UniGNN: a Unified Framework for Graph and Hypergraph Neural Networks

Jing Huang, Jie Yang*

Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University
{ziruochenxia, jieyang}@sjtu.edu.cn

Abstract

Hypergraph, an expressive structure with flexibility to model the higher-order correlations among entities, has recently attracted increasing attention from various research domains. Despite the success of Graph Neural Networks (GNNs) for graph representation learning, how to adapt the powerful GNN-variants directly into hypergraphs remains a challenging problem. In this paper, we propose UniGNN, a unified framework for interpreting the message passing process in graph and hypergraph neural networks, which can generalize general GNN models into hypergraphs. In this framework, meticulously-designed architectures aiming to deepen GNNs can also be incorporated into hypergraphs with the least effort. Extensive experiments have been conducted to demonstrate the effectiveness of UniGNN on multiple real-world datasets, which outperform the state-of-the-art approaches with a large margin. Especially for the DBLP dataset, we increase the accuracy from 77.4% to 88.8% in the semi-supervised hypernode classification task. We further prove that the proposed message-passing based UniGNN models are at most as powerful as the 1-dimensional Generalized Weisfeiler-Leman (1-GWL) algorithm in terms of distinguishing non-isomorphic hypergraphs. Our code is available at https://github.com/OneForward/UniGNN.

1 Introduction

Hypergraphs are natural extensions of graphs by allowing an edge to join any number of vertices, which can represent the higher-order relationships involving multiple entities. Recently, hypergraphs have drawn the attention from a wide range of fields, like computer vision [Gao et al., 2020; Liu et al., 2020], recommendation system [Xia et al., 2021] and natural sciences [Gu et al., 2020], and been incorporated with various domain-specific tasks.

In a parallel note, Graph Representation Learning has raised a surge of interests from researchers. Numerous powerful Graph Neural Networks (GNNs) have been presented, achieving the state-of-the-art in specific graph-based tasks, such as node classification [Chen et al., 2020], link prediction [Zhang and Chen, 2018] and graph classification [Li et al., 2020a]. Most GNNs are message-passing based models, like GCN [Kipf and Welling, 2017], GAT [Veličković et al., 2017], GIN [Xu et al., 2018a] and GraphSAGE [Hamilton et al., 2017], which iteratively update node embeddings by aggregating neighboring nodes’ information. The expressive power of GNNs is well-known to be upper bounded by 1-Weisfeiler-Leman (1-WL) test [Xu et al., 2018a] and many provably more powerful GNNs mimicking higher-order-WL test have been presented [Morris et al., 2019a; Morris et al., 2019b; Li et al., 2020b].

Furthermore, several works, like JKNet [Xu et al., 2018b], DropEdge [Rong et al., 2019], DGN [Zhou et al., 2020] and GCNII [Chen et al., 2020], have devoted substantial efforts to tackling the problem of over-smoothing, an issue when node embeddings in GNNs tend to converge as layers are stacked up and the performance downgrade significantly.

Despite the success of GNNs, how to learn powerful representative embeddings for hypergraphs remains a challenging problem. HGGN [Feng et al., 2019] is the first hypergraph neural network, which uses the clique expansion technique to approximate hypergraphs as graphs, and simplifies the problem above to the graph embedding framework. This approach, as illustrated in Fig 1, cannot cover the substructures like hyperedges which recursively contain other hyper-edges are discarded with clique expansion. HyperGCN [Yadati et al., 2019] enhances the generalized hypergraph Laplacian with additional weighted pairwise edges (a.k.a mediators). This approach still fails to reserve complete hypergraph information since Graph Laplacian can only describe pairwise connections between vertices in one training epoch. Another work, HyperSAGE [Aarya et al., 2020] learns to embed hypergraphs directly by propagating messages in a two-stage procedure. Although HyperSAGE shows the capability to capture information from hypergraph structures with a giant leap in performance, it fails to adapt powerful classic GNN designs into hypergraphs.

In view of the fact that more and more meticulously-designed network architectures and learning strategies have

*Corresponding author: Jie Yang. This work is partly supported by NSFC, China (No: 61876107, U1803261, 61806125).
appeared in graph learning, we are naturally motivated to ask the following intriguing question:

**Can the network design and learning strategy for GNNs be applied to HyperGNNs directly?**

### Contributions

This paper proposes the UniGNN, a unified framework for graph and hypergraph neural networks, with contributions unfolded by the following questions:

**Q 1.** Can we generalize the well-designed GNN architecture for hypergraphs with the least effort?

**Q 2.** Can we utilize the learning strategies for circumventing the over-smoothing in the graph learning and design deep neural networks that adapt to hypergraphs?

**Q 3.** How powerful are hypergraph neural networks?

By addressing the above questions, we highlight our contributions as follows:

**A 1.** We present the UniGNN and use it to generalize several classic GNNs, like GCN, GAT, GIN and GraphSAGE directly into hypergraphs, termed UniGCN, UniGAT, UniGIN and UniSAGE, respectively. UniGNNs consistently outperform the state-of-art approaches in hypergraph learning tasks.

**A 2.** We propose the UniGCNII, the first deep hypergraph neural network and verify its effectiveness in resolving the over-smoothing issue.

**A 3.** We prove that message-passing based UniGNNs are at most as powerful as 1-dimensional Generalized Weisfeiler-Lehman (1-GWL) algorithm in terms of distinguishing non-isomorphic hypergraphs.

### 2 Preliminaries

#### 2.1 Notations

Let \( G = (V, E) \) denote a directed or undirected graph consisting of a vertex set \( V = \{1, \ldots, n\} \) and an edge set \( E \) (pairs of vertices). A self-looped graph \( G = (\tilde{V}, \tilde{E}) \) is constructed from \( G \) by adding a self-loop to each of its non-self-looped nodes. The neighbor-nodes of vertex \( i \) is denoted by \( N_i = \{(j) | (i, j) \in E\} \). We also denote vertex \( i \)'s neighbor-nodes with itself as \( \tilde{N}_i = N_i \cup \{i\} \). We use \( x_i \in \mathbb{R}^d \) to represent a \( d \)-dimensional feature of vertex \( i \).

A hypergraph \( H = (\tilde{V}, \tilde{E}) \) is defined as a generalized graph by allowing an edge to connect any number of vertices, where \( \tilde{V} \) is a set of vertices and a hyperedge \( e \in \tilde{E} \) is a non-empty subset of \( V \). The incident-edges of vertex \( i \) is denoted by \( E_i = \{ e \in \tilde{E} | i \in e \} \). We say two hypergraphs \( H_1 = (V_1, E_1) \) and \( H_2 = (V_2, E_2) \) are isomorphic, written \( H_1 = H_2 \), if there exists a bijection \( f : V_1 \rightarrow V_2 \) such that \( \forall e \subseteq V_1 : e \in E_1 \Leftrightarrow \{ f(v_1), \ldots, f(v_k) \} \in E_2 \).

#### 2.2 Graph Neural Networks

**General GNNs.** Graph Neural Networks (GNNs) learn the informative embedding of a graph by utilizing the feature matrix and the graph structure. A broad range of GNNs can be built up by the message passing layers, in which node embeddings are updated by aggregating the information of its neighbor embeddings. The message passing process in the \( l \)-th layer of a GNN is formulated as

\[
(GNN) \quad x_i^{l+1} = \phi^l (x_i^l, \{x_j^l\}_{j \in N_i}).
\]

Classic GNN models sharing this paradigm include GCN [Kipf and Welling, 2017], GAT [Veličković et al., 2017], GIN [Xu et al., 2018a], GraphSAGE [Hamilton et al., 2017], etc.

In the following sections, we omit the superscript \( l \) for the sake of simplicity and use \( \tilde{x}_i \in \mathbb{R}^d \) to indicate the output of the message passing layer before activation or normalization.

### 2.3 HyperGraph Neural Networks

**Spectral-based HyperGNNs.** HGGN [Feng et al., 2019] and HyperConv [Bai et al., 2019] utilize the normalized hypergraph Laplacian, which essentially converts the hypergraphs to conventional graphs by viewing each hyperedge as a complete graph. HyperGCN [Yadati et al., 2019] uses the generalized hypergraph Laplacian (changing between epochs) and injects information of mediators to represent hyperedges. Both methods depend on the hypergraph Laplacian, which however, emphasizes the pairwise relations between vertices. Another work, MPNN-R [Yadati, 2020] regards hyperedges as new vertices with \( \tilde{V} = V \cup \tilde{E} \) and represents the hypergraph by a \([\tilde{V}] \times [\tilde{E}]\) matrix. MPNN-R can effectively capture the recursive property of hyperedges, but fails to describe other high-order relationships, like complex and diverse intersections between hyperedges.

**Spatial-based HyperGNNs.** A recent work, HyperSAGE [Arya et al., 2020] pioneers to exploit the structure of hypergraphs by aggregating messages in a two-stage procedure, avoiding the information loss due to the reduction of hypergraphs to graphs. With \( N(i, e) \) to denote vertex \( i \)'s intra-edge neighborhood for hyperedge \( e \), HyperSAGE aggregates information with the following rules:

\[
\begin{align*}
\{ h_{i,e} = M_1 (\{x_j\}_{j \in N(i,e;\alpha)}) \\
\tilde{x}_i = W (x_i + M_2 (h_{i,e}))
\end{align*}
\]

where \( N(i, e; \alpha) \) is a sampled subset of \( \alpha \) vertices from \( N(i, e) \). \( W \) is the linear transform and \( M_1 \) and \( M_2 \) are power mean functions.

HyperSAGE is the current state-of-the-art algorithm for hypergraph representation learning. However, there are some issues associated with HyperSAGE. Firstly, since the calculation for \( h_{i,e} \) is distinct for different \( (i, e) \) pairs, the original algorithm uses nested loops over hyperedges and vertices within hyperedges, which results in redundant computation and poor parallelism. Secondly, applying the power mean functions in both stages, neither of which is injective, fails to distinguish structures with the same distribution but different multiplicities of elements [Xu et al., 2018a]. Lastly, the original work still fails to address the over-smoothing issue associated with deep hypergraph neural networks.

### 3 UniGNN: a Unified Framework

To resolve the issues associated with HyperSAGE, we propose the UniGNN, a unified framework to characterize the
The key insight is that if we rethink Eq. (1) in GNNs as a two-stage message-passing process in GNNs and HyperGNNs:

\[
(\text{UniGNN}) \begin{cases} h_e = \phi_1 \left( \{x_j \}_{j \in e} \right) \\ \tilde{x}_i = \phi_2 \left( x_i, \{h_e \}_{e \in E_i} \right) \end{cases},
\]

where \( \phi_1 \) and \( \phi_2 \) are permutation-invariant functions for aggregating messages from vertices and hyperedges respectively. The update rule for UniGNN is illustrated in Fig 2. The key insight is that if we rethink Eq. (1) in GNNs as a two-stage aggregation process, then the designs for GNNs can be naturally generalized to hypergraphs in Eq. (3).

In the first stage, for each hyperedge \( e \), we use \( \phi_1 \) to aggregate features of all vertices within it. \( \phi_1 \) can be any permutation-invariant function satisfying \( \phi_1 \left( \{x_j \}_{j \in e} \right) = x_j \), such as the mean function \( h_e = \frac{1}{|e|} \sum_{j \in e} x_j \) or the sum function \( h_e = \sum_{j \in e} x_j \). It is obvious that if we let \( E_i = \{ e = \{j \} \mid \forall j \in N_i \} \), then \( h_e = x_j \) holds for any \( j \in N_i \). Therefore, UniGNN (3) can be reduced to GNN (1), which unifies both formulations into the same framework.

In the second stage, we update each vertex with its incident hyperedges using aggregating function \( \phi_2 \), of which the design can be inspired from existent GNNs directly. We will exhibit several effective examples in the following section.

### 3.1 Generalize Powerful GNNs for Hypergraphs

**UniGAT.** Graph Attention Networks (GAT) [Veličković et al., 2017] adopt the attention mechanism to assign importance score to each of center node’s neighbors, leading to more effective aggregation. The attention mechanism is formulated as

\[
\alpha_{ij} = \sigma(a^T [W x_i; W x_j]),
\]

\[
\tilde{\alpha}_{ij} = \frac{\exp (\alpha_{ij})}{\sum_{k \in \mathcal{N}_i} \exp (\alpha_{ik})},
\]

\[
\tilde{x}_i = \sum_{j \in \mathcal{N}_i} \tilde{\alpha}_{ij} W x_j,
\]

where \( \sigma \) is the leaky ReLU function, \( a \in \mathbb{R}^{2d} \) is the learnable attentional parameter and \( [\cdot; \cdot] \) means concatenation.

By rewriting the above equations, we can get UniGAT for hypergraphs as follows,

\[
\alpha_{ie} = \sigma(a^T [W h_i; Wh_e]),
\]

\[
\tilde{\alpha}_{ie} = \frac{\exp (\alpha_{ie})}{\sum_{e' \in \mathcal{E}_i} \exp (\alpha_{ie'})},
\]

\[
\tilde{x}_i = \sum_{e \in \mathcal{E}_i} \tilde{\alpha}_{ie} Wh_e,
\]

In this way, UniGAT learns to reweight the center node’s neighboring hyperedges.

UniGAT is essentially different from HyperGAT [Bai et al., 2019] since HyperGAT requires the hyperedges to be preprocessed into the same homogeneous domain of vertices before training, which is inflexible and unreliable in practice.

Note that based on the formulation for UniGCN and UniGAT, hypergraphs should be preprocessed with self-loops; that is, \( \forall i \in V \Rightarrow e = \{i\} \in \mathcal{E} \).

**UniGIN.** Graph Isomorphism Networks (GIN) [Xu et al., 2018a] is a simple yet effective model with the expressive power achieving the upper bound of message passing based GNNs. GIN updates node embeddings as

\[
\tilde{x}_i = W \left( (1 + \varepsilon) x_i + \sum_{j \in \mathcal{N}_i} x_j \right),
\]

where \( \varepsilon \) is a learnable parameter or some fixed scalar.

Similar to the previous deduction, UniGIN is formulated as

\[
\tilde{x}_i = W \left( (1 + \varepsilon) x_i + \sum_{e \in \mathcal{E}_i} h_e \right).
\]
UniSAGE. GraphSAGE [Hamilton et al., 2017] uses a general aggregating function, like mean aggregator, LSTM aggregator or max-pooling aggregator, which can be designed according to various tasks. We use a variant of GraphSAGE where the combining process is sum instead of concatenation following [Arya et al., 2020]:

\[ \hat{x}_i = W (x_i + \text{AGGREGATE} (\{x_j\}_{j \in N_i})) . \] (14)

UniSAGE is naturally generalized as

\[ \hat{x}_i = W (x_i + \text{AGGREGATE} (\{h_e\}_{e \in E_i})) . \] (15)

3.2 Towards Deep Hypergraph Neural Networks

Current hypergraph representation learning methods, like HGGN, HyperGCN and HyperSAGE, use a shallow network with two layers and the performance reduces significantly when layers are stacked up, which is in concordance with GNNs. This phenomenon is called over-smoothing. Although many works have focused on tackling this problem for graphs, like JKNet [Xu et al., 2018b], DropEdge [Rong et al., 2019] and GCNII [Chen et al., 2020], how to make hypergraphs deeper is still uncovered.

Since in our framework, learning strategies from graph learning domain can be incorporated into hypergraphs with the least effort, we solve this problem by presenting UniGCNII, a deep hypergraph neural network inspired from GCNII.

UniGCNN. GCNII [Chen et al., 2020] is a powerful deep graph convolutional network enhanced with Initial Residual Connection and Identity Mapping to vanquish the over-smoothing problem. We generalize GCNII to hypergraphs, dubbed UniGCNN, with the aggregation process defined as

\[ \begin{align*}
\hat{x}_i &= \frac{1}{\sqrt{d_i}} \sum_{e \in E_i} \frac{1}{\sqrt{d_e}} h_e \\
\hat{x}_i &= ((1 - \beta)I + \beta W) (1 - \alpha) \hat{x}_i + \alpha x^{0}_i
\end{align*} \] (16)

where \( \alpha \) and \( \beta \) are hyperparameters, \( I \) is identity matrix and \( x^{0}_i \) is the initial feature of vertex \( i \).

In each layer, UniGCNII employs the same two-stage aggregation as UniGCN to exploit the hypergraph structure, and then injects the jumping knowledge from the initial features and previous features. Experiments validate that UniGCNII enjoys the advantage of circumventing the over-smoothing issue when models are getting deeper.

4 How Powerful are UniGNNEs?

Message-passing based GNNs are capable of distinguishing local-substructure (like k-height subtree rooted at a node) or global structure of graphs, with the expressible power upper bounded by 1-WL test. In view of this, we are motivated to investigate the expressible power of UniGNNS for hypergraphs. We start by presenting a variant of the 1-dimensional Generalized Weisfeiler-Leman Algorithm (1-GWL) for hypergraph isomorphism test following the work of [Böker, 2019].

4.1 Generalized Weisfeiler-Leman Algorithm

1-GWL sets up by labeling the vertices of a hypergraph \( H = (V, \mathcal{E}) \) with \( \ell^0_i = 0 \) for any \( i \in V \), and in the \( t \)-th iteration the labels are updated by

\[ \begin{align*}
\ell^t_e &= \{ \{ \ell^t_i \}_{i \in V} \} \quad \forall e \in \mathcal{E} \\
\ell^{t+1}_i &= \{ \{ \ell^t_i, \ell^t_e \}_{e \in E_i} \} \quad \forall i \in V
\end{align*} \] (17)

where \( \ell^t_e \) denotes the label of a hyperedge \( e \in \mathcal{E} \), and \( \{ \ldots \} \) denotes a multiset.

1-GWL distinguish \( H_1 \) and \( H_2 \) as non-isomorphic if there exists a \( t \geq 0 \) such that

\[ \{ \{ \ell^t_{H_1,i} \mid i \in V_1 \} \} \neq \{ \{ \ell^t_{H_2,i} \mid i \in V_2 \} \} , \] (18)

where the subscript \( H_1 \) and \( H_2 \) are added for discrimination.

Proposition 1 (1-GWL). If 1-GWL test decides \( H_1 \) and \( H_2 \) are non-isomorphic, then \( H_1 \neq H_2 \).

We leave all the proofs in the supplemental files.

4.2 Discriminative Power of UniGNNs

We assign the same features to all vertices of a hypergraph \( H \) so that the UniGNN only depends on the hypergraph structure to learn. Let \( \mathcal{A} : \mathcal{H} \rightarrow \mathbb{R}^g \) be a UniGNN abiding by the aggregation rule (3), the following proposition indicates that \( \mathcal{A} \)’s expressive power is upper bounded by 1-GWL test.

Proposition 2. Given two non-isomorphic hypergraphs \( H_1 \) and \( H_2 \), if \( \mathcal{A} \) can distinguish them by \( \mathcal{A}(H_1) \neq \mathcal{A}(H_2) \), then 1-GWL test also decides \( H_1 \neq H_2 \).

The following theorem characterizes the conditions for UniGNNS to reach the expressive power of 1-GWL test.

Theorem 1. Given two hypergraphs \( H_1 \) and \( H_2 \) such that 1-GWL test decides as non-isomorphic, a UniGNN \( \mathcal{A} \) is sufficient to distinguish them by \( \mathcal{A}(H_1) \neq \mathcal{A}(H_2) \) with the following conditions:

1. **Local Level.** Two-stage aggregating functions \( \phi_1 \) and \( \phi_2 \) are both injective.

2. **Global Level.** In addition to the local-level conditions, \( \mathcal{A} \)’s graph-level READOUT function is injective.

We are also interested in UniGNNS’ capability of distinguishing local substructures of hypergraphs. We define the local substructure of a hypergraph \( H \) as the k-height subtree of its incidence graph \( \tilde{I}(H) \), where \( \tilde{I}(H) \) is the bipartite graph with vertices \( V(\tilde{I}(H)) := V \cup \mathcal{E} \) and edges \( E(\tilde{I}(H)) := \{ ve \mid v \in V \land e \in \mathcal{E} \} \).

Corollary 1. Assume that 1-GWL test can distinguish two distinct local substructures from hypergraphs, the UniGNN \( \mathcal{A} \) can also distinguish them as long as the Local Level condition is satisfied.

5 Experiments

In this section, we evaluate the performance of the proposed methods in extensive experiments.

Datasets. We use the standard academic network datasets: DBLP [Rossi and Ahmed, 2015], Pubmed, Citeseer and Cora [Sen et al., 2008] for all the experiments. The hypergraph is created with each vertex representing a document. The co-authorship hypergraphs, constructed from DBLP and Cora, connect all documents co-authored by one author as one hyperedge. The co-citation hypergraphs are built with PubMed, Citeseer and Cora, using one hyperedge to represent all documents cited by an author. We use the same preprocessed hypergraphs as HyperGCN, which are publicly available in their official implementation1.

---

1https://github.com/malllabiisc/HyperGCN
5.1 Semi-supervised Hypernode Classification

**Setting Up and Baselines.** The semi-supervised hypernode classification task aims to predict labels for the test nodes, given the hypergraph structure, all nodes' features and very limited training labels. The label rate of each dataset can be found in the supplemental materials.

We employ four two-layer UniGNN variants: UniGCN, UniGAT, UniGIN and UniSAGE. For all models, mean function is used as the first-stage aggregation. UniSAGE uses the SUM function for the second-stage aggregation. Note that as described in Section 3.1, hypergraphs are preprocessed with the SUM function for the second-stage aggregation. Note that as described in Section 3.1, hypergraphs are preprocessed with the SUM function for the second-stage aggregation.

We compare UniGNN models against the following baselines: (a) Multi-Layer Perceptron with explicit Hypergraph Laplacian Regularization (MLP+HLR), (b) HyperGraph Neural Network (HGNN [Feng et al., 2019]), (c) HyperGraph Convolutional Network (HyperGCN [Yadati et al., 2019]) and (d) HyperSAGE [Arya et al., 2020].

Closely following the previous works, for each model on each dataset, we repeat experiments over 10 data splits with 8 different random seeds, amounting to 80 experiments. We use the Adam optimizer with a learning rate of 0.01 and the weight decay of 0.0005. We fix the training epochs as 200 and report the performance of the model of the last epoch. The same training/testing split as [Yadati et al., 2019] is used. We run all experiments on a single NVIDIA 1080Ti(11GB).

**Comparison to SOTAs.** Table 1 summarizes the mean classification accuracy with the standard deviation on the test split of UniGNN variants after 80 runs. We reuse the metrics that are already reported in [Arya et al., 2020] for MLP+HLR, HGNN, HyperGCN and the best metrics reported for HyperSAGE.

Results in Table 1 demonstrate that UniGNNs are consistently better than the baselines with a considerable lift, achieving a new state-of-the-art. Especially for the DBLP dataset, we significantly improve the accuracy from 77.4% to 88.8% with negligible variance. On all datasets, our results are generally more stable than the baselines, as indicated by lower standard deviation. Whereas for the Cora co-citation dataset, we report only slight improvements against SOTA, we argue that this is due to the fact that the Cora co-citation hypergraph contains the least mean hyperedge size $|M| = 3.0 \pm 1.1$, for which the information loss from clique expansion in HGNN might be negligible.

Overall, with the powerful aggregation designs inspired from GNNs, our UniGNN models can effectively capture the intrinsic structure information from hypergraphs and perform stably better prediction with less deviation.

**Effect of Self-loops for UniGCN and UniGAT.** We further study the effect of self-loops for UniGCN and UniGAT. Table 4 reports the mean accuracies for UniGCN and UniGAT when input hypergraphs are with or without self-loops. We observe that when hypergraphs are un-self-looped, the performances

<table>
<thead>
<tr>
<th>Method</th>
<th>DBLP seen</th>
<th>DBLP unseen</th>
<th>Pubmed seen</th>
<th>Pubmed unseen</th>
<th>Citeseer seen</th>
<th>Citeseer unseen</th>
<th>Cora(cocitation) seen</th>
<th>Cora(cocitation) unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP+HLR</td>
<td>64.5</td>
<td>58.7</td>
<td>66.8</td>
<td>62.4</td>
<td>60.1</td>
<td>58.2</td>
<td>65.7</td>
<td>64.2</td>
</tr>
<tr>
<td>HyperSAGE</td>
<td>78.1</td>
<td>73.2</td>
<td>81.0</td>
<td>80.4</td>
<td>69.3</td>
<td>67.9</td>
<td>71.3</td>
<td>66.8</td>
</tr>
<tr>
<td>UniGCN</td>
<td>88.4</td>
<td>82.7</td>
<td>83.5</td>
<td>83.4</td>
<td>70.9</td>
<td><strong>71.3</strong></td>
<td>72.4</td>
<td>70.1</td>
</tr>
<tr>
<td>UniGCN*</td>
<td>88.4</td>
<td>82.6</td>
<td>83.7</td>
<td>83.3</td>
<td>71.2</td>
<td>70.6</td>
<td><strong>74.3</strong></td>
<td>71.5</td>
</tr>
<tr>
<td>UniGIN</td>
<td>89.6</td>
<td><strong>83.4</strong></td>
<td>83.8</td>
<td>83.3</td>
<td><strong>71.5</strong></td>
<td>70.8</td>
<td>73.7</td>
<td>71.3</td>
</tr>
<tr>
<td>UniSAGE</td>
<td>89.3</td>
<td>83.0</td>
<td>83.6</td>
<td>83.1</td>
<td>71.1</td>
<td>70.8</td>
<td><strong>74.2</strong></td>
<td>71.5</td>
</tr>
</tbody>
</table>

Table 2: Testing accuracies of multiple UniGNN variants and other hypergraph learning methods for Inductive Learning on Evolving Hypergraphs. The best or competitive results are highlighted for each dataset.
Table 3: Testing accuracies(%) of multiple UniGNN variants with Cocitation and use the best results reported from [Arya et al., 2020] and use the cor-
rectness of the formulations in Section 3.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Models</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>DBLP</td>
<td>UniGAT</td>
<td>89.1</td>
</tr>
<tr>
<td></td>
<td>UniGCN</td>
<td>89.2</td>
</tr>
<tr>
<td></td>
<td>UniGIN</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>UniSAGE</td>
<td>89.4</td>
</tr>
<tr>
<td></td>
<td>UniGCNII</td>
<td>88.4</td>
</tr>
<tr>
<td>Cora</td>
<td>UniGAT</td>
<td>76.0</td>
</tr>
<tr>
<td>Coauthor</td>
<td>UniGCN</td>
<td>76.2</td>
</tr>
<tr>
<td></td>
<td>UniGIN</td>
<td>75.8</td>
</tr>
<tr>
<td></td>
<td>UniSAGE</td>
<td>75.9</td>
</tr>
<tr>
<td></td>
<td>UniGCNII</td>
<td>75.1</td>
</tr>
<tr>
<td>Pubmed</td>
<td>UniGAT</td>
<td>75.2</td>
</tr>
<tr>
<td></td>
<td>UniGCN</td>
<td>74.9</td>
</tr>
<tr>
<td></td>
<td>UniGIN</td>
<td>74.8</td>
</tr>
<tr>
<td></td>
<td>UniSAGE</td>
<td>74.8</td>
</tr>
<tr>
<td></td>
<td>UniGCNII</td>
<td>75.6</td>
</tr>
<tr>
<td>Citeese</td>
<td>UniGAT</td>
<td>65.4</td>
</tr>
<tr>
<td></td>
<td>UniGCN</td>
<td>64.5</td>
</tr>
<tr>
<td></td>
<td>UniGIN</td>
<td>64.6</td>
</tr>
<tr>
<td></td>
<td>UniSAGE</td>
<td>65.0</td>
</tr>
<tr>
<td></td>
<td>UniGCNII</td>
<td>64.1</td>
</tr>
<tr>
<td>Cora</td>
<td>UniGAT</td>
<td>70.9</td>
</tr>
<tr>
<td>Coclitation</td>
<td>UniGCN</td>
<td>71.2</td>
</tr>
<tr>
<td></td>
<td>UniGIN</td>
<td>70.9</td>
</tr>
<tr>
<td></td>
<td>UniSAGE</td>
<td>71.4</td>
</tr>
<tr>
<td></td>
<td>UniGCNII</td>
<td>70.0</td>
</tr>
</tbody>
</table>

Table 3: Testing accuracies(%) of multiple UniGNN variants with different depths in Semi-supervised Hypernode Classification task. The result of the best performed model for each dataset is bolded. Note that additional validation data are used in this experiment.

5.2 Inductive Learning on Evolving Hypergraphs

Setting Up. The task for inductive learning on evolving hypergraph takes the historical hypergraph as input and predicts the unseen nodes’ labels.

We closely follow [Arya et al., 2020] and use the corrupted hypergraph which randomly removes 40% vertices as unseen data during training. 20% vertices are used for training and the rest 40% for the seen part of testing vertices. The other experimental settings are similar to those in the transductive semi-supervised learning task. We employ an additional UniGCN variant, denoted as UniGCN*, which applies the linear transform after aggregation and normalization. We compare our models against MLP+HLR and HyperSAGE and use the best results reported from [Arya et al., 2020].

Comparison to SOTAs. Table 2 reports the mean classification accuracy on seen part and unseen part of the testing data. We observe that our UniGNN models consistently show better scores across the benchmark datasets. Similar to the semi-supervised setting, our models notably show significant improvements in dataset DBLP, where the prediction accuracy increases from 78.1% to 89.6% in the seen data and from 73.2% to 83.4% in the unseen data.

Results from table 2 confirm that UniGNNs can capture the global structure information and perform well for predicting unseen nodes in the inductive learning task.

5.3 Performance of Deep-layered UniGNNs

Setting Up. To verify the effectiveness of UniGCNII, we study how the performance changes for vanilla UniGNN models with various depths. In this experiment, we use the same setting as described in the semi-supervised hypernode classification task, except that additional 20% of the original testing split is used as the validation split.

For UniGCNII, we perform all experiments in 1000 epochs and early stopping with a patience of 150 epochs. We use the Adam Optimizer with a learning rate of 0.01. We set the L2 regularizer factor to 0.01 for the convolutional layers, 0.0005 for the dense layer, which is the same as described in GCNII [Chen et al., 2020]. Please refer to the supplemental materials for more details.

Comparison with Other Deep-layered UniGNNs. Table 3 summarizes the results, in which the best performed model for each dataset is bolded. We see that UniGCNII enjoys the benefit of deep network structures and shows generally better results as layers increase. We highlight that UniGCNII outperforms the best shallow models in dataset Cora, Pubmed and Citeeseer, and obtains competitive results in dataset DBLP. On the contrary, the performance of vanilla models drop significantly as depths increase.

Overall, the results suggest that our proposed framework is capable of incorporating meticulously-designed deep GNN models for deep hypergraph learning.

6 Conclusion

We propose the UniGNN, a unified framework for graph and hypergraph neural networks. Under this framework, we naturally generalize several classic GNNs to HyperGNNs, which consistently show stably better performances than recent state-of-the-art methods. We firstly solve the oversmoothing problem of deep hypergraph neural networks by presenting the UniGCNII. Our models learn expressive representation of hypergraphs, which can be beneficial for a broad range of downstream tasks. Future works include designing provably more powerful UniGNNs with high-order GWL test. Another interesting direction for future work is to design hypergraph subtree kernel for hypergraph classification.
References


