ADAPTIVE ATTENTION GRAPH CAPSULE NETWORK

Xiangping Zheng, Xun Liang, Bo Wu, Yuhui Guo, Hui Tang

School of Information, Renmin University of China,Zhongguancun Street, Beijing

ABSTRACT

From the perspective of the spatial domain, Graph Convolutional Network (GCN) is essentially a process of iteratively aggregating neighbor nodes. However, the existing GCNs using simple average or sum aggregation may neglect the characteristics of each node and the topology between nodes, resulting in a large amount of early-stage information lost during the graph convolution step. To tackle the above challenge, we innovatively propose an adaptive attention graph capsule network, named AA-GCN, for graph classification. We explore various propagation mechanisms of graphs and present an attention mechanism combined with graph propagation and capsules to generate capsule nodes, preserving the spatial topology between nodes. We also propose a graph adaptive attention mechanism to investigate the context information in different global GCN layers, so as to effectively improve the next dynamic routing connection and the final graph classification. Experiments show that our proposed algorithm achieves either state-of-the-art or competitive results across all the datasets.

***Index Terms*—** Graph Convolutional Network, Capsule Network, Attention Mechanism

1. INTRODUCTION

Driven by a considerable number of data and powerful computing resources, deep learning has made a breakthrough in many application fields with its powerful representation ability [1]. With the increasing application of non-Euclidean data, most deep learning models have limited performance in processing graph data [2]. As a representative method of combining deep learning with graph data, the emergence of GCN promotes neural network technology in graph data learning tasks. Recent years have seen increasing attention to graph convolutional networks, such as social networks [3], protein molecules [4], and transportation networks [5].

The general approach with GCN is to treat the underlying graph as a computational graph and learn neural network primitives [6]. However, GCN for incorporating graph structure information in neural network operations aims to enforce similarity between representations of adjacent nodes, which essentially implements local smoothing of neuron activations over the graph [7, 8]. These methods have two limitations as follows: (1) *graph convolution operation may lose the structural information of nodes in neighborhoods*. For instance, GCN simply sums or averages the normalized messages from all one-hop neighbors [9]. Such aggregation loses the structural information of nodes in the neighborhood, resulting in much early information lost during the graph convolution step. (2) *the aggregators cannot capture long-range dependencies in graphs*. The neighborhood is defined as all neighbors one hop away or many hops away. In other words, only messages from nearby nodes are aggregated. In such cases, the representation ability of GCN may be significantly limited since they cannot capture the crucial features from distant but informative nodes [8].

Capsule networks use groups of neurons to represent visual features [10]. Each group (named a capsule) forms a local feature and represents one visible entity, such as a human mouth. Each lower-level capsule votes for each higher-level capsule, such as the mouth belongs to a face, by the dynamic routing algorithm. Capsule vector can save node information and reserve topological structure attributes of its space, such as location, direction, connection, etc. Inspired by Capsule Networks [11], we propose an adaptive attention graph capsule network (AA-GCN) to address the above challenges. To tackle **limitation 1**, we explore various propagation mechanisms of graphs and present an attention mechanism combined with capsules to generate capsule nodes, preserving the spatial topology between nodes. Therefore, the capsule can capture more information about the input graph than other GCN methods. We solve **limitation 2** via presenting a graph adaptive attention mechanism to make the model pay more attention to the more essential nodes or neighborhood information in the graph. The capsule idea can make up for the lack of spatial information caused by the convolution operation of the basic graph. This strategy enables AA-GCN not only to capture the topological structure of the graph but also to have the ability to capture long-range graph dependencies.

Our main contributions are summarized as follows: (1) We explore various propagation mechanisms of graphs and present an attention mechanism combined with graph propagation and capsules to generate capsule nodes to capture more graph attribute features. (2) We present a graph adaptive attention mechanism to make the model pay more attention to the related nodes and spatial domain information of some supernodes in the graph. (3) Comprehensive empirical studies

demonstrate that our proposed algorithm achieves either state-of-the-art or competitive results across multiple datasets.

2. APPROACH

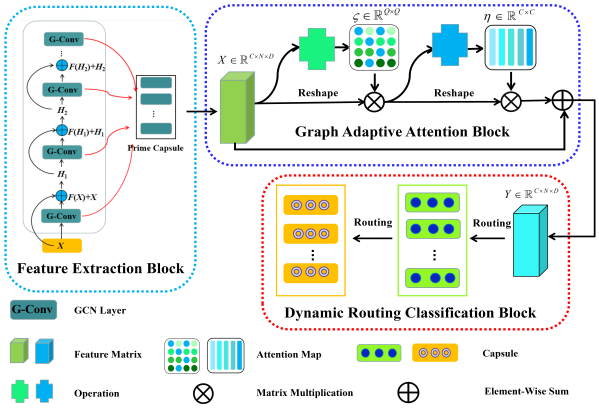

Fig. 1. Framework of AA-GCN. At first, we extract multi-scale node features from different GCN layers and make them into the form of the primary capsules. Then, the graph adaptive attention mechanism module is used to capture long-range dependencies to connect capsule layers better. Finally, Dynamic Routing is applied to perform graph classification.

As shown in **Figure 1**, our AA-GCN model comprises three blocks: feature extraction, graph adaptive attention mechanism, and capsule dynamic routing classification. We now detail the proposed model as follows.

2.1. Feature Extraction

In a real-world setting, most GCN methods are based on repeated aggregations of information from nodes' neighbors in a graph, a naive implementation leads to repeated and inefficient aggregations. What's more, each node's embedding is considered multiple separate scalar features, which is proved to have limited ability in preserving graph attributes. Furthermore, many early information is lost during each convolution step with the number of layers increasing. We aim to explore various GCN structures and get the node information from each layer in GCN to form capsule nodes and embed these capsule nodes into graphs for recognition and classification. Previous works mainly used GCN as the main body and connected it with the dense layer to distinguish graphs. Recent works [7] indicate that GCN extended to deep architecture awkwardly due to the high complexity and excessive smoothness caused by overlapping multi-layer graph convolution. This paper focuses on GCN as a node feature extraction method rather than GCN as the main body. We adopt four different GCN architectures as node feature extractors: Basic-GCN, Plain-GCN, Res-GCN, and Dense-GCN. We now detail the four feature extractors as shown in **Table 1**. We take the original GCN proposed by Kipf and Welling as a representative to briefly recapitulate its main design [9], where $\mathbf{H}^{(l+1)}$ is the output of the $(l+1)$-th convolutional layer of GCN. $\hat{\mathbf{A}}$ is the graph adjacency matrix with self-connections. $\mathbf{W}^{(l)}$ is the trainable layer-wise weight matrix. The node features are defined as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \in \mathbb{R}^{N \times d}$, where d represents the dimensions of feature.

Table 1. Four different GCN propagation formulas

Methods	Propagation formula
Basic-GCN	$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$
Plain-GCN	$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) + \mathbf{X}$
Res-GCN	$\mathbf{H}^{(l+l)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) + \mathbf{H}^{(l)}$
Dense-GCN	$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) + \sum_{i=1}^{l}\mathbf{H}^{i} + \mathbf{X}$

Next, we present an attention mechanism combined with graph propagation and capsules to generate capsule nodes to capture more graph attribute features (e.g., spatial topology) reflected by different levels. We extract multi-scale node features from different GCN layers, and the features are expressed in the form of node capsules. Then, we exhibit attention behavior and form a hierarchical node representation $\mathbf{H}_{Att} \in \mathbb{R}^{N \times d}$ as below:

$$\mathbf{H}_{Att} = \sum_{i=1}^{k} \beta_i (\mathbf{Z}^{(i+1)} + \mathbf{H}^{(i)}) \tag{1}$$

For simplicity, we use vanilla attention to identify the importance of each GCN aggregation result, in which β is the attention weight. It is worth mentioning that the achieving node capsules will be a hierarchical representation containing the most valuable information from different convolution processes. We obtain the node information from each layer in GCN to form capsule nodes and embed these capsule nodes into graphs for recognition and classification. Finally, we combine the outputs of multiple GCN layers and combine $\mathbf{H}_{Att}$ to form new capsule nodes $\mathbf{H} \in \mathbb{R}^{C \times N \times d}$ as follows:

$$\mathbf{H} = \mathcal{M}(\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(n)}, \mathbf{H}_{Att}) \tag{2}$$

2.2. Graph Adaptive Attention Mechanism

We extract multi-scale node features from different GCN layers and make them into the form of the primary capsules. However, these node features are redundant and the connection between capsule layers is a whole connection process by a dynamic routing mechanism. If the primary capsules directly connect to the following capsules, it is difficult to train the weight distribution between different capsule layers, resulting in difficult convergence and poor recognition ability. To solve the above problem, we propose a graph adaptive attention mechanism to capture the dependency of different relational layers so as to address the capsule connection task.

Graph capsule adaptive attention mechanism consists of two steps: node attention mechanism and layer attention mechanism. Node attention mechanism can establish rich context relations on local features to enhance its representation ability. In contrast, the layer attention mechanism aims to explicitly model the interdependence between the channels obtained by different GCN layers, so as to achieve the spatial mapping of mutual characteristics. Let $\mathbf{X} \in \mathbb{R}^{C\times N\times d}$ be the output feature matrix extracted from the different GCN layers, where C is number of feature channels, N is the number of nodes and d represents the size of the capsule. Let $f(\mathbf{x})$ and $g(\mathbf{x})$ be two feature extractors which are 1×1 convolutional kernels. we first feed $\mathbf{X}$ into two feature extractors to generate two new feature maps $\mathbf{A}$ and $\mathbf{B}$, respectively, where $\{\mathbf{A}, \mathbf{B}\} \in \mathbb{R}^{C\times N\times d}$. Then, we reshape $\mathbf{A}$ to $\mathbb{R}^{C\times Q}$ where $Q = N \times d$. We define the following operation as: $S_{ij}(\mathbf{x}) = f(\mathbf{x}_i)^T f(\mathbf{x}_j) = \mathbf{A}_i^T \mathbf{A}_j$. $f(\mathbf{x}) = \mathbf{W}_f x$, where $\mathbf{W}_f \in \mathbb{R}^{C\times C}$are the learned weight matrices. we apply a softmax layer to calculate the spatial attention map $\zeta_{ij} \in \mathbb{R}^{Q\times Q}$. Here, ζ_{ij} indicates the extent to which the model attends to i_{th} location when synthesizing the j_{th} region, The output of the node position mechanism is $\mathbf{C} = (\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_N) \in \mathbb{R}^{C\times N\times d}$, where,

$$\mathbf{C}_j = \sum_{i=1}^{N} (\zeta_{ij}\mathbf{B}_i), \mathbf{B}_i = g(\mathbf{x}_i) = \mathbf{W}_g \mathbf{x}_i \tag{3}$$

In the above formulation, similarly to $\mathbf{W}_f$, $\mathbf{W}_g \in \mathbb{R}^{C\times C}$ is also a learned weight matrix. We reshape $\mathbf{B}$ to $\mathbb{R}^{C\times Q}$ and perform a matrix multiplication between the attention map ζ_{ij} and the $\mathbf{B}$. In addition, we reshape the result to $\mathbb{R}^{C\times N\times d}$.

Next, we conduct the layer attention mechanism. Different from the node attention mechanism, this step does not need convolution operation but directly calculates, which is relatively simple. Specifically, We reshape $\mathbf{C}$ to $\mathbf{D} \in \mathbb{R}^{C\times Q}$, and then perform a matrix multiplication between $\mathbf{D}$ and the transpose of $\mathbf{D}$ directly. After that we apply a softmax layer to calculate the layer attention map $\eta \in \mathbb{R}^{C\times C}$,where η_{ij} measures the i_{th} layer's impact on the j_{th} channel. To obtain the final Attention map, we reshape $\mathbf{C}$ to $\mathbb{R}^{C\times Q}$ and apply matrix multiplication between η_{ij} and $\mathbf{C}$. Finally, we further multiply the output of the attention layer by a scale parameter and add back the input feature $\mathbf{X}$. Therefore, the final output $\mathbf{Y} \in \mathbb{R}^{C\times N\times d}$ is given as follow: $\mathbf{Y}_j = \alpha \sum_{i=1}^{C} (\eta_{ij}\mathbf{C}_i) + \mathbf{X}_j$. where α is initialized as 0 and gradually learns to assign more weight [12]. Note that our attention modules are straightforward. They do not increase many parameters yet effectively enhance the feature representation.

2.3. Capsule Dynamic Routing Classification

We designed this block for graph classification. The primary capsule outputs to the capsule layer through the graph adaptive attention mechanism module. The flexible and orderly dynamic capsule mechanism ensures that the output of the capsule is sent to the appropriate following capsule. In the capsule layer, the active capsules in the lower layer are transported to the higher layer through a dynamic routing mechanism. Finally, capsule layer is applied again to generate final class capsules $\mathbf{u}^{(T)} \in \mathbb{R}^{k\times d_t}$, where k is the number of graph classes. Here, we use margin loss function [11] to calculate the classification loss and it is computed as:

$$\begin{aligned} Loss_k = {} & T_k max(0, m^+ - \left\|\mathbf{u}_k^T\right\|)^2 \\ & + \lambda(1 - T_k) max(0, \left\|\mathbf{u}_k^T\right\| - m^-)^2 \end{aligned} \tag{4}$$

where $\lambda = 0.5$, $m^+ = 0.9$ and $m^- = 0.1$. $T_k = 1$ means that the graph belong to class k, otherwise it is 0 in other cases.

3. EXPERIMENTS

3.1. Datasets and setting

We selected a total of 7 graph datasets, which are divided into social and biological categories. Specifically, 4 biological graph datasets: MUTAG, PTC, ENZYMES, PROTEINS and 3 social network datasets: COLLAB, IMDB-BINARY, IMDB-MULTI [15, 25] are used for our experiments. Due to the large literature, we could not compare to every graph neural network, but to some classical ones and those closely related to our approach. Following the conventional settings, we performed 10-fold cross validation with LIBSVM [26] (8 folds for training, 1 fold for validation, and 1 fold for testing) using one training fold for hyperparameter searching, and repeated the experiments 10 times. We then report the average performances on the validation and test sets, by performing overall experiments 10 times with different seeds.

3.2. Comparing with State-of-the-art

We compare various methods on multiple datasets. These methods are classic and state-of-the-art algorithms, which are outstanding in a particular field. Therefore, we select the top-2 performers on each dataset to reflect the comprehensiveness and generalization of the algorithm. **Table 2** shows the classification accuracy achieved by our proposal methods on the datasets as mentioned above compared with several graph kernel and GNN baselines, whose results are obtained from the corresponding original papers. As the table demonstrates, our proposed algorithm achieves either state-of-the-art or competitive results across all the datasets. More specifically, our four proposal methods perform well on biological datasets. The Res-AAGCN has achieved 4 top-2 performers on 4 biological datasets, ranking first. Meanwhile, Our four models have achieved good performance results, which demonstrate that our model effectively represents the features in the form of capsules by extracting various abstract features from different GCN layers. Our methods are also very competitive with SOTA graph kernel methods and GNN. Again, this trend is

Table 2. Graph classification accuracy (%) of our method compared with state-of-the-art graph classification methods on Biological Datasets. The top-2 performers on each dataset are shown in **bold**.

Methods	Biological				Social Network		
	MUTAG	PTC	ENZYMES	PROTEINS	COLLAB	IMDB-B	IMDB-M
GK [13]	81.58 ± 0.36	57.26 ± 1.41	26.61 ± 0.99	71.67 ± 0.55	72.84 ± 0.28	65.87 ± 0.98	43.89 ± 0.38
WL [14]	82.05 ± 0.36	57.97 ± 0.49	52.22 ± 1.26	74.68 ± 0.49	79.02 ± 1.77	73.40 ± 4.63	49.33 ± 4.75
DGK [15]	87.44 ± 2.72	60.08 ± 2.55	53.43 ± 0.91	75.68 ± 0.54	73.09 ± 0.25	66.96 ± 0.56	44.55 ± 0.52
AWE [16]	-	-	35.77 ± 0.36	-	73.90 ± 1.90	74.50 ± 5.80	51.50 ± 3.60
WEGL [17]	-	-	60.00 ± 6.30	76.50 ± 4.25	80.60 ± 2.00	75.40 ± 5.00	52.30 ± 2.90
GMT [18]	83.44 ± 1.33	-	-	75.09 ± 0.59	80.74 ± 0.54	73.48 ± 0.76	50.66 ± 0.82
GraphNorm [19]	91.60 ± 6.50	64.90 ± 7.50	-	$\mathbf{77.40 \pm 4.90}$	80.20 ± 1.00	76.00 ± 3.70	-
ASAP [20]	77.83 ± 1.49	-	-	73.92 ± 0.63	78.64 ± 0.50	72.81 ± 0.50	50.78 ± 0.75
DGCNN [21]	85.83 ± 1.66	58.59 ± 2.47	51.00 ± 7.29	75.54 ± 0.94	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85
GCN [9]	87.20 ± 5.11	-	66.50 ± 6.91	75.65 ± 3.24	81.72 ± 1.64	73.30 ± 5.29	51.20 ± 5.13
GIN [22]	89.40 ± 5.60	64.60 ± 7.00	-	76.20 ± 2.80	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80
GFN [23]	90.84 ± 7.22	-	70.17 ± 5.58	76.46 ± 4.06	81.50 ± 2.42	73.00 ± 4.35	51.80 ± 5.16
GCAPS-CNN [24]	-	66.01 ± 5.91	61.83 ± 5.39	76.40 ± 4.17	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10
CapsGNN [25]	86.67 ± 6.88	-	54.67 ± 5.67	76.28 ± 3.63	79.62 ± 0.91	73.10 ± 4.83	50.27 ± 2.65
Basic-AAGCN	92.49 ± 3.36	$\mathbf{66.63 \pm 4.23}$	69.25 ± 4.23	77.14 ± 2.68	81.82 ± 1.25	$\mathbf{76.84 \pm 4.12}$	51.68 ± 2.36
Plain-AAGCN	93.24 ± 4.10	65.52 ± 3.54	71.14 ± 5.16	77.28 ± 4.85	80.75 ± 2.21	75.61 ± 2.76	51.96 ± 4.10
Dense-AAGCN	$\mathbf{93.85 \pm 4.23}$	66.54 ± 0.36	$\mathbf{72.12 \pm 5.12}$	77.21 ± 3.36	$\mathbf{82.92 \pm 2.62}$	75.82 ± 3.25	$\mathbf{52.63 \pm 3.28}$
Res-AAGCN	$\mathbf{94.01 \pm 3.16}$	$\mathbf{67.02 \pm 6.31}$	$\mathbf{72.35 \pm 5.85}$	$\mathbf{78.20 \pm 4.12}$	$\mathbf{83.17 \pm 1.85}$	$\mathbf{77.23 \pm 3.57}$	$\mathbf{52.94 \pm 4.10}$

continued to be the same on social datasets. The Res-AAGCN also has achieved the best performers on 3 social network datasets, ranking first. Experiments show that our models are general and effective methods.

3.3. Ablation Study for Adaptive Attention Mechanism

Table 3. Ablation study on **COLLAB** dataset.

Basic-AAGCN			Plain-AAGCN		
NAM	**LAM**	**Accuracy**	**NAM**	**LAM**	**Accuracy**
		77.72%			77.15%
✓		81.53%	✓		80.28%
	✓	80.62%		✓	79.45%
✓	✓	81.82%	✓	✓	80.75%
Res-AAGCN			Dense-AAGCN		
NAM	**LAM**	**Accuracy**	**NAM**	**LAM**	**Accuracy**
		79.66%			79.43%
✓		82.28%	✓		81.88%
	✓	81.56%		✓	81.28%
✓	✓	83.17%	✓	✓	82.92%

We employ the graph global adaptive attention block on the capsule graph network to capture long-range dependencies to connect capsule layers better. The graph global adaptive attention is composed of two parts: node attention mechanism **(NAM)** and layer attention mechanism **(LAM)**. To verify the performance of each attention module, we choose the dataset **COLLAB** as an example and conduct experiments with different settings in **Table 3**. The graph global adaptive attention block improves the performance remarkably. For method Res-AAGCN, compared with the baseline which does not use the attention mechanism, employing node attention module yields a result of 82.28% in Accuracy, which brings 2.62% improvement. Meanwhile, utilizing the layer attention module individually outperforms the baseline by 1.90%. When we integrate the two attention modules, the performance further improves to 83.17%.This trend is continued to be the same on the methods Basic-AAGCN, Plain-AAGCN, Dense-AAGCN. Results show that the graph global adaptive attention blocks bring great benefit to improve the accuracy of classification.

4. CONCLUSION

In this work, we propose a capsule graph neural network based on an adaptive attention mechanism named AA-GCN to tackle the potential loss of relevant node information caused by graph convolution operations. The idea of a capsule can make up for the lack of spatial information and enhance the adaptability of GCN to graph data. We design different GCN connection methods to extract multi-scale node features from various GCN layers. AA-GCN can be trained end to end, and we achieve a more considerable performance on biological and social datasets. Meanwhile, the research of capsule networks remains in its infancy, and their application in GCN has potential development, which is worthy of our further exploration.

5. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (62072463, 71531012), and the National Social Science Foundation of China (18ZDA309). Xun Liang is the corresponding author of this paper.

6. REFERENCES

[1] Miao Xin, Shentong Mo, and Yuanze Lin, "EVA-GCN: head pose estimation based on graph convolutional networks," in *CVPR*. 2021, pp. 1462–1471, Computer Vision Foundation / IEEE.

[2] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem, "Deepgcns: Can gcns go as deep as cnns?," in *ICCV*. 2019, pp. 9266–9275, IEEE.

[3] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang, "Session-based social recommendation via dynamic graph attention networks," in *WSDM*. 2019, pp. 555–563, ACM.

[4] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur, "Protein interface prediction using graph convolutional networks," in *NeurIPS*, 2017, pp. 6530–6539.

[5] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*. 2018, OpenReview.net.

[6] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, 2018, pp. 4805–4815.

[7] Qimai Li, Zhichao Han, and Xiao-Ming Wu, ," in *AAAI*. 2018, pp. 3538–3545, AAAI Press.

[8] Yimeng Min, Frederik Wenkel, and Guy Wolf, "Scattering GCN: overcoming oversmoothness in graph convolutional networks," in *NeurIPS*, 2020.

[9] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*. 2017, OpenReview.net.

[10] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov, "Capsules with inverted dot-product attention routing," in *ICLR*. 2020, OpenReview.net.

[11] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton, "Dynamic routing between capsules," in *NIPS*, 2017, pp. 3856–3866.

[12] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu, "Dual attention network for scene segmentation," in *CVPR*. 2019, pp. 3146–3154, Computer Vision Foundation / IEEE.

[13] Nino Shervashidze and S. V. N. Vishwanathan, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, 2009, vol. 5, pp. 488–495.

[14] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, 2011.

[15] Pinar Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *SIGKDD*. 2015, pp. 1365–1374, ACM.

[16] Sergey Ivanov and Evgeny Burnaev, "Anonymous walk embeddings," in *ICML*, Jennifer G. Dy and Andreas Krause, Eds. 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 2191–2200, PMLR.

[17] Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann, "Wasserstein embedding for graph learning," in *ICLR*. 2021, OpenReview.net.

[18] Jinheon Baek, Minki Kang, and Sung Ju Hwang, "Accurate learning of graph representations with graph multiset pooling," in *ICLR*. 2021, OpenReview.net.

[19] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang, "Graphnorm: A principled approach to accelerating graph neural network training," in *ICML*. 2021, vol. 139, pp. 1204–1215, PMLR.

[20] Ekagra Ranjan, Soumya Sanyal, and Partha P. Talukdar, "ASAP: adaptive structure aware pooling for learning hierarchical graph representations," in *AAAI*. 2020, pp. 5470–5477, AAAI Press.

[21] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI*. 2018, pp. 4438–4445, AAAI Press.

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, "How powerful are graph neural networks?," in *ICLR*. 2019, OpenReview.net.

[23] Ting Chen, Song Bian, and Yizhou Sun, "Are powerful graph neural nets necessary? A dissection on graph classification," *CoRR*, vol. abs/1905.04579, 2019.

[24] Saurabh Verma and Zhi-Li Zhang, "Graph capsule convolutional neural networks," *CoRR*, vol. abs/1805.08090, 2018.

[25] Zhang Xinyi and Lihui Chen, "Capsule graph neural network," in *ICLR*. 2019, OpenReview.net.

[26] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.