NCGNN: Node-Level Capsule Graph Neural Network for Semisupervised Classification

Rui Yang, Wenrui Dai, Member, IEEE, Chenglin Li, Member, IEEE, Junni Zou, Member, IEEE, and Hongkai Xiong, Senior Member, IEEE

Abstract—Message passing has evolved as an effective tool for designing graph neural networks (GNNs). However, most existing methods for message passing simply sum or average all the neighboring features to update node representations. They are restricted by two problems: 1) lack of interpretability to identify node features significant to the prediction of GNNs and 2) feature overmixing that leads to the oversmoothing issue in capturing long-range dependencies and inability to handle graphs under heterophily or low homophily. In this article, we propose a node-level capsule graph neural network (NCGNN) to address these problems with an improved message passing scheme. Specifically, NCGNN represents nodes as groups of node-level capsules, in which each capsule extracts distinctive features of its corresponding node. For each node-level capsule, a novel dynamic routing procedure is developed to adaptively select appropriate capsules for aggregation from a subgraph identified by the designed graph filter. NCGNN aggregates only the advantageous capsules and restrains irrelevant messages to avoid overmixing features of interacting nodes. Therefore, it can relieve the oversmoothing issue and learn effective node representations over graphs with homophily or heterophily. Furthermore, our proposed message passing scheme is inherently interpretable and exempt from complex post hoc explanations, as the graph filter and the dynamic routing procedure identify a subset of node features that are most significant to the model prediction from the extracted subgraph. Extensive experiments on synthetic as well as real-world graphs demonstrate that NCGNN can well address the oversmoothing issue and produce better node representations for semisupervised node classification. It outperforms the state of the arts under both homophily and heterophily.

Index Terms—Capsule, graph neural networks (GNNs), homophily and heterophily, message passing, oversmoothing, semisupervised classification.

I. INTRODUCTION

Message passing has proved to be the cornerstone of numerous graph neural networks (GNNs) for analyzing data with irregular structures, including citation networks [1]–[3], point clouds [4], [5], chemical molecules [6], [7], and recommendation systems [8]. The essential idea is to update the representation of each node by aggregating features from its topological neighborhoods. Commencing with the great success of GCN [1] in semisupervised node classification, a myriad of studies have been developed to ameliorate the vanilla message passing scheme with attention mechanisms [3], [9], edge attributes [10], multirelational features [11], structural information [9], [12], [13], and so on.

Despite the promising performance in many graph-based machine learning tasks, most existing message passing schemes adopt the naive sum or average aggregator to fuse all the neighborhood information and are designed under the assumption of strong homophily, i.e., “birds of a feather flock together” [14]. Therefore, challenges have risen for GNNs to solve semisupervised node classification, as summarized in the following.

1) **Interpretability:** Summing up or averaging all the neighboring features makes GNNs lack human intelligible explanations. Since the most influential part of node features for the prediction of the model cannot be identified, complex post hoc methods would be required to explain GNNs in certain application domains (e.g., GNNExplainer [15] and XGNN [16]).

2) **Oversmoothing:** Excessive irrelevant messages (noise) might be aggregated when stacking multiple message passing layers to capture long-range (high-order) neighborhood information. Node features in different classes would be overmixed and indistinguishable, which is the oversmoothing issue in GNNs.

3) **Heterophily:** The naive average aggregator is essentially a special form of Laplacian smoothing [17] that forces proximal nodes to have similar features. This working mechanism is desirable for graphs under the assumption of strong homophily (e.g., citation networks), but fails to generalize to networks with heterophily or low homophily, where connected nodes are more likely from different communities (classes) and have dissimilar features (e.g., amino acids in protein structures [18]). Feature averaging over the heterophily-dominant neighborhoods would also cause feature overmixing that produces undesirable similar features for nodes in different classes.

Inspired by the interpretability and dynamic routing procedure of capsule networks [19]–[23], several capsule-related GNNs [24]–[29] have been developed in the graph domain.
However, they commonly focus on graph-level classification or flatten the capsules back to conventional scalar features during message passing. In this article, we propose a node-level capsule graph neural network (NCGNN) to address the aforementioned challenges with adaptive message passing between interacting nodes. The contributions of this article are summarized as follows.

1) We propose vectorized node-level capsule (a group of neurons) for feature aggregation to efficiently preserve the distinctive properties of nodes.

2) We develop a novel dynamic routing procedure that adaptively selects appropriate node-level capsules for aggregation to prevent feature overmixing and generate interpretable high-level capsules for each class.

3) We demonstrate in theory that the proposed message passing scheme is permutation invariant and improves the expressive power of vanilla message passing scheme.

To the best of our knowledge, NCGNN is the first GNN model that leverages inherently interpretable node-level capsule-based message passing to improve semi-supervised node classification performance. In the existing work, DisenGCN [27] is the only capsule-related GNN for node classification, where message passing is realized based on the multihead mechanism shown in Fig. 1(b). By comparison, NCGNN exploits dynamic routing between node-level capsules to adaptively optimize the selection of subset of features from partial graph nodes (subgraph) and produce better node representations, as shown in Fig. 1(c). With such ability to overcome feature overmixing in neighborhood feature aggregation, NCGNN goes beyond the homophily limitation and relieves the oversmoothing problem for learning from larger receptive field.

To be concrete, NCGNN first transforms each node from the scalar-based features to a group of primary node capsules (vectors). Node features are projected into different subspaces such that each capsule reflects the distinctive properties of a node. Subsequently, for each target node, we employ two graph filters to determine a subgraph of neighboring nodes for feature aggregation and propose a novel neighborhood-routing-by-agreement mechanism that adaptively selects advantageous node-level capsules to generate class node capsules for classification. Therefore, the node-level capsule-based message passing is inherently interpretable by identifying a subset of input node capsules that are most significant to the model prediction from the extracted subgraph.

For empirical evaluations, we conduct comprehensive experiments on 12 real-world graphs with homophily or heterophily and synthetic graphs where the homophily level can be controlled. Experimental results demonstrate that, contrary to existing message passing schemes that might suffer from the oversmoothing issue in extracting high-order neighborhood information and perform poorly under the heterophily setting, NCGNN achieves performance gains with the increasing orders (hops) of neighborhoods and yields the state-of-the-art performance under both homophily and heterophily. We further show that the dynamic routing procedure between node-level capsules indeed produces more separable node embeddings for nodes in different classes, which also verifies the effectiveness of NCGNN for tackling the oversmoothing problem.

The rest of this article is organized as follows. Section II reviews the background and related work. Section III elaborates the proposed NCGNN framework. Experimental results are provided in Section IV and conclusions are drawn in Section V.

II. BACKGROUND AND RELATED WORK

A. Message Passing-Based GNNs

The core of most GNNs is message passing, in which each node aggregates features from its topological neighborhoods to update its representation. Let us denote the learnable weights matrix as $W \in \mathbb{R}^{f \times f}$, the set of neighboring node indices for node $v_i$ as $\mathcal{N}(i)$, and the nonlinear activation function (e.g., ReLU) as $\sigma(\cdot)$. A basic message passing layer takes the following form:

$$h'_j = \sigma \left( \sum_{i \in \mathcal{N}(j) \cup \{j\}} a_{ij} W_{ij} h_i \right)$$

where $h_j \in \mathbb{R}^{f}$ is the input features of node $v_j$, $h'_j \in \mathbb{R}^{f}$ is the output features of node $v_j$, $W_{ij} \in \mathbb{R}^{f \times f}$ is the transformed neural message sent from node $v_i$, and $a_{ij}$ is the edge weight for the message passing from node $v_j$ to node $v_i$, which can be computed via different mechanisms [1], [3], [9], [13], [27].

Fig. 1 provides a brief comparison of different message passing schemes. Fig. 1(a) shows the vanilla sum aggregator in GCN [1], where each node sums up all the neighboring features with fixed edge weights to update its representation. Fig. 1(b) shows the multichannel attention utilized by models such as GAT [3] and DisenGCN [27]. Although the edge weights can be learned with diverse mechanisms to improve the model capacity, each channel is still a naive averaging operation that sums up all the neighboring features, which can be viewed as an ensemble of GNN models. Fig. 1(c) shows the dynamic routing procedure in our NCGNN, where each node-level capsule adaptively aggregates an appropriate subset of capsules and restrains irrelevant messages from neighboring graphs.
nodes to prevent feature overmixing, thereby generating better node representations for the downstream classification.

B. Homophily and Heterophily

Graph-structured data are ubiquitous in a diverse set of domains. In this article, we categorize graphs into assortative and disassortative ones according to the homophily/heterophily level of node class labels. As in [12] and [30], the metric to measure the homophily ratio of graph $G$ is defined as

$$H(G) = \frac{1}{|V|} \sum_{v_i \in V} \sum_{j \in N(v_i)} \mathbb{I}(y_i = y_j) / |N(i)|$$

where $V$ is the node set, $y_i$ and $y_j$ denote the labels of nodes $v_i$ and $v_j$, respectively, and $\mathbb{I}(x) = 1$ if $x$ is true and 0 otherwise.

Assortative graphs are those with high homophily ratios (i.e., $H(G)$ is closer to one) such as citation networks and social networks. On the other hand, disassortative graphs exhibit heterophily or low homophily property (i.e., $H(G)$ is closer to zero). Connections of different types of amino acids in protein structures and web-page linking networks are two typical examples of disassortative graphs.

Many existing message passing schemes described by (1) rely on the strong homophily assumption and most common benchmark datasets in the task of node classification adhere to this principle (e.g., Cora [31], Citeseer [31], and Pubmed [32]). In this scenario, feature averaging over each node’s neighborhood would produce similar representations for nodes in the same class, which makes the subsequent classification task much easier. However, they fail to generalize to the heterophily setting and are even worse than the models that ignore the graph structure [e.g., multilayer perceptron (MLP)] [30] since summing up all the neighboring features from different communities would aggregate too much noise, thus overmixing and generating inseparable features for nodes in different classes. Several works have been proposed to address this limitation. Geom-GCN [12] builds structural neighborhoods in a latent space and proposes a novel bilevel aggregation scheme based on the observation that nodes of the same class in some disassortative graphs exhibit high structural similarity [33]. H$_2$GCN [30] identifies a set of key designs that improves learning from disassortative graphs: ego- and neighbor-embedding separation, higher order neighborhoods, and combination of intermediate representations. Different from these methods, we propose to use dynamic routing between node-level capsules to adaptively aggregate a subset of appropriate capsules from a selected subgraph. Thus, irrelevant messages can be effectively restrained for learning under both homophily and heterophily.

C. Oversmoothing in GNNs

As illustrated in [34], the oversmoothing issue is caused by the feature overmixing of nodes in different classes when enlarging the receptive field to capture high-order (multihop) neighborhood information. To address this problem, numerous works have been proposed in recent years. JK-Net [35] and DNA [36] utilize skip connection and attention mechanism for preserving the information in each message passing step. PPNP and APPNP [37] consider graph diffusion to replace the adjacency matrix with personalized PageRank matrix that takes the root node or initial node features into account. In [38], GCNs are armed with residual/dense connections. GCNII [39] modifies GCN with initial residual and identity mapping. DAGNN [40] decouples feature transformation and propagation and proposes a novel adjustment mechanism to adaptively aggregate features of different propagation hops. Scattering GCN [41] uses geometric scattering transform to enable bandpass filtering over the graph signal. PairNorm [42] and differentiable group normalization (DGN) [43] are two normalization schemes proposed for GNNs with deeper layers. Besides, some tricks have been studied. DropEdge [44] randomly removes a certain fraction of edges in each training iteration to reduce message passing. AdaEdge [34] constantly adjusts the graph topology (adding intraclass edges and removing interclass edges) according to the predictions made by GNNs in a self-training-like fashion. BBGDC [45] further generalizes DropEdge and Dropout [46] by adaptive connection sampling. In comparison, NCGNN relieves the oversmoothing issue by directly restraining noisy messages passing between interacting nodes to prevent feature overmixing.

D. Interpretability Methods for GNNs

Designing explainable approaches for GNNs is an emerging research field. Most of the existing interpretability methods adopt the post hoc explanation for a trained GNN. In [47], several explanation techniques that are originally used for standard neural networks are applied to GNNs, such as sensitivity analysis [48], guided backpropagation [49], and layerwise relevance propagation [50]. Gradient-based saliency maps [51], class activation mapping [52], and excitation backpropagation [53] are extended from conventional convolutional neural networks (CNNs) to GNNs in [54]. GNNExplainer [15] generates explanation by identifying a subgraph and a subset of node features that are crucial for the model’s prediction of each instance. PGM-Explainer [55] further resorts to a probabilistic graphical model to capture the dependencies among explained features. PGEExplainer [56] uses a parameterized neural network to identify the important subgraphs at the model level. XGNN [16] and RG-Explainer [57] generate an explanatory graph with reinforcement learning at the model level and instance level, respectively. GNN-LRP [58] identifies relevant walks in the input graph to interpret GNN’s prediction. More recently, SubgraphX [59] explores significant subgraphs for explaining GNNs with Monte Carlo tree search. Gem [60], CF-GNNEExplainer [61], and RCEXplainer [62] provide explanations through causal inference. ReFine [63] approaches multigrained interpretability that provides both model- and instance-level explanations. Also, ie-HGCN [64] focuses on interpreting heterogeneous-information-network-oriented GCNs via automatically learning useful meta-paths. Contrary to most of the above interpretability methods that provide a post hoc explanatory subgraph at the structure level (e.g., PGEExplainer [56], SubgraphX [59], and Gem [60]), we design an inherently interpretable message passing scheme...
for homogeneous graphs at the feature level by identifying important input node capsules from the extracted subgraph.

### E. Capsule-Related Networks

The concept of capsule was first put forward in [19], and recently, it has achieved enormous advances (e.g., [20]–[23], [65]). The key is to replace the scalar-based features in conventional CNNs with vector-based capsules to extract high-quality image representations. In CapsNet [20], each capsule is a group of neurons whose overall length represents the existence of an entity and its orientation encapsulates the instantiation parameters. The capsules in the last layer are called class capsules, and the length of each class capsule’s output vector indicates the predicted probability of each class. The advantage is that it utilizes dynamic routing to determine the coupling degree between every two capsules in successive layers, thereby benefiting the information propagation in the network to select appropriate capsules for generating higher level representations. Capsule-related networks have good nature of interpretability since the instantiation parameters encode various types of properties and the capsule selection (CS) mechanism of dynamic routing can identify crucial features for the network output, which helps explain the model prediction.

Capsules have been extended to the graph domain. GCAPS-CNN [24] concatenates higher order statistical moments of each node’s neighboring features as its capsules. However, capsules are flattened back to scalar features during message passing. In [25], explicit kernel graph representations and CapsNet are combined to perform classification. CapsGNN [26] employs a GNN to produce graph-level representations and utilizes dynamic routing on top of the GNN to benefit graph embeddings. HG-CN [29] further utilizes capsule to characterize the part–whole relationship for learning hierarchical graph representations. All of the above methods focus on learning more effective graph-level embeddings for graph classification, while our work is proposed for semisupervised node classification. As for node-level learning, DisenGCN [27] views the disentangled node representations as capsules. However, its message passing scheme is essentially a multichannel ensemble, as shown in Fig. 1(b). There is no dynamic routing between different channels (capsules) and capsules are also flattened back to scalar-based features such as GCAPS-CNN. By contrast, NCGNN leverages dynamic routing to adaptively aggregate appropriate node-level capsules and restrain harmful noise from higher order neighborhoods for each target node. Caps2NE [28] treats each node as one capsule for unsupervised node representation learning, while NCGNN treats each node as a group of capsules for semisupervised node classification.

### III. NODE-LEVEL CAPSULE GRAPH NEURAL NETWORK

In this section, we elaborate the proposed NCGNN for semisupervised node classification. NCGNN has good inherent interpretability with an adaptive message passing scheme that represents each node as a group of capsules and leverages dynamic routing between capsules. NCGNN is capable of adaptively identifying a subset of crucial node-level capsules during the message passing procedure. Therefore, NCGNN can help relieve the oversmoothing issue and learn effectively under both homophily and heterophily by avoiding feature overmixing. Fig. 2 shows the workflow of NCGNN.

#### A. Problem Formulation

We focus on the task of graph-based semisupervised node classification. An undirected graph with \( N \) nodes is represented as a tuple \( G = (\mathcal{V}, A) \), where \( \mathcal{V} = \{v_i\}_{i=1}^{N} \) is the node set and \( A \in \mathbb{R}^{N \times N} \) is the adjacency matrix where each entry \( a_{ij} \) is set to 1 for the existence of edge linking nodes \( v_i \) and \( v_j \) and 0 otherwise. The normalized graph Laplacian is defined as \( L = I_N - \text{D}^{-1/2} \text{AD}^{-1/2} \), where \( I_N \) is the identity matrix and \( \text{D} = \text{diag}(d_1, \ldots, d_N) \) is the degree matrix with \( d_i \) denoting the degree of node \( v_i \). The eigendecomposition of \( L \) gives us \( L = U \Lambda U^T \). Here, \( \Lambda \) is a diagonal matrix of the eigenvalues of \( L \), and the columns of \( U \) are the orthonormal eigenvectors termed graph Fourier basis. According to the spectral graph theory, a filter on \( G \) can be defined as a parameterized polynomial function \( g_{\theta}(L) \) of \( L \), and a graph signal \( x \in \mathbb{R}^N \) filtered by \( g_{\theta} \) can be written as \( y = g_{\theta}(L)x = Ug_{\theta}(\Lambda)U^Tx \in \mathbb{R}^N \). We also have a feature matrix \( X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times N} \) with each row corresponding to the feature vector of an individual node. Given a subset of graph nodes \( \mathcal{V} \subset \mathcal{V} \) with labels, the task of semisupervised node classification is to predict the classes of unlabeled nodes \( \mathcal{V}' \subset \mathcal{V} \) based on \( G \) and \( X \).

#### B. Motivations

In most existing GNNs [1], [3], [66], message passing is realized by summing up or averaging the node features within a neighborhood. They cannot distinguish the nodes and their features that would lead to feature overmixing in certain scenarios such as heterophily and larger receptive field and, consequently, harm the downstream node classification. Therefore, we design a message passing scheme for NCGNN to address two enlightening questions as follows.

1) **Q1:** For each target node, how to identify the graph nodes (a subgraph) whose information should be aggregated?

2) **Q2:** For each target node-level capsule, how to determine and aggregate an appropriate subset of capsules that contains little noise from the subgraph?

First, we address Q1 by selecting a subgraph for each target node with two graph filters designed in the attention- and diffusion-based manner. Contrary to most existing GNNs (e.g., GCN [1], MoNet [66], and GAT [3]) that only aggregate one-hop neighborhood information in each message passing step and stack multiple layers with entangled feature transformation and propagation to enlarge the receptive field, the designed graph filters can adapt to higher order neighborhoods to directly capture long-range dependencies without increasing the model complexity. Therefore, NCGNN can reduce the risk of oversmoothing by decoupling feature aggregation and transformation [40] and enables explanations from both spatial and spectral aspects.

Moreover, we solve Q2 by designing a capsule graph layer where message passing is improved with a novel
producing better node representations. We elaborate the mechanisms of the designed graph filter and capsule graph layer to solve questions Q1 and Q2 for producing better node representations.

**D. Graph Filter**

For each target node, the graph filter identifies a subgraph where information should be aggregated and determines the influence of each subgraph node on it. In the vanilla GCN [1], the graph filter is defined as the symmetric normalized transition matrix $\tilde{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, where $\tilde{A} = A + I_N$ and $\tilde{D} = \text{diag}(\text{tr}_\text{in} A)$. This graph filter and its variants [3], [9] are widely used in existing GNNs where only the one-hop neighborhood features are aggregated. However, higher order neighborhood information is also useful for generating node representations, as studied in [37], [68], and [69]. To extract long-range dependencies, we define the generalized graph filter as in (4) with two specific forms in the attention- and diffusion-based manner

$$\hat{A} = f_\xi(\tilde{A}) = \sum_{i\in\mathcal{M}, \mathcal{M}\subseteq\mathcal{N}} \zeta_i \hat{A}^i.$$  

Here, $\mathcal{M}$ is the set of multiple orders (hops) (e.g., $\mathcal{M} = \{1, 2, 4, 5\}$) and $f_\xi$ is a polynomial function parameterized by coefficients $\zeta_i$ ($i \in \mathcal{M}$). Considering that the exponentiation operation $\hat{A}^i$ may result in a dense matrix even for a small $i$ due to the small-world property [70] of real-world graphs, we maintain the sparsity property by using either an $\epsilon$-threshold or top-$k$ sparsification as in [71], i.e., entries of $\hat{A}^i$ that are smaller than a threshold or are not the top-$k$ values in each row are set to 0, and the symmetric normalization is recalculated for the sparsified version of $\hat{A}^i$ subsequently. Then, each row of $\hat{A}$ identifies part of the $N$ nodes (a subgraph) for the corresponding node to aggregate information, and each nonzero entry denotes the influence of the message passing between two interacting nodes.

1) **Attention-Based Graph Filter:** The attention-based graph filter is flexible and learnable, i.e., there is no constraint on $\mathcal{M}$ and the coefficients are data-driven. Given $\mathcal{M}$ and $\hat{A}^i$ of distance $i$ ($i \in \mathcal{M}$), the corresponding coefficient satisfies

$$\zeta_i = \frac{\exp(\zeta_i)}{\sum_{j\in\mathcal{M}} \exp(\zeta_j)} \quad \forall i \in \mathcal{M}$$

which is the common softmax operation, and $\zeta_j$ ($j \in \mathcal{M}$) are free parameters that can be initialized as all zeros or appropriate priors and directly learned via backpropagation. Thus, the coefficients of $f_\xi$ can be interpreted as the attention scores assigned to neighborhoods of different hops. Note that

**Fig. 2.** Workflow of NCGNN: NCGNN first transforms each node from the scalar-based features to $K$ primary node capsules. Subsequently, the designed graph filter selects a subgraph for each target node to perform message passing and dynamic routing is leveraged to generate $C$ class node capsules for it, where $C$ is the number of classes. In the capsule graph layer, each node-level capsule adaptively selects appropriate capsules to update its representation via dynamic routing and irrelevant messages from neighborhoods are restrained to avoid feature overmixing. For a better view, we only plot the message passing procedure for generating the class node capsules of the brown node within its ego network of subgraph. Note that capsule graph layers can be further stacked if necessary, though we only use one capsule graph layer to produce $C$ class node capsules for each target node.
although this attention mechanism is not node-adaptive (i.e., shared across graph nodes), this weight sharing fashion helps prevent overfitting and performs well on all the benchmark datasets.

2) Diffusion-Based Graph Filter: In recent works [37], [71], [72], graph diffusion has demonstrated promising performance by propagating features using personalized PageRank (PPR) and heat kernel (HK). In this work, we evaluate the performance of PPR matrix as the fixed diffusion-based graph filter (HK is omitted due to comparable performance) in NCGNN

$$\tilde{A} = a(I_N - (1 - a)\tilde{A})^{-1} = \sum_{i=0}^{\infty} a(1 - a)^i \tilde{A}^i$$ (6)

where $a \in (0, 1)$ is termed teleport probability, $M = \mathbb{N}$, and $\tilde{A} = a(1 - a)I$. When tackling large-scale graphs for which it is inefficient to compute matrix inversion, we use the truncated form as in (7) by restricting the sum to a finite number $P$, similar to [37]. We also apply the sparsification operation to the fixed diffusion-based graph filter to identify a subset of important graph nodes for feature aggregation

$$\tilde{A} \approx \sum_{i=0}^{P} a(1 - a)^i \tilde{A}^i.$$ (7)

3) Discussion: The designed graph filter in (4) can be explained from both spatial and spectral aspects. In the spatial domain, each entry $\tilde{a}_{ij}$ of $\tilde{A}$ represents the influence of the message passing from node $v_i$ to node $v_j$. In the spectral domain, the following proposition indicates the relationship between $\tilde{A}$ and the polynomial filter $g_0(L)$.

Proposition 1: Consider the graph $G$ with added self-loops and choose the graph Laplacian as $L = I_N - \tilde{A}$. The graph filter defined by (4) can be expressed as a J-order polynomial filter $g_0(L) = \sum_{i=0}^{J} \theta_i L^i$ with $J = \max(M)$ in the spectral domain.

Proof: Rewrite $\tilde{A}$ as $f_\tilde{A}(\tilde{A}) = \sum_{i=0}^{I} \tilde{\chi}_i \tilde{A}^i$ with $I = \max(M)$ and $\tilde{\chi}_i \neq 0$ for $i \notin M$. The equivalence between $f_\tilde{A}(\tilde{A})$ and $g_0(L)$, i.e., $\sum_{i=0}^{J} \tilde{\chi}_i \tilde{A}^i = \sum_{i=0}^{I} \theta_i L^i$, can be constructed via $\tilde{\chi}_i = \sum_{i=0}^{I} \theta_i L^i$ and $J = I$ [71].

$\tilde{A}$

E. Capsule Graph Layer

After obtaining the subgraph nodes $\{v_j | \tilde{a}_{ij} \neq 0\}$ identified by the graph filter $\tilde{A}$ and a set of primary node capsules $S = \{h_k^{(i)} | \tilde{a}_{ij} \neq 0, k = 1, \ldots, K\}$ for each target node $v_i \in \mathcal{V}$, this section elucidates how the designed capsule graph layer transforms node $v_i$ from $K$ primary node capsules to $C$ class node capsules $T = \{u_l^{(i)} | l = 1, \ldots, C\}$ for classification with adaptive feature aggregation from $S$ to $T$. Note that although capsule graph layers can be further stacked to generate multiple high-level capsules in intermediate layers if necessary, we use one capsule graph layer in our study to directly produce class node capsules attributed to its efficiency, scalability, and remarkable performance. More discussions are provided in Section III-F.

Denote with $f_c$ the dimension of class node capsules, $W_{kl} \in \mathbb{R}^{f_c \times f_p}$ the learnable weights matrix between the $k$th primary node capsules and the $l$th class node capsules, and $b_l \in \mathbb{R}^{f_c}$ the learnable bias for generating the $l$th class node capsules. The feature generation rule for the $l$th class node capsule $u_l^{(i)} (l = 1, \ldots, C)$ of node $v_i$ consists of CS and class message passing (CMP)

$$p_j^{(i)} = \sum_{k=1}^{K} c_{jkl} W_{kl} h_k^{(i)} \forall j \in \{j | \tilde{a}_{ij} \neq 0\} \text{ (CS)}$$ \hspace{1cm} (8)

$$u_l^{(i)} = \sum_{j \in \{j | \tilde{a}_{ij} \neq 0\}} \tilde{a}_{ij} p_j^{(i)} + b_l \text{ (CMP)}$$ \hspace{1cm} (9)

where $c_{jkl}$ is the coupling coefficient between $h_k^{(i)}$ and $p_j^{(i)}$, an intermediate capsule that represents the properties of class $l$ extracted from node $v_j$ and is used for generating the $l$th class node capsules via CMP in (9). As per dynamic routing, $c_{jkl}$ (k = 1, \ldots, K and l = 1, \ldots, C) are computed via an iterative neighborhood-routing-by-agreement procedure to adaptively select an appropriate subset of primary node capsules for aggregation from node $v_j$, which will be discussed later. $\tilde{a}_{ij}$ is computed by the graph filter $\tilde{A}$, determining the influence of the message passing from node $v_j$ to node $v_i$. As can be seen in (8) and (9), $u_l^{(i)}$ only aggregates the primary node capsules that are in agreement (i.e., larger $c_{jkl}$) to class $l$ from $S$, while harmful noise is adaptively restrained (i.e., smaller $c_{jkl}$) to relieve the risk of feature overmixing.

As in CapsNet [20], we want the length of the output vector of each class node capsule $u_l^{(i)}$ to indicate the presence of an instance of class $l$. Hence, we use the “squashing” activation function as in (10) to ensure that the length of the output vector of $u_l^{(i)}$ lies between zero and one, representing the probability that node $v_i$ belongs to class $l$. We also use the separate margin loss as in (11) for each class node capsule

$$v_l^{(i)} = \text{squash}(u_l^{(i)}) = \frac{\|u_l^{(i)}\|^2}{1 + \|u_l^{(i)}\|^2} \cdot u_l^{(i)}$$ \hspace{1cm} (10)

$$L_l^{(i)} = T_l^{(i)} \cdot \max(0, m + \|v_l^{(i)}\|^2) + \lambda \cdot \left(1 - T_l^{(i)}\right) \cdot \max(0, \|v_l^{(i)}\|^2 - m)^2$$ \hspace{1cm} (11)

where $L_l^{(i)}$ is the loss for the $l$th class node capsule of node $v_i$, $v_l^{(i)}$ is the output vector of capsule $u_l^{(i)}$, $T_l^{(i)} = 1$ if node $v_i$ belongs to class $l$ and 0 otherwise, $m^+$ is a constant close to 1, $m^−$ is close to 0, and $\lambda$ is a parameter for preventing all class node capsules from shrinking to 0 at the beginning of training. The total loss is $L = (1/|\mathcal{V}|) \sum_{v_i \in \mathcal{V}} \sum_{l=1}^{K} L_l^{(i)}$.

Neighborhood-Routing-by-Agreement: While in CapsNet and CapsGNN, the coupling coefficients between a capsule and all the capsules in the layer above sum to 1, which implies that each capsule gets sent to an appropriate parent in the next layer, our neighborhood-routing-by-agreement results from different motivations. In our NCGNN framework, the coupling coefficients between a class node capsule and the primary node capsules of each individual node in the subgraph sum to 1, i.e., $\sum_{k=1}^{K} c_{jkl} = 1$. This means that for each subgraph node, some of its primary node capsules are in agreement, i.e., to be aggregated, while some are considered as noise to be restrained. The agreement is in the form of dot product of two vectors, and $c_{jkl}$ is iteratively adjusted according to (12)
and (13) for routing iterations \( t = 0, \ldots, T - 1 \), starting by initializing \( c_{jkl}^{(0)} = \exp(b_{jkl}^{(0)}) / \sum_{m=1}^{K} \exp(b_{jml}^{(0)}) \) with \( b_{jkl}^{(0)} \) \((k = 1, \ldots, K, l = 1, \ldots, C)\) of all zeros or set as some other appropriate priors

\[
c_{jkl}^{(t)} = \frac{\exp(b_{jkl}^{(t)})}{\sum_{m=1}^{K} \exp(b_{jml}^{(t)})}
\]

and

\[
b_{jkl}^{(t+1)} = b_{jkl}^{(t)} + (v_{j}^{(t)})^T (W_{jkl} h_{j}^{(k)})
\]

where \( v_{j}^{(t)} \) is computed in parallel with \( v_{j}^{(t)} \). As can be seen, if the output of the class node capsule \( v_{j}^{(t)} \) has a large dot product with \( W_{jkl} h_{j}^{(k)} \), the corresponding coefficient \( c_{jkl}^{(t)} \) of node \( v_{j} \) associated with \( h_{j}^{(k)} \) and class \( l \) would be iteratively increased. In this way, for the generation of each class node capsule, the neighborhood-routing-by-agreement adaptively identifies a subset of capsules (with larger \( c_{jkl}^{(t)} \)) for aggregation from each subgraph node, which has the following two advantages: 1) feature overmixing of interacting nodes can be suppressed to some extent, as irrelevant messages from the subgraph are restrained, which helps relieve over-smoothing in larger receptive field and learn effectively over disassortative graphs, and 2) the selected primary node capsules that are in agreement with large coupling coefficients can be used to explain the model prediction as they are most influential for generating the corresponding class node capsules in label prediction. The detailed interpretability analysis is discussed in Section IV-D3.

The procedure of the capsule graph layer in nodewise fashion is summarized in Algorithm 1, which consists of only differentiable operations and can be optimized via gradient-based backpropagation.

F. Analysis

1) Complexity Analysis: Let \( N \) denote the number of graph nodes needed for computation, \( L \) be the number of layers of GCN [1], \( T \) be the routing iterations of NCGNN, and \( D \) be the dimension of node representations. For simplicity, we ignore the normalization step in (3), the activation function, the bias terms in (3) and (9), and the softmax function of (12). We also assume that \( D = K^{T} f_{p} = C f_{c} \) is fixed for all layers. The time complexity of GCN is \( O(LN D^2 + L|A||D|) \), and the time complexity of NCGNN is about \( O((T + 1) ND^2 + T|A||D| + 2TNKD) \). Although our graph filter \( \tilde{A} \) contains more higher order neighborhood information than the first-order normalized adjacency matrix \( \tilde{A} \) utilized in GCN, \( \tilde{A} \) is processed with \( \epsilon \)-threshold or top-\( k \) sparsification, as illustrated above. Thus, the time complexity of our NCGNN framework is still comparable with GCN. Moreover, NCGNN shows scalability in a distributed manner [72] for processing large graphs, as it only has one capsule graph layer in our work to directly incorporate multihop neighborhood features, which is exempt from the exponential neighborhood expansion issue with the increasing message passing layers, i.e., to generate the class node capsules for each target node, we only need to load the features of its neighboring nodes identified by the sparsified graph filter \( \tilde{A} \).

2) Permutation Invariance: Permutation invariance or equivariance is essential for designing aggregators in GNNs. In this section, we prove by induction that our proposed capsule graph layer described in Algorithm 1 is permutation invariant to the order of graph nodes.

Lemma 1: Consider the input set \( Z = \{z_1, \ldots, z_M\} \), \( z_m \in \mathcal{Z} \), i.e., the input domain is the power set \( 2^\mathcal{Z} \).

If a function \( f_1 : 2^\mathcal{Z} \rightarrow \mathbb{R}^n \) operating on \( Z \) can be decomposed in the form of \( \rho(\sum_{z_m \in \mathcal{Z}} \phi_1(f_2(Z), \phi_1(z_m))) \) for suitable transformations \( \phi_1, \phi_2 \) and \( \rho \) and another permutation invariant function \( f_2 : 2^\mathcal{Z} \rightarrow \mathbb{R}^n \) operating on \( Z \), then \( f_1 \) is permutation invariant to the order of elements in \( Z \).

Proof: Permutation invariance implies that the elements of a set have no particular order, and consequently, any function on the set must be constant to arbitrary ordering of its elements. Since \( f_2 \) is permutation invariant and addition is commutative, the function \( \rho(\sum_{z_m \in \mathcal{Z}} \phi_2(f_2(Z), \phi_1(z_m))) \) satisfies permutation invariance.

Proposition 2: Given the graph \( G \) and the graph filter \( \tilde{A} \), the message passing scheme described in Algorithm 1 is permutation invariant for arbitrary routing iteration.

Proof: Let us define \( h = PNC(x) \) and \( p = CS(c, h) \) according to (3) and (8), respectively. Here, \( x \) is the given node features and \( c \) is the vectorized coupling coefficients. Note that the superscripts are omitted for simplicity. The input set is defined as \( Z = \{x_j | v_j \in \mathcal{V}\} \).

From Lemma 1, \( v_{j}^{(t)} \) is permutation invariant to the ordering of elements in \( Z \) at routing iteration \( t = 0 \), as

\[
v_{j}^{(t)} = \rho(\sum_{z_m \in \mathcal{Z}} \phi_2(f_2(Z), \phi_1(z_m))) \quad \text{with} \quad \phi_1(x_j) = PNC(x_j), \quad f_2(Z) = e^{(0)}, \quad \phi_2(f_2(Z), \phi_1(x_j)) = \tilde{a}_{j} \cdot CS(f_2(Z), \phi_1(x_j)), \quad \rho(\square) = \text{squash}(\square + b_1).
\]

Here, \( e^{(0)} \) is initialized as constant values, and thus, \( f_2 \) is permutation invariant.

Induction Step: Assume that for the \( t \)th routing iteration, the message passing scheme of the capsule graph layer is permutation invariant. We consider the \((t+1)\)th routing iteration. Since \( v_{j}^{(t)} \) \((u_j \in \mathcal{V}) \) computed in the \( t \)th routing iteration is permutation invariant, \( v_{j}^{(t+1)} \) computed via (13) are...
constant for any permutation. Thus, the coupling coefficients $c^{(t+1)}$ are permutation invariant. Let $f_2(Z) = e^{f_2(t+1)}$, and consequently, $h^{(t)}$ is permutation invariant at the end of the $(t+1)$th routing iteration in the way similar to $t = 0$. Therefore, the induction hypothesis is verified for the $(t+1)$th routing iteration. As a result, we conclude that Algorithm 1 is permutation invariant for any routing iteration.

3) Expressive Power: We analyze the expressive power of our method by comparing with the naive sum aggregator.

**Proposition 3:** The message passing scheme of capsule graph layer is more powerful than the naive sum aggregator, without nonlinear activation function.

**Proof:** Consider the following naive sum aggregator without nonlinear activation function:

$$h'_i = \sum_{j \in \{i | \overline{a}_{ij} \neq 0\}} \overline{a}_{ij} W h_j \quad \forall v_i \in \mathcal{V}$$

(14)

where $h_i \in \mathbb{R}^l$, and $h'_i \in \mathbb{R}^l$ are input and output node features, respectively, and $W \in \mathbb{R}^{h \times l}$ is the matrix of learnable weights for feature transformation.

Let us denote by $K$ and $C$ the numbers of each node’s input capsules and output capsules, respectively. For arbitrary $v_j \in \mathcal{V}$, $k = 1, \ldots, K$ and $l = 1, \ldots, C$, we suppose that $h_j^{(k)} = h_j$ for the $k$th input capsule of $v_j$, $W_{kj} = W$, $b_k$ are all constant zeros, and $b_{jkl}^{(0)} = 0$. Thus, $c_{jkl}^{(0)} = 1/K$. For the first routing iteration, the output capsules of each node can be generated by

$$p_j^{(l)} = \sum_{k=1}^{K} c_{jkl}^{(0)} W_{kj} h_j^{(k)} = W h_j$$

(15)

$$u_j^{(l)} = \sum_{j \in \{i | \overline{a}_{ij} \neq 0\}} \overline{a}_{ij} p_j^{(l)} = h_j$$

(16)

Equation (16) implies that, for each node, its output capsules are all equal to its output node features shown in (14). Thus, $b_{jkl}^{(l)}$ of each node are equal according to (13), and consequently, $c_{jkl}^{(1)} = 1/K$.

We can easily prove by induction that each node’s output capsules are $C$ repetitions of its output node features for any routing iteration. Under this specific condition, the naive sum aggregator is a special case of our capsule graph layer.

**IV. EXPERIMENTS**

We evaluate NCGNN on the task of semisupervised node classification under various scenarios. Extensive experiments on 12 real-world graph datasets as well as synthetic graphs show that NCGNN outperforms a variety of state-of-the-art methods under both homophily and heterophily and well addresses the oversmoothing problem in GNNs. We also provide visualization results to demonstrate the inherent interpretability of our proposed message passing scheme.

**A. Datasets**

1) **Real-World Graphs:** Twelve real-world graph benchmarks are considered in the experiments, including six assortative graphs and six disassortative graphs.

2) **Disassortative Graphs:** First, Squirrel and Chameleon are two subgraphs of Wikipedia networks [76], in which nodes represent web pages, edges indicate mutual links between pages, node features are bag-of-words representations corresponding to some informative nouns in the Wikipedia pages, and graph nodes are classified into five categories.

3) **Cora:** The Cora dataset is a citation network, where nodes represent documents, edges denote citations, and node labels are bag-of-words encoded product reviews, and class labels are given by the product category. For each dataset, we use 20 labeled nodes per class for training, 300 nodes for validation, and the rest is set for test. For a fairer comparison, we select five random train/validation/test splits for each dataset and run ten turns with different weight initialization seeds in each split, as suggested in [34]. Then, we measure the average test accuracy of all the 50 runs’ results. We further take Amazon Computers as an example dataset to evaluate the model performance under different training set sizes.

Table I summarizes the statistics of these real-world benchmark datasets.
validation, and test, respectively. We also measure the average test accuracy over ten random weight initializations for each graph.

B. Experimental Setup

1) Baselines: For node classification on assortative graphs, we compare with the following baselines: 1) three representative GNNs: GCN [1], GAT [3], and SGC [78]; 2) GNN with graph diffusion-based (i.e., PPR) message passing: APPNP [37] (PPNP [37] is omitted due to comparability with APPNP); 3) four recently proposed GNNs for tackling oversmoothing: JK-Net [35], DAGNN [40], GCNII [39], and Scattering GCN [41]; and 4) some other tricks with GCN as the backbone for relieving the oversmoothing issue: AdaEdge [34], DropEdge [44], and BBGDC [45]. For learning over disassortative graphs, we consider the following baselines: 1) the simplest deep learning model that ignores the graph structure: MLP; 2) two popular GNNs: GCN and GAT; 3) two strong baselines in solving semisupervised node classification for assortative graphs: DAGNN and GCNII; 4) Geom-GCN with different embedding methods [12]: Geom-GCN-I, Geom-GCN-P, and Geom-GCN-S; and 5) two variants of H2GCN [30] which achieve the state-of-the-art performance under heterophily: H2GCN-1 (one round for neighborhood aggregation) and H2GCN-2 (two rounds for neighborhood aggregation). Note that DisenGCN [27] leverages a message passing scheme similar to the multihread mechanism adopted in GAT, as discussed in Section II-A. Since the source code of DisenGCN is not publicly available and its reported performance is inferior to GCNII and hard to be reproduced, we do not include it as a baseline in our evaluations.

2) Model Configurations: The dimension of class node capsules ($f_v$) is set to 16. Dropout of $\rho = 0.9$ is applied to the primary node capsules. We use the Adam [79] optimizer to minimize the error function with a learning rate of $1e-3$. Training epochs are manually set for each dataset. $\ell_2$ regularization on the learnable weights is employed with weight decay of $1e-3$ for the six real-world disassortative graphs and $5e-3$ for the others. $\mathcal{M} = \{1, \ldots, \max(\mathcal{M})\}$ or $\mathcal{M} = \{0, \ldots, \max(\mathcal{M})\}$ is employed for the attention-based graph filter and we choose the setting that achieves better performance on the validation set for each dataset. The teleport probability in (6) is set to $\alpha \in \{0.05, 0.1\}$ for the diffusion-based graph filter. Graph filters are sparsified by top-$k$ sparsification with $k \in \{128, 256\}$ or $\epsilon$-threshold sparsification with $\epsilon = 0.0001$. $\lambda$ is set to $0.5$. The other hyperparameters are tuned in the following search space: $K \in \{4, 6, 8, 10, 12\}$, $f_p \in \{32, 64, 96, 128\}$, $T \in \{2, 3, \ldots, 8\}$, $m^- \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$, and $m^+ \in \{0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$.

C. Semisupervised Node Classification

1) Citation Networks: We compare NCGNN with the baselines under different sizes of receptive field on the four citation networks. We only consider the maximum hop of neighborhoods of 5 for NCGNN-A, i.e., $\mathcal{M} = \{1, \ldots, 5\}$, with which the size of interacting neighborhoods is large enough to aggregate appropriate information and the potential risk of suffering from oversmoothing is high. Note that for most GNN models that only aggregate one-hop neighborhood information in a message passing layer (e.g., GCN), we stack multiple layers to enlarge the receptive field, while some other methods directly aggregate high-order neighborhood features (e.g., SGC). We also report the best performance for each baseline with the hyperparameter setting used in their corresponding papers. Table III summarizes the average test accuracy and the best accuracy in each column is highlighted in bold.

Overall, our NCGNN is able to achieve competitive performance on these four assortative graphs. Specifically, we make the following observations.

1) Compared with APPNP that also employs PPR to directly aggregate multihop neighborhood information, the classification accuracy improvement of NCGNN-D is significant. This fact verifies that the designed capsule graph layer indeed brings in more learning capacity than the naive sum aggregator to help learn effective node representations.

2) AdaEdge, DropEdge, and BBGDC can help relieve the oversmoothing issue to some extent but still suffer from drastic performance degradation when stacking more GCN layers to capture long-range dependencies. By contrast, NCGNN-A can achieve better classification accuracy with larger receptive field.

3) DAGNN and GCNII are two strong baselines in solving semisupervised node classification for assortative graphs. Compared with these two methods, NCGNN achieves the state-of-the-art performance for all datasets.

2) Amazon Co-Purchase Graphs: For Amazon co-purchase graphs, we just consider two-hop neighbors for NCGNN-A
TABLE III

| Datasets | Cora | CiteSeer |
|----------|------|----------|
| # of Hop | 1 | 2 | 3 | 4 | 5 | best | 1 | 2 | 3 | 4 | 5 | best |
| GCN [1]  | 75.72 | 80.90 | 81.03 | 79.74 | 77.20 | 81.03 (3) | 66.66 | 68.58 | 66.96 | 64.84 | 59.30 | 68.58 (2) |
| GAT [3]  | 76.57 | 81.28 | 79.54 | 78.70 | 74.76 | 81.28 (2) | 68.39 | 68.79 | 66.19 | 62.85 | 58.32 | 68.79 (2) |
| SGIC [78] | 75.72 | 80.13 | 80.88 | 81.16 | 80.98 | 81.16 (4) | 66.66 | 68.26 | 68.62 | 68.84 | 69.03 | 69.03 (5) |
| APPNP [37] | - | - | - | - | - | - | 82.08 (10) | - | - | - | - |
| JK-Net [35] | - | - | - | - | - | - | 82.08 (10) | - | - | - | - |
| DAGNN [40] | 72.93 | 77.72 | 79.91 | 81.19 | 81.91 | 83.25 (10) | 66.02 | 67.65 | 65.41 | 64.35 | 61.90 | 67.10 (2) |
| GCNN [39] | 77.16 | 80.30 | 80.71 | 81.31 | 81.57 | 83.28 (64) | 65.97 | 67.01 | 66.85 | 68.75 | 70.07 | 70.15 (10) |
| Scattering GCN [41] | - | - | - | - | - | - | 81.04 (17) | - | - | - | - |
| AdaEdge [34] | 75.62 | 80.79 | 79.73 | 78.33 | 76.27 | 80.79 (2) | 66.61 | 68.66 | 66.52 | 64.13 | 59.72 | 68.64 (2) |
| DropEdge [44] | 75.58 | 80.56 | 80.71 | 79.94 | 77.31 | 80.71 (3) | 65.52 | 68.54 | 67.60 | 65.33 | 61.27 | 68.54 (2) |
| BBGDC [45] | - | 80.89 | 81.19 | 79.94 | 75.94 | 81.19 (3) | - | 69.87 | 68.11 | 64.48 | 63.00 | 69.87 (2) |

NCGNN: 79.69 | 81.81 | 82.21 | 83.18 | 86.62 | 83.62 (5) | 68.66 | 69.84 | 70.32 | 70.63 | 70.96 | 70.96 (5) |

as its size is large enough for aggregating useful information (the number of two-hop neighbors of Amazon Photo is averagely 815 and that of Amazon Computers is about 1887). The average test accuracy is reported in Table IV. The best accuracy in each column is highlighted in bold. It shows that NCGNN achieves the best performance on both datasets and outperforms the baselines by large margins on Amazon Computers. We argue that the performance boost is gained from the dynamic routing between node-level capsules. The reason is that previous message passing schemes would inevitably aggregate excessive noise in Amazon Computers where the average receptive field size is very large, while NCGNN can effectively restrain noise from the large neighborhoods to prevent feature overmixing. We also plot t-SNE plots of node representations (hidden features for GCN and GAT, and the output vector of each node’s class node capsule with the maximum length for NCGNN) learned on Amazon Computers in Fig. 3. As can be seen, node embeddings generated by NCGNN are much more separable across different classes (colors), which also verifies that dynamic routing can indeed aggregate appropriate features and restrain harmful noise to relieve the oversmoothing issue.

3) Disassortative Graphs: We now evaluate the performance of our method and existing baselines on six real-world graphs.

TABLE IV

| Datasets | Amazon | Photo | Amazon | Computers |
|----------|--------|-------|--------|-----------|
| GCN [1]  | 92.21  | 82.45 |
| GAT [3]  | 89.36  | 82.80 |
| SGIC [78] | 90.49  | 83.38 |
| APPNP [37] | 91.96  | 81.39 |
| JK-Net [35] | 91.76  | 79.93 |
| DAGNN [40] | 92.69  | 82.78 |
| GCNN [39] | 89.88  | 81.13 |
| Scattering GCN [41] | 81.51  | 68.69 |
| AdaEdge [34] | 92.22  | 82.18 |
| DropEdge [44] | 92.21  | 83.30 |
| BBGDC [45] | 90.59  | 83.00 |

NCGNN: 92.86 | 88.11 |

NCGNN-A: 93.08 | 88.12 |

* "-D" denotes diffusion-based graph filter. "-A" denotes attention-based graph filter. "-" means that the corresponding setting is not applicable. OOM denotes out-of-memory error. The number in parentheses indicates the best-performing size of receptive field. Note that in our NCGNN, we compute the fully personalized PageRank matrix in Eq. (6) for Cora, CiteSeer, and Cora-ML, while use the truncated form in Eq. (7) with $p = 5$ for Pubmed.
disassortative graphs that exhibit low homophily ratio compared with citation networks and Amazon co-purchase graphs. We only consider NCGNN-A here since PPR is commonly based on the assumption of homophily [71]. We consider the receptive field as $\max(M) = 3$. Experimental results are summarized in Table V and the best accuracy in each column is highlighted in bold. From the table, we can find the following facts.

1) The two popular GNN models (i.e., GCN and GAT) and the two strong baselines for learning over assortative graphs (i.e., DAGNN and GCNII) do not perform well on these disassortative graphs and are even outperformed than the simplest deep learning model (i.e., MLP), while our NCGNN-A achieves significant performance gains compared with all these methods.

2) Compared with Geom-GCN and $H_2$GCN, NCGNN-A achieves the state-of-the-art performance on five disassortative graphs. Although NCGNN-A is outperformed by $H_2$GCN-1 on the Texas dataset, the number of misclassified nodes on the test set is averagely only one more than $H_2$GCN-1. We think the reason is that the Texas graph is too small (183 nodes) to demonstrate the superiority of our method.

4) Synthetic Graphs: Following [30], we also use synthetic graphs to further validate the performance of our method. Table VI shows the average test accuracy over ten random weight initializations for each synthetic graph. We evaluate the following baselines: MLP, GCN, GAT, DAGNN, GCNII, and $H_2$GCN. The best accuracy in each column is highlighted in bold. As can be seen, NCGNN-A achieves the best performance with homophily ratio ranging from 0.1 to 0.7 and almost ties with GCNII under the strong homophily setting (i.e., $H(G) = 0.8$ and 0.9). The performance of GCN, GAT, DAGNN, and GCNII is very poor under the heterophily setting and consistently increases with $H(G)$ varying from 0.1 to 0.9. They are even outperformed by MLP that ignores the graph structure when $H(G) < 0.5$, verifying that their message passing schemes cannot aggregate desirable features from the heterophily-dominant neighborhoods. $H_2$GCN also demonstrates competitive performance under both heterophily and homophily compared with other baselines but is outperformed by NCGNN-A in all cases.

### Table V

| Datasets      | Squirrel | Chameleon | Actor | Texas | Cornell | Wisconsin |
|---------------|----------|-----------|-------|-------|---------|-----------|
| MLP           | 29.68    | 46.36     | 35.76 | 81.89 | 81.08   | 85.29     |
| GCN [1]       | 36.89    | 59.82     | 30.26 | 59.46 | 57.03   | 59.80     |
| GAT [3]       | 30.62    | 54.69     | 26.28 | 58.38 | 58.92   | 55.29     |
| DAGNN [40]    | 35.57    | 52.40     | 33.68 | 59.35 | 59.89   | 55.84     |
| GCNII [39]    | 33.55    | 55.32     | 33.85 | 75.46 | 78.11   | 78.04     |
| Geom-GCN-1 [12]| 33.32    | 60.31     | 28.65 | 57.58 | 56.76   | 58.24     |
| Geom-GCN-P [12] | 38.14    | 60.90     | 31.28 | 67.57 | 60.81   | 64.12     |
| Geom-GCN-S [12] | 36.24    | 59.96     | 29.83 | 59.73 | 55.68   | 56.67     |
| $H_2$GCN-1 [30]| 36.42    | 57.11     | 35.86 | 84.86 | 82.16   | 86.67     |
| $H_2$GCN-2 [30]| 37.90    | 59.39     | 35.62 | 82.16 | 82.16   | 83.88     |
| NCGNN-A       | 48.80    | 63.97     | 36.84 | 84.43 | 83.79   | 86.90     |

* The results of DAGNN, GCNII, and NCGNN-A are obtained by our implementations, while the others are from [12], [30].

---

**D. Analysis**

1) **Effect of Training Set Size:** In semisupervised learning, the number of training samples is usually limited. In this section, we take Amazon Computers as an example dataset to evaluate the model performance with varying training set sizes (i.e., 5, 10, 20, and 30 labels per class). Experimental results are demonstrated in Table VII. The best accuracy in each column is highlighted in bold. We can observe that NCGNN consistently achieves the best performance in all cases. Notably, when the training samples are severely scarce (i.e., five labels per class), the superiority of NCGNN becomes more pronounced. Moreover, compared with the two strong baselines (i.e., DAGNN and GCNII) that perform well on the four citation networks, NCGNN demonstrates remarkable improvement, which implies that NCGNN is more robust across different types of graph datasets.

2) **Hyperparameter Sensitivity Analysis:** We investigate the impacts of three important hyperparameters in NCGNN, i.e., the number of primary node capsules ($K$), the routing iterations ($T$), and the dropout rate ($p$). Fig. 4 shows the influence of these hyperparameters on the classification performance of NCGNN-A for Cora and Citeseer with a fixed dataset split. Note that NCGNN-D shows similar performance to NCGNN-A. The results indicate that NCGNN is not sensitive to $K$ and $T$. For $K$ ranging from 4 to 12 and $T$ ranging from 2 to 8, NCGNN consistently outperforms GCN by large margins. Note that more routing iterations should lead to better interpretability. Moreover, we find that our model needs a large dropout rate to avoid overfitting, which is common in the case of complex models and scarce training data of semisupervised learning.

3) **Interpretability Analysis:** The node-level capsule-based message passing is inherently interpretable by adaptively identifying a subset of input node capsules that are most significant to the model’s prediction from the extracted subgraph.
TABLE VI
AVERAGE TEST ACCURACY (%) ON THE SYN-CORA DATASETS

| Homophily Ratio $H(G)$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| MLP                    | 71.60 | 71.97 | 71.01 | 71.92 | 73.18 | 71.14 | 69.61 | 71.30 | 69.34 |
| GCN [1]                | 32.83 | 38.95 | 49.77 | 55.49 | 65.13 | 77.29 | 83.60 | 90.18 | 96.19 |
| GAT [3]                | 30.71 | 36.46 | 47.27 | 53.91 | 64.56 | 75.17 | 81.96 | 89.37 | 95.27 |
| DAGNN [40]             | 48.30 | 55.25 | 63.54 | 66.28 | 75.01 | 83.09 | 87.09 | 91.97 | 96.97 |
| GCNII [39]             | 57.16 | 62.47 | 70.20 | 71.54 | 78.07 | 84.99 | 88.83 | 93.67 | 97.26 |
| H1GCN-I [30]           | 75.06 | 73.88 | 73.77 | 75.68 | 80.08 | 83.28 | 87.41 | 91.97 | 96.48 |
| H2GCN-II [30]          | 74.21 | 72.56 | 73.77 | 74.66 | 80.08 | 84.59 | 86.82 | 92.32 | 95.94 |

TABLE VII
AVERAGE TEST ACCURACY (%) ON THE AMAZON COMPUTERS DATASET WITH VARYING TRAINING SET SIZES

| # of Labels per Class | 5   | 10  | 20  | 30  |
|-----------------------|-----|-----|-----|-----|
| GCN [1]               | 73.21 | 79.73 | 82.45 | 83.92 |
| GAT [3]               | 73.86 | 76.87 | 82.80 | 83.25 |
| SGC [78]              | 73.79 | 80.47 | 83.38 | 84.19 |
| APPNP [37]            | 71.28 | 77.82 | 81.39 | 82.09 |
| JK-Net [35]           | 68.67 | 76.50 | 79.93 | 80.37 |
| DAGNN [40]            | 72.85 | 78.42 | 82.78 | 84.49 |
| GCNII [39]            | 65.88 | 73.21 | 81.13 | 82.73 |
| Scattering GCN [41]   | 50.79 | 65.44 | 68.69 | 69.11 |
| AdaEdge [34]          | 73.09 | 79.68 | 82.18 | 83.79 |
| DropEdge [44]         | 73.36 | 80.03 | 82.30 | 83.98 |
| NCGNN-D               | 83.29 | 86.44 | 88.11 | 88.36 |
| NCGNN-A               | 83.95 | 86.34 | 88.12 | 88.33 |

Note that the interpretability of primary node capsules has been addressed in existing methods such as [65]. Therefore, we elaborate on the inherent interpretability within the message passing scheme.

We first analyze the attention-based graph filter for identifying the subgraph. As shown in Fig. 5, larger attention scores are assigned to higher order neighborhoods, which also verifies that NCGNN-A is indeed exempt from the oversmoothing problem in extracting long-range dependencies. Moreover, we take a close look at the selected target node. Taking one node from Cora that is misclassified by GCN while assigned true label by NCGNN-A as an example, we visualize the receptive field (a subgraph of ego network) of NCGNN-A with $\max(M_j) = 5$ and compare it with the two-hop neighborhood. For a better view, we only plot the top-32 neighbors in Fig. 5(f). We find that: 1) the two-hop receptive field of GCN is too small to aggregate adequate node features for correctly classifying the target node; 2) GCN (five-hop) exaggerates the classification error and suffers from the oversmoothing issue; and 3) NCGNN-A (five-hop) correctly classifies the target node with the largest probability, verifying that higher order and more relevant messages from neighboring nodes can be extracted by the five-hop attention-based graph filter to help NCGNN-A learn effective node representation for classification.

Furthermore, we provide the heatmaps of average coupling coefficients (i.e., $(1/N)\sum_{j=1}^{N}c_{ijkl}$) adaptively computed on eight benchmark datasets in Fig. 6 to visualize the average contributions of the input primary node capsules to each class. As shown in [65] that interprets capsule-based networks, the coupling coefficient $c_{ijkl}$ can be regarded as the contribution of the input node capsule $h_{ji}^{(k)}$ to the capsule $p_{lj}^{(l)}$ associated with the properties of class $l$ extracted from node $v_j$. Here, we set the number of primary node capsules to 8 for all datasets. The brightness values in each row of the coupling coefficient matrices measure the average influence of a subset of important primary node capsules on the corresponding class. For example, in the Amazon Photo dataset, the fourth primary node capsule is most influential for predicting the node as the second class on average. Thus, these adaptively computed coupling coefficients make the node-level capsule-based message passing transparent in interpreting which part of input node capsules contributes most to the model’s prediction.

4) Robustness Analysis: The message passing scheme of our capsule graph layer is able to adaptively restrain noisy features from neighborhoods. In this section, we study the robustness of our method under the missing edge scenarios, where we perturb graphs by randomly removing a certain fraction of edges of the original graph. We use two synthetic graphs with heterophily ($H(G) = 0.3$) and homophily ($H(G) = 0.7$) to evaluate the model robustness.

We compare with two strong baselines for assortative graphs (i.e., DAGNN and GCNII) and the state-of-the-art method for disassortative graphs (i.e., H2GCN). Fig. 7 presents the classification results of these methods in the incomplete graph scenarios, where the results are obtained by averaging over ten random weight initializations for each perturbation rate. We observe that our NCGNN consistently outperforms all the
baselines across all perturbation rates on both assortative and disassortative graphs and demonstrates surprising robustness for learning over the disassortative graph. H2GCN is also disassortative graphs and demonstrates surprising robustness across all perturbation rates on both assortative and homophily.

Fig. 7. Robustness analysis of NCGNN under both heterophily and homophily.

V. CONCLUSION

In this article, we propose NCGNN for semisupervised node classification. NCGNN represents nodes as groups of capsules. By leveraging the designed graph filter and capsule graph layer, each node-level capsule adaptively aggregates a subset of advantageous capsules and restrains irrelevant messages from multithop neighborhoods. This novel message passing scheme is demonstrated to overcome feature overmixing to produce better node representations. Experiments on 12 real-world graphs as well as synthetic graphs show that NCGNN performs well under both homophily and heterophily and is exempt from the oversmoothing issue. Moreover, it is inherently interpretable by identifying a subset of node features that are crucial for the model’s prediction from the extracted subgraph.

REFERENCES

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. 5th Int. Conf. Learn. Rep., Toulon, France, Apr. 2017, pp. 1–14.

[2] H. Gao, Z. Wang, and S. Ji, “Large-scale learnable graph convolutional networks,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, London, U.K., Jul. 2018, pp. 1414–1426.

[3] P. Velićković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in Proc. 6th Int. Conf. Learn. Rep., Vancouver, BC, Canada, Apr. 2018, pp. 1–12.

[4] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Honolulu, HI, USA, Dec. 2017, pp. 3693–3702.

[5] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” ACM Trans. Graph., vol. 38, no. 5, pp. 1–12, Nov. 2019.

[6] D. Duvenaud et al., “Convolutional networks on graphs for learning molecular fingerprints,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 2015, pp. 2224–2232.

[7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in Proc. 34th Int. Conf. Mach. Learn., Sydney, NSW, Australia, Aug. 2017, pp. 1263–1272.

[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional networks for web-scale recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, London, U.K., Jul. 2018, pp. 974–983.

[9] K. Zhang, Y. Zhu, J. Wang, and J. Zhang, “Adaptive structural fingerprints for graph attention networks,” in Proc. 8th Int. Conf. Learn. Rep., Addis Ababa, Ethiopia, Apr. 2020, pp. 1–12.

[10] L. Gong and Q. Cheng, “Exploiting edge features for graph neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 9203–9211.

[11] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, “Composition-based multi-relational graph convolutional networks,” in Proc. 8th Int. Conf. Learn. Rep., Addis Ababa, Ethiopia, Apr. 2020 pp. 1–15.

[12] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geom-GCN: Geometric graph convolutional networks,” in Proc. 8th Int. Conf. Learn. Rep., Addis Ababa, Ethiopia, Apr. 2020, pp. 1–12.

[13] Z. Ye, K. S. Liu, T. Ma, J. Gao, and C. Chen, “Curvature graph network,” in Proc. 8th Int. Conf. Learn. Rep., Addis Ababa, Ethiopia, Apr. 2020, pp. 1–15.

[14] M. McPherson, L. Smith-Lovin, and J. M. Cook, “Birds of a feather: Homophily in social networks,” Am. J. Sociol., vol. 77, no. 5, pp. 1415–1444, 2001.

[15] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, “GNNEplanation: Generating explanations for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., Vancouver, BC, Canada, Dec. 2019, pp. 9244–9255.

[16] H. Yuan, J. Tang, X. Hu, and S. Ji, “XGNN: Towards model-level explanations of graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining, San Diego, CA, USA, Aug. 2020, pp. 430–438.

[17] Q. Li, Z. Han, and X. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), New Orleans, LA, USA, Feb. 2018, pp. 3538–3545.

[18] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, “Protein interface prediction using graph convolutional networks,” in Proc. Adv. Neural Inf. Process. Syst., Long Beach, CA, USA, Dec. 2017, pp. 6530–6539.

[19] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” in Proc. Int. Conf. Artif. Neural Netw., Espoo, Finland, Jun. 2011, pp. 44–51.
A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and J. Tang, J. Sun, C. Wang, and Z. Yang, “Social influence analysis in B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph disentangled networks,” in Proc. 34th AAAI Conf. Artif. Intell., New York, NY, USA, Feb. 2020, pp. 4916–4923.

S. Abu-El-Haija et al., “Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” in Proc. Int. Conf. Mach. Learn. (ICML), Long Beach, CA, USA, May 2019, pp. 21–29.

J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, “Graph convolutional networks with motif-based attention,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manag., Beijing, China, Nov. 2019, pp. 499–508.

H. Mehlhorn and F. Schreiber, Small-World Property of Biomedical Informatics, University of California at San Diego, La Jolla, USA: Springer, 2013, pp. 1957–1959.

J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in Proc. Adv. Neural Inf. Process. Syst., Vancouver, BC, Canada, Dec. 2019, pp. 13354–13366.

A. Bojchevski et al., “Scaling graph neural networks with approximate PageRank,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Diego, CA, USA, Aug. 2020, pp. 2464–2473.

A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of internet portals with machine learning,” Inf. Retr., vol. 3, no. 2, pp. 127–163, Jul. 2000.

A. Bojchevski and S. Günnemann, “Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking,” in Proc. 6th Int. Conf. Learn. Rep., Vancouver, BC, Canada, May 2018, pp. 1–13.

O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” in Proc. Relat. Rep. Learn. Workshop (NIPS), Montréal, QC, Canada, Dec. 2018, pp. 1–11.

B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node embedding,” 2019, arXiv:1909.13021.

J. Tang, J. Sun, C. Wang, and Z. Yang, “Social influence analysis in large-scale networks,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Paris, France, Jun. 2009, pp. 807–816.

F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” in Proc. ICML, Long Beach, CA, USA, Feb. 2019, pp. 6861–6871.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. 3rd Int. Conf. Learn. Rep., San Diego, CA, USA, May 2015, pp. 1–15.

L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.

Rui Yang received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include graph signal processing and graph-based machine learning.

Wenrui Dai (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2005, 2008, and 2014, respectively. He is currently an Associate Professor with the Department of Computer Science and Engineering, SJTU. Before joining SJTU, he was with the faculty of the University of Texas Health Science Center at Houston, Houston, TX, USA, from 2018 to 2019. He was a Post-Doctoral Scholar with the Department of Biomedical Informatics, University of California at San Diego, La Jolla, CA, USA, from 2015 to 2018, and a Post-Doctoral Scholar with the Department of Computer Science and Engineering, SJTU, from 2014 to 2015. His research interests include learning-based image/video coding, image/signal processing, and predictive modeling.

Hongkai Xiong (Senior Member, IEEE) received the B.S. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2003. Since 2003, he has been with the Department of Electronic Engineering, SJTU. He was an Associate Professor from 2005 to 2011 and an Assistant Professor from 2003 to 2005. From 2007 to 2008, he was a Research Scholar with the Department of Electrical and Computer Engineering, Carnegie Mellon University (CMU), Pittsburgh, PA, USA. From 2011 to 2012, he was a Scientist with the Division of Biomedical Informatics, University of California at San Diego (UCSD), La Jolla, CA, USA. He is currently a Distinguished Professor with the Department of Electronic Engineering and the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai. From June 2011 to June 2012, he was with the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, CA, USA, as a Visiting Professor. Her research interests include multimedia networking and communications, machine learning, and network resource optimization. She has published over 100 international journal articles/conference papers on these topics.

Dr. Zou was granted the National Science Fund for Outstanding Young Scholar in 2016. She was a recipient of Shanghai Young Rising-Star Scientist in 2011. She is an Associate Editor of Digital Signal Processing. She was the coauthor of a Top 10% Paper Award in VCIP 2016.

Chenglin Li (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2007, 2009, and 2015, respectively. From 2011 to 2013, he was with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, as a Visiting Ph.D. Student. From 2015 to 2017, he was a Post-Doctoral Research Fellow with the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. From 2017 to 2018, he was a Senior Researcher with the Chair of Media Technology (LMT), Technical University of Munich (TUM), Munich, Germany, supported by the Hildegard Maier Research Fellowship of the Alexander von Humboldt Foundation for post-doctoral researchers. Since 2018, he has been an Associate Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. His main research interests include network-oriented image/video processing and communication, network-based optimization for video sources, adaptive multimedia communication systems, and caching and edge computing for multimedia delivery.

Dr. Li was awarded the Microsoft Research Asia (MSRA) Fellow in 2011 and the Alexander von Humboldt Research Fellow in 2017. He received the IEEE VCIP Best 10% Paper Award in 2016.

Junnii Zou (Member, IEEE) received the M.S. and Ph.D. degrees in communication and information system from Shanghai University, Shanghai, China, in 2004 and 2006, respectively. From March 2006 to Jan. 2017, she was with the School of Communication and Information Engineering, Shanghai University, where she is currently a Full Professor. Since February 2017, she has been a Full Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai. From June 2011 to June 2012, she was with the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, CA, USA, as a Visiting Professor. Her research interests include multimedia networking and communications, machine learning, and network resource optimization. She has published over 100 international journal articles/conference papers on these topics.

Dr. Zou was granted the National Science Fund for Outstanding Young Scholar in 2016. She was a recipient of Shanghai Young Rising-Star Scientist in 2011. She is an Associate Editor of Digital Signal Processing. She was the coauthor of a Top 10% Paper Award in VCIP 2016.

Rui Yang received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include graph signal processing and graph-based machine learning.

Wenrui Dai (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2005, 2008, and 2014, respectively. He is currently an Associate Professor with the Department of Computer Science and Engineering, SJTU. Before joining SJTU, he was with the faculty of the University of Texas Health Science Center at Houston, Houston, TX, USA, from 2018 to 2019. He was a Post-Doctoral Scholar with the Department of Biomedical Informatics, University of California at San Diego, La Jolla, CA, USA, from 2015 to 2018, and a Post-Doctoral Scholar with the Department of Computer Science and Engineering, SJTU, from 2014 to 2015. His research interests include learning-based image/video coding, image/signal processing, and predictive modeling.