72.55	75.15	82.68	83.29	88.96	87.01	86.14	83.99
HGNN	85.17	84.52	77.45	79.11	87.31	86.27	88.64	86.98	84.60	87.58
HNHN	83.62	80.51	77.81	78.53	85.79	84.72	71.72	71.10	80.53	87.50
HGNN+	87.14	85.60	79.52	81.93	90.11	90.72	89.83	88.59	85.82	88.96
DHMConv-DD	87.46	86.52	80.78	80.38	89.54	88.51	88.40	87.49	90.35	89.45
DHMConv-DB	87.52	88.07	78.70	81.86	90.08	90.31	95.34	95.37	93.36	93.79
DHMConv-BD	88.88	88.80	81.69	83.35	91.43	90.73	91.17	90.37	91.83	91.37
DHMConv-BB	87.07	87.23	78.86	82.05	89.92	91.54	95.22	95.38	93.47	93.72

4.4 Performance Comparison

In this section, we present the results of experiments on both vertex classification and link prediction tasks, where we examine the impact of using different types of vertex features and the effects of removing different modules in our model. In all tables, the top score is presented in red, the second best score is presented in blue, and the third best score is presented in cyan. The data in the table represents the average of the results obtained from 100 repetitions of the experiment.

4.4.1 Link prediction with implicit features

In the majority of scenarios, obtaining vertex features is challenging, which is why most directed graph datasets do not include initial features for vertices. In this experiment, the vertex features of Citeseer and Cora datasets will not be used, all datasets will be randomly assigned 64-dimensional initial features for the vertices.

Table 2 lists the results of our link prediction experiments based on latent features. Our model outperforms the baseline model on almost all AUC and AP metrics for all five datasets. This demonstrates the impressive structural feature representation capability of our model.

Graph models are designed for undirected graphs and may be less effective on directed datasets. Although GAT performs well on the DBLP dataset, its performance is mediocre on other datasets. The hypergraph models are designed to extract high-order features of hypergraphs, which have been demonstrated to be advantageous over graphs on several datasets. However, due to the lack of directional information, its performance is poor on the DBLP and Survey datasets. In contrast, directed graph models preserve directional information, but their ability to capture high-order information is limited. Overall, our experimental results support capturing high-order and directional information for accurate link prediction.

DHMConv is a novel methodology that retains high-order information of hypergraphs while introducing a directed hypergraph momentum encoder to encode directional information. This approach amalgamates the advantages of hypergraphs and directed graphs and has demonstrated remarkable potential on directed graph datasets. It is worth noting that the selection of direction is crucial for DHMConv, and in most scenarios, we recommend integrating the results from multiple directions to obtain optimal performance.

4.4.2 Vertex classification with explicit features

Vertex classification is one of the most common tasks on graph. After applying directed hypergraph convolution, we obtain embeddings for the vertices and reduce their dimensionality to match the number of categories. We then use the Softmax function to obtain the probabilities of vertices belonging to each category.

The results of vertex classification using explicit features are presented in Table 3. It is evident that the utilization of directed hypergraph convolution can enhance the classification accuracy on the Citeseer and Cora datasets. Among the observed outcomes, DHMConv achieved the highest performance.

In order to visualize the vertex embeddings generated by DHMConv during the process of vertex classification, we employed t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton, 2008) to reduce the dimensionality of the embed-

Table 3: The accuracy of vertex classification tasks with explicit features on the Citeseer and Cora datasets.

Model	Citeseer	Cora
GCN	70.37	79.36
GraphSAGE	68.28	78.85
GAT	70.79	80.13
DiGCN	70.81	80.60
HGNN	71.22	81.21
HNHN	68.87	79.41
HGNN+	70.30	80.64
DHMConv	73.31	82.20

dings. Our results, as shown in Figure 2, demonstrate that the vertex information after the model output can clearly distinguish data points with different class labels, a separation that is not evident in their initial feature space.

DHMConv exhibits superior performance on the Citeseer dataset. However, its comparative performance on the Cora dataset does not demonstrate a significant advantage. In order to investigate the classification results, we conducted an analysis of the confusion matrix and performed a visualization of the multi-class classification outcomes. The heat map in Figure 3 displays the multi-class classification performance of DHMConv on both datasets.

Based on the heatmap analysis, it is evident that the classification performance of the Citeseer dataset is comparatively superior in the Agents and IR categories, whereas it exhibits only average performance on the remaining datasets. This could potentially be attributed to the existence of inter-referencing among papers from different categories. Notably, the classification results across all categories on the Cora dataset are quite satisfactory.

4.4.3 Link prediction with explicit features and ablation experiment

We conduct link prediction using the explicit vertex features present in the Citeseer and Cora datasets. Additionally, we perform ablation experiments to investigate the significance of individual modules within the model.

Experimental results of link prediction with explicit features, as shown in Table 4, demonstrate the superiority of DHMConv over the baseline on two citation networks with explicit features. DHMConv consistently achieves the best results across all metrics, indicating its effectiveness in preserving graph structure and feature information. Importantly, the second-

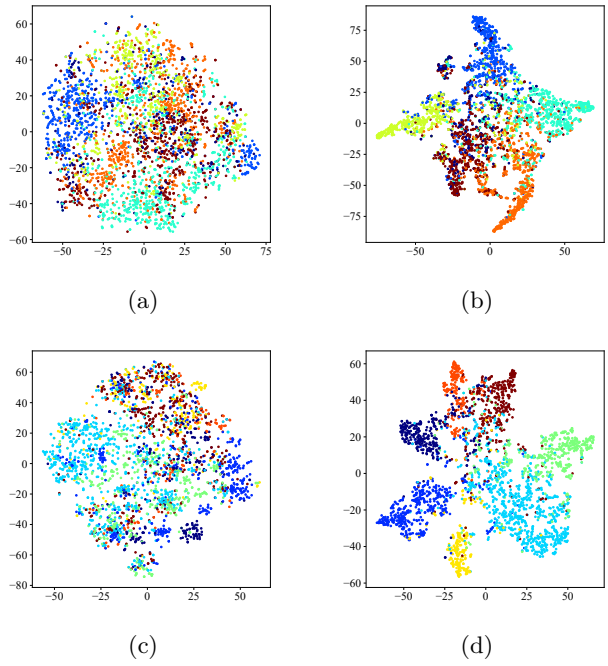
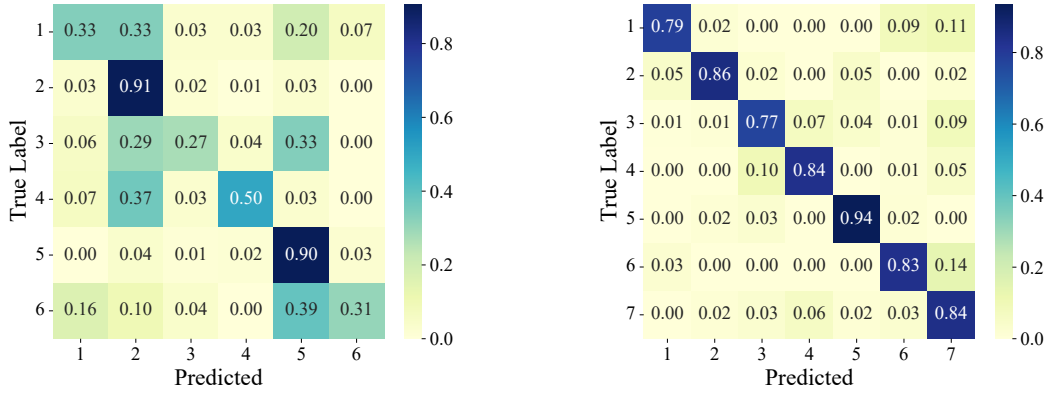


Figure 2: **(a)** The distribution of raw data in Citeseer dataset; **(b)** The distribution of data in Citeseer dataset output by DHMConv model; **(c)** The distribution of raw data in Cora dataset; **(d)** The distribution of data in Cora dataset output by DHMConv model.

best results are mainly observed in the experiments conducted on directed graph models, highlighting the significance of preserving asymmetric information for directed graphs with explicit features.

The *DHMConv* in Table 4 represents the best performance achieved by DHMConv on four different direction combinations, the AUC and AP metrics are obtained from the results of a single model tested under identical conditions. The specific performances of DHMConv-DD, DHMConv-DB, DHMConv-BB, and DHMConv-BD are illustrated in Figure 4. It indicates that in this task, the performance of DHMConv-DD is slightly weaker than the other three models, but still better than most baselines.

The results of the ablation experiments in Table 5 demonstrate how we achieved optimal performance compared to the baseline experiment. It is evident that both the attention mechanism and momentum mechanism significantly improve the two metrics on both datasets compared to the directed hypergraph spatial convolution. This highlights the usefulness and necessity of the attention mechanism and momentum encoder.



(a) (b)

Figure 3: **(a)** Visualization results of multi-class classification on Citeseer dataset, which is divided into 6 categories; **(b)** Visualization results of multi-class classification on Cora dataset, which is divided into 7 categories.

Table 4: The results of link prediction tasks with explicit features on the Citeseer and Cora datasets.

Model	Citeseer		Cora	
	AUC	AP	AUC	AP
GCN	88.37	89.29	91.99	90.27
GraphSAGE	87.35	88.85	93.11	92.95
GAT	82.27	84.46	91.33	91.18
DiGCN	93.16	93.96	93.28	93.24
HGNN	92.98	92.49	82.94	82.36
HNHN	88.40	89.51	92.27	92.38
HGNN+	93.06	93.43	93.60	93.12
DHMConv	95.34	95.34	93.91	93.76

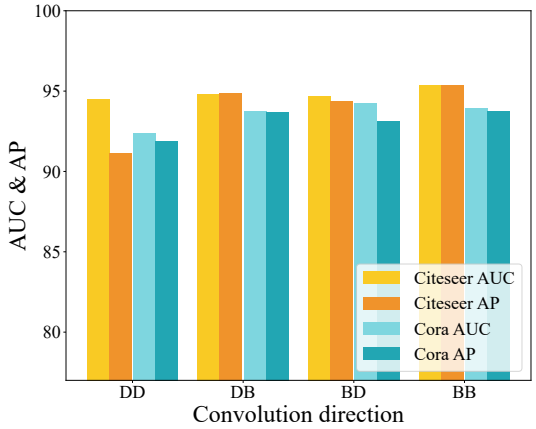


Figure 4: The impact of different convolution directions on the results of Link Prediction task

Table 5: The results of ablation experiments on directed hypergraph with Attention and Momentum.

Model	Citeseer		Cora	
	AUC	AP	AUC	AP
Spatial convolution	91.79	91.22	90.98	90.27
+ Attention	94.53	91.15	92.37	91.90
+ Momentum	95.34	93.34	94.23	93.10

5 Relation Work

We categorize the existing research related to our work into two categories: 1) Enhancing graph structures with higher-order information; 2) Enhancing graph structures with directional information.

5.1 Enhancing graph structures with higher-order information

Early analysis of spectral-based convolution on graphs was conducted by Hammond et al. (2011), and Bruna et al. (2013) proposed a spectral-based graph convolution model based on their work, which successfully extracted structural information from graphs. Defferrard et al. (2016) refined the spectral-based convolution model by utilizing Chebyshev polynomials. Kipf and Welling (2016) further optimized the graph convolution process and proposed GCN, which employs a first-order Chebyshev approximation. GCN has lower complexity and is more easily trained, faster, and more effective, rendering it one of the most classic and practical methods in graph machine learning. Message-passing neural network (MPNN) is considered a classi-

cal graph convolutional algorithm based on the spatial domain (Gilmer et al., 2017). On this basis, Hamilton et al. (2017) proposed the GraphSAGE model, which endows it with inductive learning ability. Veličković et al. (2017) proposed the graph attention network (GAT), which can assign different weights to arbitrary pairs of vertices during the graph convolution process, thus better incorporating the correlation between vertex features into the model.

To capture higher-order neighbor relationships in graphs, building upon the inspiration from the K-WL algorithm, Morris et al. (2019) introduced the K-GNN model, which leverages Graph Neural Networks (GNNs) to explicitly capture higher-order relationships. As hypergraphs can preserve higher-order information compared to traditional graph structures, Huang and Yang (2021) proposed a unified framework UniGNN, for explaining the message-passing process in both graph and hyper-layered neural networks, which can generalize the traditional GNN models to hypergraph neural networks.

Zhou et al. (2006) proposed a hypergraph inference learning model. Feng et al. (2019) proposed HGNN, which introduced spectral domain convolution on hypergraphs. Dong et al. (2020) proposed HNHN, a framework for hypergraph representation learning. Bai et al. (2021) proposed a hypergraph attention model, further enhancing the representation learning capabilities. Gao et al. (2022) proposed HGNN+, a framework for modeling the correlation of high-order multimodal, multitype data, which learns optimal representations within the hypergraph framework.

5.2 Enhancing graph structures with directional information

Ghorbanzadeh et al. (2021) modeled social networks as graph structures and performed link prediction. This methodology inherently omits unidirectional information present in social networks. Zhang et al. (2021) proposed a directed graph neural network based on the magnetic Laplacian operator, which can apply classic algorithms on directed graphs. Niepert et al. (2016) proposed a convolutional framework that can be used for both undirected and directed graphs. Ma et al. (2019) proposed a model for directed graph spectral domain convolution, which can be directly applied to semi-supervised node classification tasks in directed graphs. Tong et al. (2020) proposed a directed graph convolutional model DiGCN, which utilizes convolution and k-order approximation to learn multiscale features in directed graphs. Kollias et al. (2022) proposed a directed graph autoencoder model DiGAE, which is used for learning pair-wise interpretable latent representations of nodes in directed graphs.

In the field of directed hypergraphs, Ausiello and Laura (2017) provided a comprehensive overview of fundamental algorithms for directed hypergraphs. Tran and Tran (2020) proposed a directed hypergraph neural network methodology, which was applied to node classification tasks. Xiao et al. (2022) introduced a directed hypergraph convolutional network based on hyperbolic space modeling, effectively addressing the issue of asymmetric correlation matrices in directed hypergraphs.

Certainly, the research on the convolutional directionality issue goes beyond the studies mentioned earlier. Beaini et al. (2021) proposed an intuitive idea of directional flow and their DGN method provided an interpretable solution to several issues of GNNs, including lack of anisotropy and low expressive power. Huang et al. (2021) defined four different signed directed relationships based on social theory and proposed corresponding GNN models to aggregate and propagate information of nodes in signed networks. Clearly, exploring convolutional directionality in this way has positive implications.

6 Conclusion and Future Work

In this work, we proposed a framework for spatial convolution on directed hypergraphs, which optimizes the aggregation and broadcasting of hyperedge information through attention mechanisms and directed hypergraph momentum encoders. We construct four distinct models based on directionality to address the needs of varied data and task scenarios.

We perform diverse comparative experiments on multiple benchmark datasets to assess the performance of our framework. Our findings reveal that our proposed framework possesses strong feature representation capabilities and achieves exceptional results on various fundamental tasks.

In the future, our plan is to employ the proposed framework in practical applications such as product recommendation and refine the model details for specific tasks to improve its real-world performance.

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) and Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

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

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

Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. (2021). Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR.

Bretto, A. (2013). Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chang, J., Gao, C., Zheng, Y., Hui, Y., Niu, Y., Song, Y., Jin, D., and Li, Y. (2021). Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 378–387.

Chen, Z., Chen, C., Zhang, Z., Zheng, Z., and Zou, Q. (2019). Variational graph embedding and clustering with laplacian eigenmaps. In *IJCAI*, pages 2144–2150.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

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

Fan, H., Zhang, F., Wei, Y., Li, Z., Zou, C., Gao, Y., and Dai, Q. (2021). Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4125–4138.

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*.

Ghorbanzadeh, H., Sheikhahmadi, A., Jalili, M., and Sulaimany, S. (2021). A hybrid method of link prediction in directed graphs. *Expert Systems with Applications*, 165:113896.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

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

Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Huang, J., Shen, H., Hou, L., and Cheng, X. (2021). Sdgnn: Learning node representation for signed directed networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 196–203.

Huang, J. and Yang, J. (2021). Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*.

Jiang, W. and Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, page 117921.

Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., and Xing, E. P. (2019). Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11487–11496.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7211–7219.

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

Ley, M. (2002). The dblp computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval: 9th International Symposium, SPIRE 2002 Lisbon, Portugal, September 11–13, 2002 Proceedings 9*, pages 1–10. Springer.

Liao, L., He, X., Zhang, H., and Chua, T.-S. (2018). Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270.

Ma, Y., Hao, J., Yang, Y., Li, H., Jin, J., and Chen, G. (2019). Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990*.

Mercado, R., Rastemo, T., Lindelöf, E., Klambauer, G., Engkvist, O., Chen, H., and Bjerrum, E. J. (2021). Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023.

Micheli, A. (2009). Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511.

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

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR.

Rossi, R. and Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

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.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

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*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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

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.

Xie, F., Chen, L., Ye, Y., Liu, Y., Zheng, Z., and Lin, X. (2018). A weighted meta-graph based approach for mobile application recommendation on heterogeneous information networks. In *Service-Oriented Computing: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings 16*, pages 404–420. Springer.

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

Zhang, X., He, Y., Brugnone, N., Perlmutter, M., and Hirn, M. (2021). Magnet: A neural network for directed graphs. *Advances in neural information processing systems*, 34:27003–27015.

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.

Checklist

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]
   Not Applicable, the main comparison in this study is made between the baseline and the metrics such as AUC, AP, and ACC.
   (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]
Yes

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