# FEATURE-AUGMENTED HYPERGRAPH NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph neural networks (GNNs) and their variants have demonstrated superior performance in learning graph representations by aggregating features based on graph or hypergraph structures. However, it has become evident that most existing graph-based GNNs are susceptible to over-smoothing and are non-robust to perturbations. For representation learning tasks, hypergraphs usually have more expressive power than graphs through their ability to encode higher-order data correlations. In this paper, we propose Feature-Augmented Hypergraph Neural Networks (FAHGNN) focusing on hypergraph structures. In FAHGNN, we explore the influence of node features for the expressive power of GNNs and augment feature representations by introducing common features and personal features to model information. Specifically, for a node, the common features contain the shared information with other nodes in hyperedges, while the personal features represent its special information. In this way, the introduced feature types possess different distinguishing powers. Considering the different properties of these two kinds of features, we design different propagation strategies for information aggregation on hypergraphs. Furthermore, during the propagation process, we further augment features by randomly dropping node features. We leverage consistency regularization across different data augmentations of the two feature types to optimize the prediction consistency for the model. Extensive experiments on several benchmarks show that FAHGNN significantly outperforms other state-of-the-art methods for node classification tasks. Our theoretical study and experimental results further support the effectiveness of FAHGNN for mitigating issues of over-smoothing and enhancing robustness of the model.

## 1 INTRODUCTION

Graphs are ubiquitous across the real world, such as social networks, biological networks, and e-commerce networks. Graph neural networks (GNNs) have shown superiority in using both the graph structure and node features to produce a vectorial representation, which can be used for various prediction tasks on graph data. In this work, we focus on semi-supervised node classification (Kipf & Welling (2017); Zhu et al. (2003)). Most popular GNNs (Kipf & Welling (2017); Velickovic et al. (2018); Hamilton et al. (2017); Li et al. (2018b)) learn expressive node representations by designing effective feature propagation strategies based on a graph structure.

Considering that the relationship among entities could go beyond pairwise connections in real-world problems, it is desirable to capture complicated relations among objects for graph representations. The hypergraph, which can encode higher-order correlations of multiple entities by hyperedges, has gained recent attention. A series of GNNs based on hypergraphs (Feng et al. (2019); Jiang et al. (2019); Bai et al. (2021); Yadati et al. (2019)) have been proposed using different neighborhood-aggregation schemes in order to achieve greater modelling expressiveness.

However, recent studies have pointed out that neighborhood-information propagation procedures suffer from over-smoothing (Li et al. (2018a); Wu et al. (2019)). For example, standard graph convolutional networks (GCNs) stack layers of learned first-order spectral filters, and as the depth grows, the features of nodes become indistinguishable (Li et al. (2018a); Oono & Suzuki (2020)). This issue hinders existing GNNs from learning an effective node representation from deep layers.

Given that they also follow a message passing scheme, GNNs on hypergraphs are also susceptible to these issues, but this has been largely unexplored thus far.

For structural data in which structure information supplements node features, GNNs' distinguishing power can be evaluated: two vertices represented by the same feature are considered the same with respect to subsequent feature-based prediction tasks. Under the assumption of homophily of graph data (McPherson et al. (2001)), the vertices closed in the feature domain tend to be connected. A hypergraph captures the similarities of nodes into a hyperedge, i.e., the nodes in a hyperedge have similar information with each other and are distinguished from others outside the hyperedge. For each node, we refer to this information shared with other nodes as its common feature. As a single object, each node naturally has its own personality different from other nodes even in a hyperedge. We therefore also introduce personal features to represent the special information about each node. For example, in a social network, some students are connected into a group (as a hyperedge) because of their common interest in basketball (the attribute to construct hyperedges), while they also each have their own special interests such as music, dance, and so on. Here, the feature indicated by basketball (a hyperedge) is the common feature of these students and the features presented by music and dance are their personal features. Using these definitions, these two feature types possess different distinguishing powers before the propagation procedure.

Whilst augmenting the original features with common features and personal features can provide new perspectives, a key question is how these new feature types perform during propagation strategies for node representations. We study the performance of these two kinds of features with increasing depth of GNNs and exploit their advantages to achieve powerful expressiveness, mitigate over-smoothing and obtain a robust model.

In this work, we present Feature-Augmented Hypergraph Neural Networks (FAHGNN), a hypergraph-based semi-supervised learning framework. We consider enhancing the expressive of node representations by augmenting node features. Augmenting original features into common and personal features can naturally enhance expressive power as common information makes nodes indistinguishable while personal information make nodes distinguishable. In other words, our proposed FAHGNN can leverage more expressive information from these two feature type, compared to existing GNN methods that just use original features. Compared to most GNNs utilizing a deterministic message-passing scheme, we augment data by randomly dropping some nodes' features. Thus, in the propagation process, the node features are insensitive to deterministic neighborhoods. Moreover, this allows feature propagation without nonlinear transformation. Thus, multi-layer feature propagation can benefit from higher-order features without increasing the risk of over-smoothing for FAHGNN.

Targeting specific neighborhood-aggregation, we develop a novel spectral convolution operation on the common features to exploit higher-order hyperedge correlations for representation learning, while the personal information is propagated by convolutions based on the hypergraph Laplacian. With these updated features, we employ a simple Multilayer Perception (MLP) to predict unlabeled nodes. Leveraging the idea that a classifier should output the same class distribution for an unlabeled example even after it has been augmented, we utilize consistency regularization to enhance similar predictions on multiple augmentations with the two feature types. We theoretically validate the effect of the common feature and personal feature on over-smoothing and illustrate that data augmentation with dropping features and consistency regularization can enforce the consistency of classification confidence between each node and its multi-hop neighborhoods. Experimental results demonstrate FAHGNN can achieve better classification accuracy on four benchmark datasets and our proposed methods can mitigate the issues of over-smoothing and can be more robust.

## 2 Related Works

**Graph Neural Networks.** GNNs (Scarselli et al. (2009); Kipf & Welling (2017)) have emerged by extending neural techniques to learning tasks on graph-structured data. Using both structural information and original features, representations of nodes are computed by repeatedly aggregating information of neighborhoods using some deterministic propagation rules. According to spectral graph theory, spectral-based GNNs (Henaff et al. (2015); Defferrard et al. (2016); Yuan et al. (2019); Kipf & Welling (2017); Velickovic et al. (2018)) developed graph convolution in the Fourier domain. Standard GCNs adopt first-order spectral filtering based on the Laplacian matrix with the layer-

wise propagation rule $\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$, where $\hat{\mathbf{A}}$ is the symmetric normalized adjacency matrix with added self-connections, $\mathbf{H}^{(l)}$ is the hidden node representation in $l^{th}$ layer, $\mathbf{W}^{(l)}$ is the weight matrix to be learned in the training process and $\sigma(\cdot)$ denotes the nonlinear activation function, *i.e.*, the ReLU function. To address the scalability problem of spectral approaches, spatial-based GNNs (Atwood & Towsley (2016); Niepert et al. (2016); Hamilton et al. (2017); Monti et al. (2017)) work with convolutions directly on the graph, operating on groups of spatially close neighbors. Meanwhile, sampling-based techniques have been developed (Hamilton et al. (2017); Chen et al. (2018); Huang et al. (2018)) for fast graph representation learning.

In regard to studies for deep GNNs, it has been observed that the best performance of GCNs (Kipf & Welling (2017)) is achieved with 2 layers and one encounters severe performance degradation when the depth grows too large. Li et al. (2018a) reported that the failure of deep GCNs is due to the node features becoming indistinguishable with increasing depth, i.e., over-smoothing. Oono & Suzuki (2020) explained over-smoothing as convergence to Laplacian sub-eigenspaces when the depth increases to infinity. Based on the explanations in Oono & Suzuki (2020), Rong et al. (2020) proposed DropEdge to impede over-smoothing. A recent method (Feng et al. (2020)) proposed GRAND with augmented graph data and employed regularization to the augmentations for node classification on graphs. It mitigated the over-smoothing and showed the robustness of model on graph learning. However, few works have studied the challenges of over-smoothing on hypergraphs.

**Hypergraph Learning.** For certain applications, a simple graph has limitations for representing the higher-order connections of multiple vertices. Hypergraph learning addresses this problem. In a hypergraph, the complex relationships are encoded by hyperedges that can connect two or more vertices. Hypergraphs have been widely used (Zhou et al. (2006); Gao et al. (2012); Sun et al. (2008); Li & Milenkovic (2017)) due to their great influence on modeling high-order correlations. Motivated by the success of hypergraph applications for nonstructural data, some works have aimed to fully explore the complicated correlations among structural data by a hypergraph. Such works (Feng et al. (2019); Bai et al. (2021); Yadati et al. (2019)) have designed neighborhood aggregation strategies based on hypergraphs and have shown superiority in higher-order data representation for graph learning. Following a similar message passing scheme, hypergraph-based GNNs also suffer from the over-smoothing issue. In this work, we consider graph data under the homophily assumption (McPherson et al. (2001)), and thus structural information corresponds to features (edges/hyperedges connect nodes with similar features). Therefore, hyperedges present structural information, as well as similar (additional) information in features (explicitly by hyperedge features or implicitly by hyperedge generating methods). Though the advantages of hypergraphs for modelling high-order data correlations have been demonstrated in proof of concept studies for existing GNNs, it is still very challenging to fully explore and exploit the information in hypergraphs.

## 3 Feature-Augmented Hypergraph Neural Networks

We present Feature-Augmented Hypergraph Neural Networks (FAHGNN) for semi-supervised learning based on hypergraphs, as illustrated in Figure 1. Inspired by inherent properties of hypergraphs, we model the original features of the input hypergraph by both common features and personal features, thus obtaining greater expressiveness. In the information propagation procedure, we further augment features by DropFeature (drop node and hyperedge features). Leveraging the idea that a classifier should output the same class distribution for an unlabeled example even after it has been augmented, we use consistency regularization to predict outputs.

### 3.1 Hypergraph Formalism

Given an input hypergraph $G = (V, E)$ with vertex set $V$ and hyperedge set $E$, each hyperedge is defined as a subset of vertices. For each node $v_i \in V$, we define the personal features as its personal information presenting the differences with other nodes, while the common features are its public information sharing with some nodes in hyperedges. Note that, most times, the hyperedge feature represents the public information of nodes in a hyperedge, but a node may be involved in several hyperedges. Therefore, a node's common feature may be obtained from multiple hyperedge features. For some hypergraphs, we can directly obtain the common features and personal features by division of the original node features, which is discussed later in the experiments. Here, we

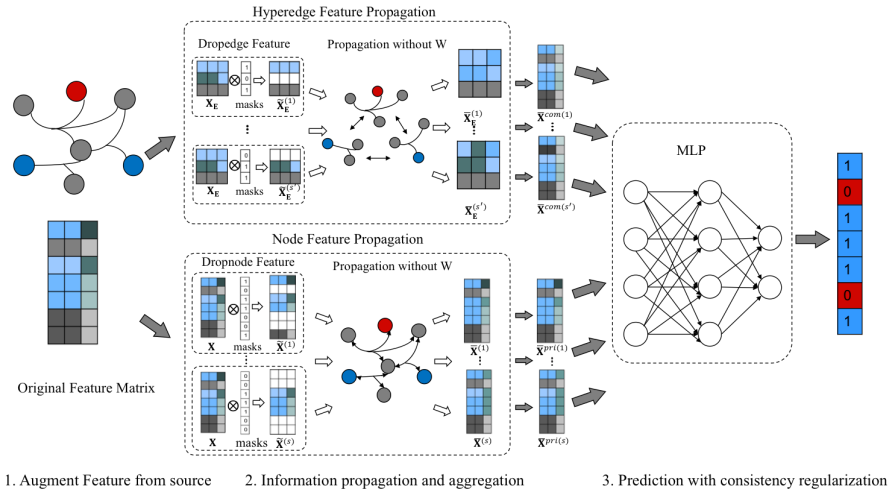



Figure 1: The Framework of FAHGNN. 1. Augment features from the source by introducing common features (hyperedge features) and personal features (node features). 2. Different information propagation and aggregation strategies for the two feature types with further feature augmentation. 3. 2-Layer MLP for prediction with supervised and consistency regularized loss.

present a general case. We denote the node feature matrix $\mathbf{X} \in \mathbb{R}^{N\times d}$ and the hyperedge feature matrix $\mathbf{X_E} \in \mathbb{R}^{M\times d^{'}}$. For node $v_i \in V$, its personal features are denoted as $x_i^p \in \mathbb{R}^{d\times 1}$ while its common features are defined as $x_i^c \in \mathbb{R}^{d^{'}\times 1}$.

For hypergraph $G$, the correlations of vertices and hyperedges can be represented in a $|V| \times |E|$ incidence matrix $\mathbf{H}$ with entries $h(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e. \end{cases}$ Let $w(e)$ be the weight corresponding to the hyperedge $e$. The degrees of $v$ and $e$ can be defined as $d(v) = \sum_{e\in E} w(e)h(v,e)$ and $d(e) = \sum_{v\in V} h(v,e)$, respectively. We define the matrix $\mathbf{\Theta} = \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$, and then the standard hypergraph Laplacian $\mathbf{\Delta}$ (Zhou et al. (2006)) can be defined as $\mathbf{\Delta} = \mathbf{I} - \mathbf{\Theta}$. Here, $\mathbf{W}$ is a diagonal matrix representing the weights of hyperedges. The diagonal matrices $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the vertex and hyperedge degrees, respectively. Note that the hypergraph Laplacian is positive semidefinite. In spectral-based GNNs on hypergraphs (Feng et al. (2019)), the convolution operation is the first-order filter based on hypergraph Laplacian by matrix $\mathbf{\Theta}$. Analogous to the adjacency relationship of nodes, we define the following hyperedge correlations by matrix $\mathbf{\Theta_E} = \mathbf{D}_e^{-1/2}\mathbf{H}^\top\mathbf{D}_v^{-1}\mathbf{H}\mathbf{D}_e^{-1/2}$. We demonstrate that it is positive semidefinite in Appendix B.

### 3.2 Augmentation for Prediction

Given a hypergraph $G$ with node feature matrix $\mathbf{X}$ and hyperedge feature matrix $\mathbf{X_E}$, we adopt a data augmentation method to augment features in the information propagation process. For generated augmentations $\overline{\mathbf{X}}$ and $\overline{\mathbf{X}}_\mathbf{E}$ associated with originals $\mathbf{X}$ and $\mathbf{X_E}$, we combine them into a classification model for unseen node prediction.

In information propagation, we design different propagation rules for the two feature types and address oversmoothing by a series of approaches including a dropping strategy, mixing higher-order features, and propagation without nonlinear transformation. Also, we use both supervised loss and consistency regularized loss for better prediction.

**Information Propagation.** For two input features, we first adopt DropFeature. By randomly dropping out entire feature vectors of some nodes and hyperedges, we get perturbed feature matrices $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{X}}_\mathbf{E}$. We randomly sample a binary mask $\epsilon_i \sim \text{Bernoulli}(1-\delta)$ for each node or hyperedge, and obtain the perturbed node feature matrix $\widetilde{\mathbf{X}}_i = \epsilon_i \cdot \mathbf{X}_i$. Then, we scale using the factor of $\frac{1}{1-\delta}$ to guarantee the perturbed feature matrix is in expectation equal to $\mathbf{X}$. Similarly, we obtain the

perturbed hyperedge feature matrix. Note that sampling is only performed during training. During inference, we directly set $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{X}}_{\mathbf{E}}$ as the input feature matrices. There are other ways to generate perturbed feature matrices, such as a direct dropout strategy (Srivastava et al. (2014)). However, this strategy ignores the structural information that may lead to a noisy feature matrix. By comparison, dropping entire feature vectors of nodes allows generating more stochastic data augmentations by completely ignoring some nodes' features, and thus helps increase the model's robustness.

Then, we design different propagation rules for these two perturbed features $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{X}}_{\mathbf{E}}$ to obtain data augmentations. The spectral convolution on node feature is described as Feng et al. (2019):

$$\overline{\mathbf{X}}^{(k+1)} = \mathbf{\Theta}\overline{\mathbf{X}}^{(k)}. \tag{1}$$

The convolution operation on hyperedges is defined by

$$\overline{\mathbf{X}}_{\mathbf{E}}^{(k+1)} = \mathbf{\Theta}_{\mathbf{E}}\overline{\mathbf{X}}_{\mathbf{E}}^{(k)}. \tag{2}$$

Where $\overline{\mathbf{X}}^{(k)}$ is the node representation in $k^{th}$ layer with $\overline{\mathbf{X}}^{(0)} = \widetilde{\mathbf{X}}$, and $\overline{\mathbf{X}}_{\mathbf{E}}^{(k)}$ is the hyperedge feature in $k^{th}$ layer with $\overline{\mathbf{X}}_{\mathbf{E}}^{(0)} = \widetilde{\mathbf{X}}_{\mathbf{E}}$. Note that this information propagation is without non-linear transformation.

Here, instead of just using the features in the final layer, we mix $k$-order propagation features to obtain augmented node features $\overline{\mathbf{X}} = \sum_{k=0}^{K} \frac{1}{K+1}\overline{\mathbf{X}}^{k}$, and $\overline{\mathbf{X}}_{\mathbf{E}} = \sum_{k=0}^{K^{'}} \frac{1}{K^{'}+1}\overline{\mathbf{X}}_{\mathbf{E}}^{k}$. In this way, the mixed-order propagation features incorporate more shallow layer information to reduce the risk of over-smoothing.

In this way, the feature of each node or hyperedge is not a single deterministic vector. Each node's or hyperedge's features is augmented by random mixes of information from its neighbors. Under the homophily assumption, for each node or hyperedge, the augmentations are approximate representations.

**Prediction with augmentations.** Assume generating a total of $S$ augmentations including $S^N$ augmented node feature matrices and $S^E$ hyperedge feature augmentations. The node feature augmentation can be directly used as personal features $X^p$ for prediction, while we need to obtain the node representation from public information of multiple hyperedges. Taking an average of augmented features of hyperedges which contain node $i$, we obtain the common feature representation of it $x_i^c = \mathbf{H}_i\overline{\mathbf{X}}_{\mathbf{E}}$, where $\mathbf{H}_i$ denotes the $i^{th}$ row vector of $\mathbf{H}$. The whole common feature matrix is obtained by $\overline{\mathbf{X}} = \mathbf{H}\overline{\mathbf{X}}_{\mathbf{E}}$. Then we apply a two-layer MLP to get the corresponding outputs:

$$\mathbf{Z}^{(s)} = f_{MLP}\left(\overline{\mathbf{X}}^{(s)}, W\right), 1 \leq s \leq S, \tag{3}$$

where $\mathbf{Z}^{(s)} \in [0,1]^{N\times C}$ is the prediction probabilities on $\overline{\mathbf{X}}^{(s)}$ and $W$ are the model parameters.

### 3.3 Semi-supervised Classification with Regularization

Considering a classifier should output the same class distribution (Berthelot et al. (2019)) for an unlabeled sample even it has been augmented, it is natural to design a consistency regularized loss (Deng et al. (2019); Feng et al. (2021; 2020)) for semi-supervised learning.

Assume there are $l$ labeled nodes with labels $\mathbf{Y}_i \in \{0,1\}^{N\times C}$ ($C$ is the number of class) among $N$ nodes, the supervised loss for each epoch is defined as the average cross-entropy loss over $S$ augmentations:

$$\mathcal{L}_{sup} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{l-1}\mathbf{Y}_i^{\top}\log\mathbf{Z}_i^{(s)}. \tag{4}$$

With $S$ outputs corresponding to $S$ augmentations, we propose to optimize the prediction consistency among the multiple outputs for unlabeled data. Considering scenario of two augmentations, we minimize the distance between two outputs $\min\sum_{i=0}^{N-1}\left\|\mathbf{Z}_i^{(1)} - \mathbf{Z}_i^{(2)}\right\|_2^2$. For a situation

with multiple augmentations, we average all distributions to achieve the label distribution centre $\overline{\mathbf{Z}}_i = \frac{1}{S}\sum_{s=1}^{S}\mathbf{Z}_i^{(s)}$. Then, based on the average distributions, the probability of node $i$ on class $j$ is computed $\overline{\mathbf{Z}}'_{ij} = \overline{\mathbf{Z}}_{ij}^{\frac{1}{T}} / \sum_{c=0}^{C-1}\overline{\mathbf{Z}}_{ic}^{\frac{1}{T}}, (0 \leq j \leq C-1)$, where $0 < T \leq 1$ is used to control the categorical distribution (Berthelot et al. (2019)). Thus, the consistency regularization can be defined as:

$$\mathcal{L}_{con} = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{N-1}\left\|\overline{\mathbf{Z}}'_i - \mathbf{Z}_i^{(s)}\right\|_2^2. \tag{5}$$

In each epoch, we define the loss function by combining the supervised classification loss and the consistency regularization loss: $\mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{con}$, where $\lambda$ is a hyper-parameter.

**Complexity.** The complexity of information propagation is $\mathcal{O}(S^N K_1 d(N + |E|) + S^E K_2 d^{'}(M + |E^{'}|))$, where $S^N$ and $S^E$ denote the number of node and hyperedge feature augmentations, respectively; $K_1$ and $K_2$ denote the propagation layers of node and hyperedge, respectively; $d$ and $d^{'}$ are the dimension of node feature and hyperedge feature, respectively; $N$ and $M$ are the number of nodes and hyperedges and $|E|$ and $|E^{'}|$ denote connecting counts of nodes and hyperedges. Employing a two-layer MLP, the complexity of node personal feature prediction is $\mathcal{O}\left(S^N N d_h(d + C)\right)$ and the complexity of node common feature prediction is $\mathcal{O}\left(S^E M d_h(d^{'} + C)\right)$ where $d_h$ denotes the hidden size and $C$ is the number of classes.

### 3.4 Theoretical Analysis

We theoretically analyse: i) the different performances of personal information and common information in the propagation process while increasing the depth of GNNs and ii) the positive effects of data augmentation and consistency regularization on mitigating over-smoothing.

It has been explained that the node features converge to a subspace (Oono & Suzuki (2020)), thus meaning in that node representations become indistinguishable (known as over-smoothing) as the network depth increases. Based on this concept, we provide several relevant definitions that will facilitate our later presentations.

**Definition 1** (subspace). *Let $\mathcal{M} := \left\{\boldsymbol{EC} \mid \boldsymbol{C} \in \mathbb{R}^{M\times C}\right\}$ be an M-dimensional subspace in $\mathbb{R}^{N\times C}$, where $\boldsymbol{E} \in \mathbb{R}^{N\times M}$ is orthogonal with $\boldsymbol{E}^\top\boldsymbol{E} = \boldsymbol{I}_M$ , and $M \leq N$.*

**Definition 2** ($\epsilon$-smoothing). *We say that $\epsilon$-smoothing of node features happens for a GCN, if all its hidden vectors $H^l$ beyond a certain layer $L$ have a distance no larger than $\epsilon(\epsilon > 0)$ with respect to a subspace $\mathcal{M}$ that is independent to the input features, namely,*

$$d_{\mathcal{M}}\left(\boldsymbol{H}^{(l)}\right) < \epsilon, \forall l \geq L, \tag{6}$$

*where $d_{\mathcal{M}}(\cdot)$ computes the distance between the input matrix and the subspace $\mathcal{M}$.*

With these preparations, we introduce a theorem for the propagation of common feature and personal feature. As common feature matrix $\mathbf{X}^c \in \mathbb{R}^{N\times C}$ indicates the similarity of nodes in the feature domain, and personal feature matrix $\mathbf{X}^p \in \mathbb{R}^{N\times C^{'}}$ describes the differences of node features. We can therefore conclude:

**Proposition 1.** *According to the properties of common features and personal features, the dimension of the common feature subspace is smaller than that of personal features subspace, i.e., $dim(\mathbf{X}^c) \leq dim(\mathbf{X}^p)$.*

**Lemma 1.** *Assume there are two matrix $\mathbf{A} \in \mathbb{R}^{F\times C}$ and $\mathbf{B} \in \mathbb{R}^{F\times C}$ with $dim(\mathbf{A}) \leq dim(\mathbf{B})$. When they multiply with a matrix $\mathbf{C} \in \mathbb{R}^{C\times D}$ which is reversible, the dimension of matrix subspace of $\mathbf{AC}$ is also no larger than $\mathbf{BC}$, i.e., $dim(\mathbf{AC}) \leq dim(\mathbf{BC})$.*

**Theorem 1.** *In the propagation process of GCNs on hypergraphs, the hidden vector of personal features converges to subspace $\mathcal{M}$ more slowly than common feature when increasing the network depth, before a certain layer $L$. In other words, personal features can reduce the convergence speed of over-smoothing compared to common features.*

We give a detailed proof of Theorem 1 in Appendix B.2. Next, we discuss the positive effects achieved through data augmentation and consistency regularization. We consider the simple case that the MLP applied in the model has a single output layer, and it is a binary classification task, thus, the output of personal node feature $\mathbf{Z} = \mathrm{sigmoid}(\mathbf{\Theta}\widetilde{\mathbf{X}}W)$. For each node $i$, the corresponding conditional distribution is $z_i^{y_i}\left(1-z_i\right)^{1-y_i}$, where $z_i \in \mathbf{Z}$ and $y_i \in \{0,1\}$. We assume the augmentations with $S = 2$, then, the consistency regularization loss is simplified to $\mathcal{L}_{\text{con}} = \frac{1}{2}\sum_{i=0}^{N-1}\left(z_i^{(1)} - z_i^{(2)}\right)^2$, in which $z_i^{(1)}$ and $z_i^{(2)}$ are two outputs of augmentations. Based on these representations, we have the theorems as follows.

**Theorem 2.** *In expectation, the consistency loss $\mathcal{L}_{con}$ is approximate to a regularization term:* $\mathbb{E}_\epsilon\left(\mathcal{L}_{con}\right) \approx \mathcal{R}^c(W) = \sum_{i=0}^{N-1} z_i^2\left(1-z_i\right)^2 \mathrm{Var}_\epsilon\left(\mathbf{\Theta}\widetilde{\mathbf{X}}W\right)$.

We now further discuss the regularization effect of data augmentation with respect to the supervised loss. Note that the supervised classification loss $\mathcal{L}_{sup}$ in FAHGNN refers to the perturbed classification loss with DropFeature, while the general supervised loss is $\mathcal{L}_{\text{org}} = \sum_{i=0}^{l-1} -y_i \log\left(z_i\right) - \left(1-y_i\right)\log\left(1-z_i\right)$. We have the following theorem about the perturbed supervised loss and original loss:

**Theorem 3.** *In expectation, optimizing the perturbed classification loss $\mathcal{L}_{sup}$ is equivalent to optimizing $\mathcal{L}_{org}$ with an extra regularization term* $\mathcal{R}(W) \approx \mathcal{R}^q(W) = \frac{1}{2}\sum_{i=0}^{l-1} z_i\left(1-z_i\right)\mathrm{Var}_\epsilon\left(\mathbf{\Theta}_i\widetilde{\mathbf{X}}W\right)$.

Employing DropFeature as the perturbation method, we provide the full details of Theorem 2 and 3 in the supplementary material B.3 and B.4, respectively.

## 4 Experiments

We compare our proposed FAHGNN with state-of-the-art graph-based GNNs and hypergraph-based GNNs methods.

**Datasets.** We conduct experiments on four benchmark datasets with graph-structure and hypergraph-structure. (1) Cora, Citeseer and Pubmed (Sen et al. (2008)) are three citation datasets with original graph structure. (2) A modified version of the 20-newsgroup dataset (Zhou et al. (2006)) with binary occurrence values for 100 words is used for text categorization. Although without pre-defined data structure, a hypergraph (Zhou et al. (2006); Bai et al. (2021)) is usually constructed according to attributes, i.e., words. The details of all datasets are listed in the supplemental materials A.1. The experimental setting for graph and hypergraph construction, common and personal feature selection, and parameters setting can be found in the supplemental materials A.2.

### 4.1 The Results

Table 1 shows the classification results of our proposed method and other state-of-the-art methods on four datasets. While three citation datasets are graph-structured data, the relationship among objects of 20-newsgroup can be naturally described by a hypergraph. The final performance of our proposed FAHGNN is measured by average classification accuracy over 10 runs.

From the results, we can observe that FAHGNN can significantly outperform other methods on all datsets. Note that FAHGNN_drop means directly adopting dropout for feature matrix in information propagation process. We can see that FAHGNN consistently achieves better results than FAHGNN_dropout. Specifically, for graph-structure datasets, hypergraph-based GNNs methods (i.e., HGNN, Hyper-Conv) can only achieve gains of approximate 0.1% and 1.2% in comparison to GCN. Using common or personal features, FAHGNN_common and FAHGNN_personal achieve significant improvements (3% and 4.3%, respectively). Together with these two feature types and consistency regularized loss, FAHGNN achieves 86.1% accuracy. Recent regularization methods including Dropedge and GRAND show advantages by mitigating the smoothing problem compared to other graph-based GNNs methods, however, the complicated data correlations among data cannot be well explored. It can be seen from the results of 20-newsgroup that hypergraph-based methods can generally achieve better performance than graph-based methods. Particularly, our proposed

Table 1: Classification accuracy (%) on four datasets

| Methods | Cora | Citeseer | pubmed | 20-newsgroup |
|---|---|---|---|---|
| **GCN** | 81.5 | 70.3 | 79.0 | 64.4 |
| **GAT** | 83.0±0.7 | 72.5±0.7 | 79.0±0.3 | 58.7 ±2.7 |
| **MixHop** | 81.9±0.4 | 71.4±0.8 | 80.8±0.6 | - |
| **GraphSAGE** | 78.9±0.8 | 67.4±0.7 | 77.8±0.6 | 58.3 ±2.9 |
| **FastGCN** | 81.4±0.5 | 68.8±0.9 | 77.6±0.5 | 62.2±1.3 |
| **Dropedge** | 82.8 | 72.3 | 79.6 | 61.1 |
| **GRAND** | **85.4±0.5** | **75.4±0.5** | **82.7±0.6** | 71.5±0.2 |
| **HGNN** | 81.6 | 71.6 | 80.1 | 74.08 |
| **Hyper-Conv** | 82.7±0.3 | 71.2±0.4 | 78.4±0.3 | 69.1 |
| **HyperGCN** | 80.2±0.8 | 68.9±0.5 | 78.2±0.4 | - |
| **FAHGNN** | **86.1±0.3** | **75.7±0.1** | **83.0±0.4** | **79.9±1.0** |
| **FAHGNN_dropout** | 85.3 ±0.4 | 74.8 ±0.1 | 82.5 ±0.4 | **78.6 ±2.4** |
| **FAHGNN**_common | 84.5±0.3 | 76.0±2.2 | 79.8±1.8 | **75.6±0.4** |
| **FAHGNN**_personal | **85.8±0.3** | **75.5±0.2** | **83.3±0.7** | 75.3± 0.8 |

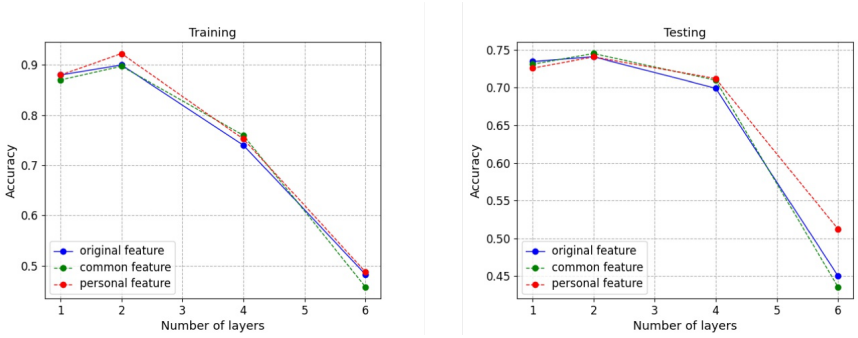
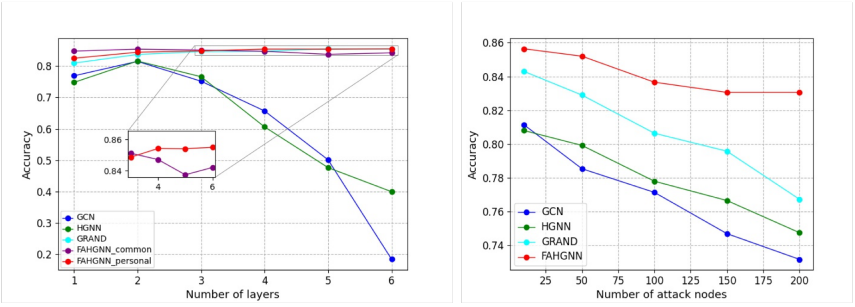



Figure 2: The performance of HGNN with different features when stacking layer

FAHGNN gains significant improvements over GCN and HGNN, 10% and 2%, respectively. From these results, we can see that using common features or personal features can yield better classification accuracy than using the original features. By augmenting features, FAHGNN consistently achieves superior performance on semi-supervised classification. We show the results of an ablation study to examine the contributions of different components of FAHGNN in Appendix A.3.

### 4.2 The effect of features on deep GNNs

We evaluate the effect of features on deep GNNs and demonstrate the influence of features on over-smoothing. Figure 2 shows the classification accuracy of HGNN with different input features on the 20-newsgroup dataset. The accuracy is observed to decrease when the depth of HGNN increases from 2. We can see that these three features achieve comparable results when the number of layers is small. However, with personal features as input features, HGNN can achieve the best training and testing accuracy when the depth is larger than 4. With common features, HGNN achieves lowest accuracy with 6 layer networks. For 20-newsgroup, the personal features of nodes represent more personal information by dropping common attributes, while common features represent more public information by dropping personal attributes. Personal features can reduce the convergence speed of over-smoothing compared to common features. Common features usually show better performance




(a) Over-Smoothing on Cora

(b) Attack on Cora

Figure 3: Over-Smoothing and robustness analysis on Cora.

in shallow layers, while personal feature get advantages in deep layers, therefore, augmenting original features into common and personal feature can help mitigate the oversmoothing problem.

### 4.3 Oversmoothing and Robustness analysis

GNNs face the over-smoothing issue for feature propagation steps or increasing layers. We study the performance of our proposed method for mitigating oversmoothing. Figure 3 (a) shows classification accuracy with respect to different propagation layers on Cora. In the GRAND and FAHGNN, the layer is controlled by the propagation steps, while for GCN and HGNN, it is adjusted by stacking different layers. The plots suggest that as the number of layers increases, the performance of GCN and HGNN decreases dramatically, independent of data structures, due to the over-smoothing issue. However, FAHGNN related methods including (FAHGNN_common and FAHGNN_personal) can benefit from stacking more propagation steps. This indicates that FAHGNN can relieve over-smoothing. From the inner subgraph of the picture, we observe that the performance of FAHGNN_personal with personal features as input features overtakes that of FAHGNN_common by increasing the depth of FAHGNN model, although FAHGNN_common achieves higher accuracy with shallow layers. The different performance of common information and personal information as stacking layers indicates their inherent different distinguishing powers. Fully utilizing the advantages of FAHGNN_personal in deep layers and the advantages of FAHGNN_common in shallow layers enables FAHGNN to achieve improvements for representation learning.

We further validate that FAHGNN is more robust facing adversarial attack by adding feature augmentation and regularization. The details of attacking can found in A.3. Figure 3 (b) illustrates the classification accuracies of different methods with respect to different numbers of attack nodes on the Cora dataset. We observe that FAHGNN achieves superior performance compared to GCN, HGNN and GRAND across all perturbation cases on both datasets. Especially, there is only a 3% drop in classification accuracy for FAHGNN with 200 attack nodes, while there are drops over 7% for other three methods. This study suggests that FAHGNN is more robust facing adversarial attack by adding feature augmentation and regularization.

## 5 Conclusion

We present Feature-augmented Hypergraph Neural Networks, a novel and more powerful GNN model on hypergraphs. Leveraging the expressive power of the hypergraph's original input features, we augment features into common and personal features. In this way, the hypergraph can represent richer information about nodes. We further augment features during the information propagation process by adopting a data augmentation method to mitigate the problem that the node representation highly depends on its special neighbourhoods. Fully using these augmentations, a consistency regularization is used to achieve better representation learning. Experimental results show that common features and personal features have different sensitivities to propagation layers. The strong performance of HFAHGNN on node classification demonstrates its advantages in terms of robustness and alleviating over-smoothing.

## REFERENCES

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1993–2001, 2016.

Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognit.*, 110:107637, 2021.

David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5050–5060, 2019.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. *CoRR*, abs/1902.09192, 2019. URL `http://arxiv.org/abs/1902.09192`.

Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Trans. Knowl. Data Eng.*, 33(6):2493–2504, 2021.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3558–3565. AAAI Press, 2019.

Yue Gao, Meng Wang, Dacheng Tao, Rongrong Ji, and Qionghai Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Trans. Image Process.*, 21(9):4290–4303, 2012.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4563–4572, 2018.

Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 2635–2641. ijcai.org, 2019.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2308–2318, 2017.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3538–3545. AAAI Press, 2018a.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3546–3553. AAAI Press, 2018b.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5425–5434. IEEE Computer Society, 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2014–2023. JMLR.org, 2016.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014.

Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pp. 668–676. ACM, 2008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 2019.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha P. Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1509–1520, 2019.

Yuan Yuan, Zhitong Xiong, and Qi Wang. ACM: adaptive cross-modal graph convolutional neural networks for RGB-D scene recognition. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 9176–9184. AAAI Press, 2019.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 5829–5836. AAAI Press, 2019.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pp. 1601–1608. MIT Press, 2006.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 1399–1407. ACM, 2019.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 912–919. AAAI Press, 2003.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 2847–2856. ACM, 2018.

# A Details of experiments

## A.1 Datasets Details

Table 2 summarizes the statistics of the four benchmark datasets including Cora, Citeseer, Pubmed and 20-newsgroup. We use exactly the same experimental settings on the three citation datasets for semi-supervised learning Kipf & Welling (2017). For 20-newsgroup, 20 samples per category are used for training with another 500 samples for validation purposes and 1000 samples for performance evaluation.

## A.2 Experiment Settings

In the experiments, we need to construct hypergraphs for citation networks, and a graph and a hypergraph for 20-newsgroup. In addition, common features and personal features are needed to be augment from the original features.

Table 2: Dataset Statistics

| Dataset | Type | Nodes | Edges | Classes | Features |
|---|---|---|---|---|---|
| Cora | graph network | 2708 | 5,429 | 7 | 1,433 |
| Citeseer | graph network | 3,327 | 4,732 | 6 | 3,703 |
| Pubmed | graph network | 19717 | 44338 | 3 | 500 |
| 20-newsgroup | hypergraph network | 16242 | - | 4 | 100 |

**Graph and hypergraph construction.** For three citation networks with simple graph structures, we just need to construct hypergraphs. Viewing each document as a hypernode, we generate a hyperedge for each hypernode with its 1-hop neighbors in a simple graph and $k$-nn neighbors with $k = 1$. Consequently, the generated hyperedges contain structural information and similar information in the feature domain. For 20-newsgroup dataset, the feature of each article is one-hot (according to the article contains the word or not). To generate a graph, each article is viewed as a vertex, we connect two vertices if there is at least one shared word. Note that we do not necessarily distinguish how many words they share. However, it is quite likely that multiple articles share the same word, thus naturally forming a hypergraph structure. Hence, using each word as attribute to connect multiple articles containing it to generate a hyperedge. As a result, the hyperedge features are word features just including a 1 element.

**Common features and personal features.** According to the definition of common feature and personal feature mentioned before, in a hypergraph, the common features can be explicitly represented by hyperedge features, or implicitly illustrated by different hyperedge generating stategies, while the personal feature is inherent. On citation datasets, we construct a hyperedge for each node which is viewed as a central node. Thus, all nodes in the hyperedge are similar to the central node, and the central node feature can be considered as a hyperedge feature, i.e., the common feature for all nodes in this hyperedge. Note that a node usually is involved in several hyperedges, and its common features are obtained by a linear combination (in Section 3.2). The personal feature for each vertiex is the original node feature.

Different from citation networks, we design a hyperedge according to attributes (words) for the modified 20-newsgroup dataset. Consequently, hyperedge features are naturally presented by word features. Because node features and hyperedge features are all based on attributes, we separate the node features into common features and personal features for each vertex based on the degrees of hyperedges. Generally, the hyperedges (words) with larger degree capture its public profile while the hyperedges with smaller degrees present its personal profile. Therefore, for each node $x_i$, we obtain its common features by dropping the elements corresponding to words with smaller degrees, i.e., $x_{ij} = 0$ if $d(\mathbf{H}_j)$ is the minimum degree for $x_{ij} > 0$ and $sum(x_i) \geq m$. In addition, the personal features are achieved by dropping the elements corresponding to words with larger degrees. In other words, we get the personal features by setting $x_{ij} = 0$ if $d(\mathbf{H}_j)$ is the maximum degree for $x_{ij} > 0$ and $sum(x_i) \geq m$. Here, $m$ is a parameter. It is easy to see that if a node's features just have several elements, dropping even one element will lead to a large information loss. Figure 4 shos the accuracy of HGNN (Feng et al. (2019)) with different values of $m$. The blue line shows HGNN accuracy with the original feature. Using more complex dropping element strategies, one can get more suitable common and personal features.

**Hyperparameter Details**

FAHGNN introduces several additional hyperparameters including propagation step $K$, data augmentation $S^E$, and DropFeature probability $\delta$ in data augmentation for the common feature and corresponding parameters $K^{'}$, $S^N$, and $\delta^{'}$ for the personal features. At each training epoch, we sharpen temperature $T$ to calculate consistency regularization loss and controlling the coefficient $\lambda$ to balance $\mathcal{L}_{sup}$ and $\mathcal{L}_{con}$. We perform hyperparameter search for each dataset, such as $K$ and $K^{'}$ from 2 to 10, $S^E$ and $S^N$ from 2 to 10. We usually set the drop rate for the node feature as 0.5, and drop rate for the hyperedge feature as 0.7. For other hyperparameters used in our experiments includes learning rate, early stopping patience, hidden layer size, dropout rates of input layer and hidden layer, we adopt the similar setting as Feng et al. (2020).

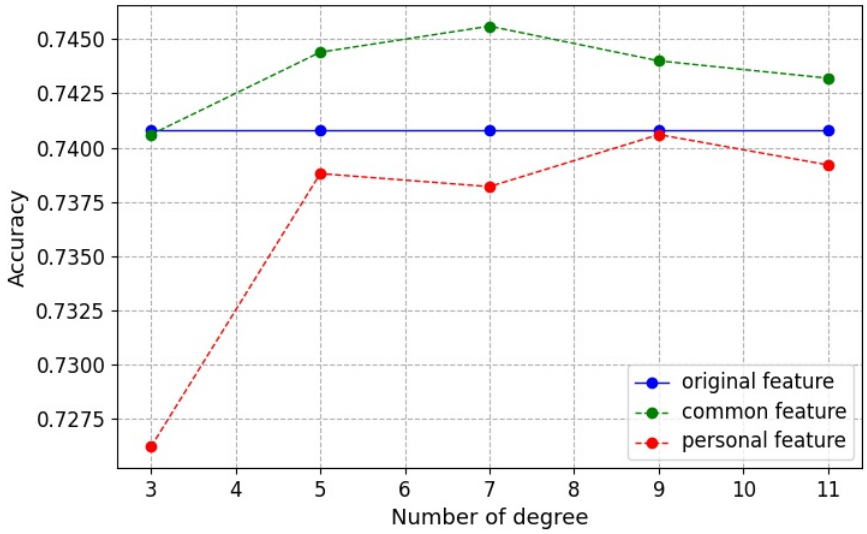



Figure 4: Dropping feature element according to the number of words.

Table 3: Classification accuracy (%) on four datasets

| Methods | Cora | Citeseer | pubmed | 20-newsgroup |
|---|---|---|---|---|
| **FAHGNN** | **86.1±0.3** | **75.7±0.1** | **83.0±0.4** | **79.9±1.0** |
| **FAHGNN/CR** | 84.3 ±0.2 | 73.5 ±0.3 | 81.9 ±0.3 | 77.9 ±0.6 |
| **FAHGNN/Drop** | 83.8 ±0.4 | 74.4 ±0.3 | 81.6 ±0.3 | 77.6 ±1.9 |

### A.3 Additional Experiments

**Ablation Study** In Table 1, we show the results of an ablation study to examine the contributions of different components of FAHGNN. First, all variants with some components removed show clear performance drops when compared to the full model, suggesting that each of the designed components contributes to the success of FAHGNN. Note that FAHGNN/CR denotes without consistency regularization, FAHGNN/CR denotes without DropFeature in propagation process, i.e., $S^N = 1$, $S^E = 1$.

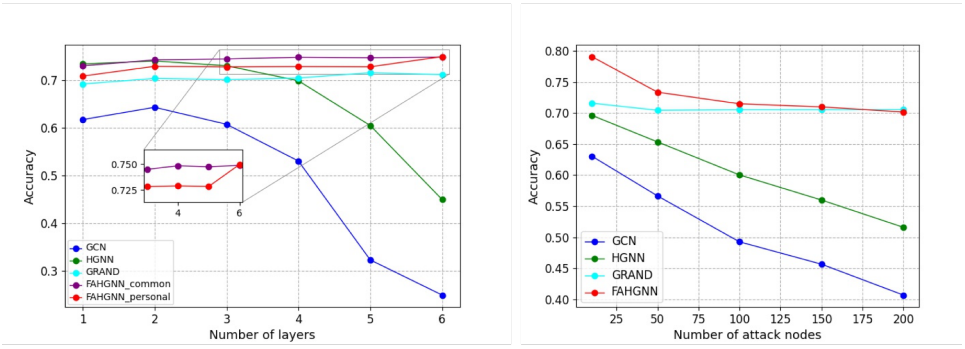
**Attack setting**

The vulnerability of exsiting GNNs models to graph attacks has been demonstrated (Zhu et al. (2019); Zügner et al. (2018)). As existing research mainly focuses on simple graphs, a study on hypergraphs is improtant. We analyse the robustness of FAHGNN by generating perturbed graphs and hypergraphs with a random attack method.

We randomly choose 10, 50, 100, 150, 200 test nodes as target nodes. Then, we constrain the attack by setting the perturbation. A limited number of perturbations are performed on the target nodes (similar to Zhang et al. (2019)) by $\Delta = d_{v_0} + 2$, where $v_0$ is the target node and $d_{v_0}$ is the degree of $v_0$. The random attack contains delete and add edges. Specifically, $(d_{v0} + 2)/2$ edges between $v_0$ and its neighbors are randomly removed and $(d_{v0} + 2)/2$ nodes that have different labels than the target node are chosen to add edges. Note that in a hypergraph, for each target node, we just remove and add this node in random $(d_{v0} + 2)/2$ hyperedges.

**Oversmoothing and Robustness analysis on Pubmed** Figure 5 illustrates the classification accuracies of different methods with respect to different numbers of attack nodes on the 20-newsgroup dataset. We see that FAHGNN can mitigate oversmoothing on 20-newsgroup, and the robustness of

FAHGNN is also clear. Note that 20-newsgroup just has 100 hyperedges, thus, attacking nodes on the hypergraph will lead to a huge perturbation compared to directly perturbing a simple graph.




(a) Over-Smoothing on 20-newsgroup

(b) Attack on 20-newsgroup

Figure 5: Over-Smoothing and robustness analysis on Pubmed.

# B PROOF

## B.1 PROOF OF POSITIVE SEMIDEFINITE

*Proof* Based on the defination of positive semidifinite of a matrix, we can get that $H^T H$ is positive semidefinite according to $\mathbf{X}^\top(\mathbf{H}^\top\mathbf{H})\mathbf{X} = (\mathbf{H}\mathbf{X})^\top(\mathbf{H}\mathbf{X}) = \|\mathbf{H}\mathbf{X}\|^2 \geq 0$. Therefore, $\mathbf{\Delta_E} = \mathbf{D}_e^{-1/2}\mathbf{H}^\top\mathbf{D}_v^{-1}\mathbf{H}\mathbf{D}_e^{-1/2}$ is also positive semidefinite.

## B.2 PROOF OF THEOREM 1

*Proof.* According to Proposition 1, we obtain $dim(\mathbf{X}^c) \leq dim(\mathbf{X}^p)$. In GCNs, $\hat{\mathbf{A}}$ is a reversible matrix, then $dim(\hat{\mathbf{A}}\mathbf{X}^c) \leq dim(\hat{\mathbf{A}}\mathbf{X}^p)$. As for hypergraph-based GNNs, it is also valid $dim(\mathbf{\Theta}\mathbf{X}^c) \leq dim(\mathbf{\Theta}\mathbf{X}^p)$. Therefore, stacking many network layers will lead to

$$dim\left(\mathbf{H}^{c(l)}\right) \leq dim\left(\mathbf{H}^{p(l)}\right), \forall l < L \tag{7}$$

before a certain layer $L$ makes $d_{\mathcal{M}}\left(\boldsymbol{H}^{(l)}\right) < \epsilon$. Here, $\mathbf{H}^{c(l)}$ and $\mathbf{H}^{p(l)}$ are the hidden common information representations and personal infromation representations in $l^{th}$ layer, respectively. Therefore, the propagation on common features more easily tends to subspace $\mathcal{M}$. □

## B.3 THE DETAILS OF THEOREM 2

Employing DropFeature as the perturbation method, the variance term $\mathrm{Var}_\epsilon\left(\mathbf{\Theta}\widetilde{\mathbf{X}}W\right) = \frac{\delta}{1-\delta}\sum_{j=0}^{n-1}\left(\mathbf{X}_j W\right)^2(\mathbf{\Theta})^2$ here, $\delta$ is the drop rate. We can obtain the corresponding regularization term as:

$$\mathcal{R}_{DF}^c(W) = \frac{\delta}{1-\delta}\sum_{j=0}^{n-1}\left[\left(\mathbf{X}_j W\right)^2\sum_{i=0}^{n-1}\left(\mathbf{\Theta}_{ij}\right)^2 z_i^2\left(1-z_i\right)^2\right] \tag{8}$$

Note that $z_i^2\left(1-z_i\right)^2$ (or $z_i\left(1-z_i\right)$) indicate the classification uncertainty for the node $i$, and it reaches a maximum at $z_i = 0.5$ and minimum at $z_i = 0$ or 1. The term $\sum_{i=0}^{n-1}\left(\mathbf{\Theta}_{ij}\right)^2 z_i^2\left(1-z_i\right)^2$ can be viewed as the weighted average classification uncertainty over $n$ neighborhoods of the node $j$ with the weights $\mathbf{\Theta}_{ij}^2$. Here, it is required that the node's neighborhoods have lower classification uncertainty scores. Hence, $\mathbf{X}_j \cdot W^2$ is an indicator of the classification confidence for the node

$j$. During optimization which aims to achieve a higher classification confidence for a node, the information propagation with the consistency regularization loss can enforce the consistency of the classification confidence between each node and its multi-hop neighborhoods.

*Proof* The expectation is:

$$\frac{1}{2}\sum_{i=0}^{n-1}\mathbb{E}\left[\left(\tilde{z}_i^{(1)}-\tilde{z}_i^{(2)}\right)^2\right]=\frac{1}{2}\sum_{i=0}^{n-1}\mathbb{E}\left[\left(\left(\tilde{z}_i^{(1)}-z_i\right)-\left(\tilde{z}_i^{(2)}-z_i\right)\right)^2\right] \tag{9}$$

For the term $\tilde{z}_i - z_i$, we appproximate it with its first-order Taylor expansion around

$$\begin{aligned}\frac{1}{2}\sum_{i=0}^{n-1}\mathbb{E}\left[\left(\tilde{z}_i^{(1)}-\tilde{z}_i^{(2)}\right)^2\right] &\approx \frac{1}{2}\sum_{i=0}^{n-1} z_i^2\left(1-z_i\right)^2\mathbb{E}\left[\left(\overline{\mathbf{A}}_i\left(\widetilde{\mathbf{X}}^{(1)}-\widetilde{\mathbf{X}}^{(2)}\right)\cdot\mathbf{W}\right)^2\right] \\ &= \sum_{i=0}^{n-1} z_i^2\left(1-z_i\right)^2\operatorname{Var}_\epsilon\left(\overline{\mathbf{A}}_i\widetilde{\mathbf{X}}\cdot\mathbf{W}\right)\end{aligned} \tag{10}$$

### B.4 The Detail of Theorem 3

With DropFeature as augmentation method, the regularization term can be expressed as:

$$\mathcal{R}_{DF}^q(W)=\frac{1}{2}\frac{\delta}{1-\delta}\sum_{j=0}^{n-1}\left[\left(\mathbf{X}_jW\right)^2\sum_{i=0}^{m-1}\left(\mathbf{\Theta}_{ij}\right)^2 z_i\left(1-z_i\right)\right] \tag{11}$$

The expectation of $\mathcal{L}_{sup}$

$$\frac{1}{2}\sum_{i=0}^{n-1}\mathbb{E}\left[\left(\tilde{z}_i^{(1)}-\tilde{z}_i^{(2)}\right)^2\right]=\frac{1}{2}\sum_{i=0}^{n-1}\mathbb{E}\left[\left(\left(\tilde{z}_i^{(1)}-z_i\right)-\left(\tilde{z}_i^{(2)}-z_i\right)\right)^2\right] \tag{12}$$

$$\mathcal{L}_{\text{org}}=\sum_{i=0}^{m-1}\left[-y_i\overline{\mathbf{A}}_i\mathbf{X}\cdot\mathbf{W}+\mathcal{A}\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)\right] \tag{13}$$

where $\mathcal{A}\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)=-\log\left(\frac{\exp\left(-\overline{\mathbf{A}}_i\mathbf{X}\cdot\mathbf{W}\right)}{1+\exp\left(-\overline{\mathbf{A}}_i\mathbf{X}\cdot\mathbf{W}\right)}\right)$. Then the expectation of perturbed classification loss can be rewritten as:

$$\mathbb{E}_\epsilon\left(\mathcal{L}_{\text{sup}}\right)=\mathcal{L}_{\text{org}}+\mathcal{R}(\mathbf{W}) \tag{14}$$

where $\mathcal{R}(\mathbf{W})=\sum_{i=0}^{m-1}\mathbb{E}_\epsilon\left[\mathcal{A}\left(\overline{\mathbf{A}}_i,\widetilde{\mathbf{X}}\right)-\mathcal{A}\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)\right]$ acts as a regularization term of $\mathbf{W}$. To demonstrate that, we can take a second-order Taylor expansion of $\mathcal{A}\left(\overline{\mathbf{A}}_i,\widetilde{\mathbf{X}}\right)$ around $\overline{\mathbf{A}}_i\mathbf{X}\cdot\mathbf{W}$

$$\mathbb{E}_\epsilon\left[\mathcal{A}\left(\overline{\mathbf{A}}_i,\widetilde{\mathbf{X}}\right)-\mathcal{A}\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)\right]\approx\frac{1}{2}\mathcal{A}''\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)\operatorname{Var}_\epsilon\left(\overline{\mathbf{A}}_i\widetilde{\mathbf{X}}\cdot\mathbf{W}\right) \tag{15}$$

We can easily check that $\mathcal{A}''\left(\overline{\mathbf{A}}_i,\mathbf{X}\right)=z_i\left(1-z_i\right)$. Applying this quadratic approximation to $\mathcal{R}(\mathbf{W})$, we get the quadratic approximation form of $\mathcal{R}(\mathbf{W})$:

$$\mathcal{R}(\mathbf{W})\approx\mathcal{R}^q(\mathbf{W})=\frac{1}{2}\sum_{i=0}^{m-1}z_i\left(1-z_i\right)\operatorname{Var}_\epsilon\left(\overline{\mathbf{A}}_i\widetilde{\mathbf{X}}\cdot\mathbf{W}\right) \tag{16}$$

□