On Positional and Structural Node Features for Graph Neural Networks on Non-attributed Graphs

Hejie Cui[1,*], Zijie Lu[2,*], Pan Li[3], and Carl Yang[1,†]
[1]Department of Computer Science, Emory University
[2]Department of Computer Science, University of Illinois at Urbana-Champaign
[3]Department of Computer Science, Purdue University
hejie.cui@emory.edu, zijielu2@illinois.edu, panli@purdue.edu, j.carlyang@emory.edu

ABSTRACT

Graph neural networks (GNNs) have been widely used in various graph-related problems such as node classification and graph classification, where the superior performance is mainly established when natural node features are available. However, it is not well understood how GNNs work without natural node features, especially regarding the various ways to construct artificial ones. In this paper, we point out the two types of artificial node features, *i.e.*, *positional* and *structural* node features, and provide insights on why each of them is more appropriate for certain tasks, *i.e.*, *positional node classification*, *structural node classification*, and *graph classification*. Extensive experimental results on 10 benchmark datasets validate our insights, thus leading to a practical guideline on the choices between different artificial node features for GNNs on non-attributed graphs. The code is available at https://github.com/zjzijielu/gnn-exp/.

ACM Reference Format:
Hejie Cui, Zijie Lu, Pan Li, Carl Yang. 2021. On Positional and Structural Node Features for Graph Neural Networks on Non-attributed Graphs. In *Proceedings of Workshop of Deep Learning on Graphs: Methods and Applications, The 27th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (DLG-KDD'21).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Graphs provide a concise yet rich representation of data across different domains such as social networks, citation networks, gene-protein interactions, molecular structures and so on. How to effectively mine valuable information underneath graph data has become an appealing problem for data mining community. Recently, various kinds of powerful Graph Neural Networks (GNNs) demonstrate their privilege on common graph tasks such as node classification [20, 21], link prediction [50, 52] and graph classification [2, 39, 51]. GNNs combine node features and graph structures by aggregating node features through links into low-dimensional vector representations. Recently, considerable efforts have been put on studying the complicated contents of attributed networks, such as user attributes [44], node types [42], pooling layers [23, 27], design spaces [49], long-range dependency [12, 26], heterophily graph [54], subgraph contexts [41, 43, 45], robust graph representation [38], informativeness of nodes [18] and so on, where the superior performances are mainly established when natural node features (i.e., attributes) are available.

However, a great number of graphs in the wild do not contain node attributes [7, 10], which deteriorates the performance of GNNs [5, 11]. For example, in the molecules dataset QM9 [31, 33], a graph represents a molecule, i.e., nodes are atoms and edges are chemical bonds. For typical tasks on this dataset such as predicting the properties of molecules, i.e., toxicity or biological activity, GNNs cannot be directly applied under this situation due to the lack of natural node features [7, 37].

To apply GNNs on non-attributed graphs, several intuitive methods have been commonly practiced to initialize node features, such as degree-based [13], random [1, 34], one-hot [8], position-based [48], distance-based [24, 47] and so on. However, to the best of our knowledge, there exists no generic understanding or guideline towards the initialization of artificial node features based on the needs of downstream tasks. In this paper, we categorize common artificial node features and study their utility towards different types of graph mining tasks. From a high level, these intuitive node feature initialization methods can be grouped into two categories, positional and structural ones (Section 2). Take Figure 1 as an example. Positional features can help GNNs put node A and node B closer in the embedding space, whereas structural features facilitates putting node A and node C closer.

Extensive experiments are performed on 10 datasets with 8 common artificial features. Based on the information needs of different tasks, we further categorize them into multiple divisions, namely, positional node classification, structural node classification and graph classification (Section 3). Observations on the results validate our understanding that positional node features are more suitable for positional node classification, while structural node features benefit more for structural node classification and graph classification tasks. With appropriately designed artificial node features, the performance of GNNs can even surpass that with real features in some cases (Table 3). Besides, our proposed novel degree-based node feature initialization method, i.e., degree bucket range, achieves state-of-the-art performance on structural node classification (Section 2.3). We believe this empirical study on the selection of artificial node features can facilitate the understanding of feature initialization on non-attributed graphs and inspire new

[*]These two authors made equal contributions to this work.
[†]Corresponding Author.

DLG-KDD'21, August 14-18, 2021, Virtual Event

Figure 1: Illustration of Position vs. Structure: A and B are "positionally close"– having relatively close positions in the global network, whereas A and C are "structurally close"– having relatively similar local neighborhood structures.

designs of artificial node features, thus shedding light on various GNN applications on graphs in the wild.

2 TWO TYPES OF ARTIFICIAL NODE FEATURES

Several node feature initialization methods have been proposed for non-attributed graphs and commonly applied in various GNN models. We group these artificial node features into two main families: positional node features and structural node features.

2.1 Positional node features

Positional node features help GNNs capture node distance information regarding their relative positions in the graph. For example, in Figure 1, nodes A and B are positionally close. A real case is the publication network, where two authors who cite each other and also cite / get cited by similar other authors should be close considering their graph positions, and recognized as sharing similar research interests. Some intuitive positional node feature initialization methods include:

- *random*: A feature vector following random distribution is generated for each node, which is decided by the random seed in the data initialization. The random feature of each node varies among training runs with different random seeds initialization. This feature itself does not reflect relative positions, but it records a high-dimensional identity for each node, which can indirectly help GNNs learn the relative node positions.
- *one-hot*: A unique one-hot feature vector is initialized for each node [11, 48]. This feature is essentially equivalent to the random feature, when the parameters in the first linear layer of the GNN are randomly initialized.
- *eigen*: Eigen decomposition is performed on the normalized adjacency matrix and then the top k eigen vectors are used to generate a k-dimensional feature vector for each node [6, 19, 53], where the optimal value of k is decided by grid search [25].
- *deepwalk*: The initial feature of a node is generated based on the DeepWalk algorithm from [29] with the walk length set as 40 by default. Deep walk features with walk length longer than 2 can help to capture higher-order positional information in the graph.

Correspondingly, positional node classifications target at grouping nodes with respect to their positions, which corresponds to coarse global information in the graph. For example, in Figure 1, nodes A and B should be classified into the same class in the task of positional node classification. Specifically, *eigen* and *deepwalk* methods which generate features by matrix decomposition [30], are essentially dimension reduction, where the complex graph structures (i.e., adjacency matrices) information are embedded into a low dimensional representation. Therefore, *eigen* and *deepwalk* methods also incorporate structural information. However, as the features based on *eigen* and *deepwalk* reflect the position of nodes, with some abuse of terminology, we keep calling them positional features.

2.2 Structural node features

On the other hand, structural node features help GNNs capture structural information of nodes, such as degree information and neighborhood connection patterns. For example, in Figure 1, node A and C are similar regarding their neighborhood structures in the graph, though they are far away from each other in position. A real case is the molecular network, where two nodes with similar degrees and connection patterns should be put close considering their structures, and recognized as atoms with similar properties or functions. Some intuitive node feature initialization methods focusing on the structural aspects include:

- *shared*: An initial feature vector is shared across all nodes [11]. The shared feature vector used in the experiments is simply a vector of all 1's.
- *degree*: The degree value is converted to a one-hot degree vector for each node, where the vector dimension is selected based on the max degree of all nodes [13, 39].
- *pagerank*: The original PageRank score [4] of a given node is calculated and then flattened into a vector in order to fully utilize the embedding dimensions of neural networks, where the dimension of the extended vector is selected by grid-search [25]. It can be viewed as generalized higher-order node degree information.

Structural node classifications target at classifying nodes according to their structural patterns. For example, nodes A and C in Figure 1 should be put into the same class considering their similar "structural roles". Different from positional node features that characterize the position of nodes in a graph, structural node features target at representing node structural roles. Note that recent distance-based features [24, 47] also helps to learn node structural roles while GNNs that leverage distance-based features cannot make inference over multiple nodes in parallel, which increases the computational complexity. Interested readers may refer to the experiments in [46] to check how distance-based features help with learning node structural roles.

2.3 Byproduct: new SOTA for structural node classification

Motivated by our empirical studies on structural node features, we propose a novel node feature initialization method based on bucketing node degrees, which we name as *degree+*. Specifically, we divide degree values into several buckets, then map the degree values distributed in each bucket range into one class, and finally construct a unique one-hot vector for each class. Our proposed *degree+* feature can be regarded as an improved version of the original degree-based node feature, which better handles the sparse and skewed distribution of node degrees in the graph.

3 EXPERIMENTAL RESULTS

3.1 Basic settings

To conduct a fair and unbiased evaluation on the effectiveness of node features, we adopt the popular GNN of GraphSAGE [15] with

mean and *sum* aggregators for all the artificial node feature initialization methods across different types of graph mining tasks. Results with real features are also provided wherever natural node features are available. The train/test/validation split of each dataset follows the standard practice in the literature [11, 22, 32]. Graph level experiments are conducted with artificial features of sizes ranging from 100 to 500 with step 100. In addition, we perform comprehensive grid search for the best hyper-parameter settings including the learning rate, number of epochs and neighborhood sample size. The final performance of each feature initialization method is averaged over five runs under the optimal hyper-parameter settings.

3.2 Positional node classification

Definition and Datasets. The tasks of positional node classification target at predicting the "positional role" of each node [17, 36]. We consider three datasets for positional node classification tasks, including Cora [35], Citeseer [35] and Pubmed [28]. These three citation networks consist of scientific publications as nodes, which can be classified into several content categories. Edges connecting those nodes denote the citation relationships between publications. Real features for each publication node are included in these three datasets, which are bag-of-word vectors indicating the word presence in the text content. In these three datasets, since the publications are connected by citation links, the research topic based node classification tasks should be mainly driven by the positions of nodes in the graph. Performance with the real node feature is also presented as a baseline for comparison.

Protocols. We train and test the GraphSAGE model using the same data splits as in [22], namely 20 randomly-selected samples for each class during training with an additional validation set of 500 samples.

Performances. Experiment results of different node feature initialization methods on the three positional node classification datasets are presented in Table 1, where $\mathcal{P}$ and $\mathcal{S}$ indicate the *Type* of artificial node features, corresponding to *Positional* or *Structural* respectively. *Aggr.* denotes the aggregation method used in each GNN layer. Classification accuracy *Acc.(%)* is adopted here for evaluation and comparison.

Aggr.	Type	Feature	**Cora** *Acc.(%)*	**Pubmed** *Acc.(%)*	**Citeseer** *Acc.(%)*
Mean	$\mathcal{P}$	random	56.1±1.6	42.3±1.4	36.0±1.0
		one-hot	58.2±4.0	51.4±3.1	37.3±2.5
		eigen	73.2±2.3	70.0±4.8	42.9±2.3
		deepwalk	**75.3±1.0**	**74.0±2.6**	**46.8±0.9**
	$\mathcal{S}$	shared	17.9±0.0	38.6±0.0	20.2±0.0
		degree	37.4±2.1	41.1±2.9	36.0±1.3
		pagerank	25.2±2.4	39.8±1.9	20.5±3.4
		real feat.	80.2±1.1	79.0±2.2	68.0±4.0
Sum	$\mathcal{P}$	random	45.2±3.9	41.7±2.7	32.8±2.7
		one-hot	47.0±3.7	46.4±4.4	33.0±1.8
		eigen	70.5±5.1	68.8±4.1	40.1±5.0
		deepwalk	70.0±2.3	72.5±2.2	43.7±2.7
	$\mathcal{S}$	shared	17.1±5.2	33.3±6.4	22.3±4.6
		degree	50.7±3.7	42.6±1.8	32.0±3.5
		pagerank	27.8±4.4	33.0±6.3	23.4±1.3
		real feat.	70.5±3.7	75.4±3.7	59.3±4.0

Table 1: Positional node classification results

Observations.

- *Aggregation*: For positional node classification, *mean* aggregation shows better performance than sum aggregation. This is because mean aggregation can effectively filter out the influence of neighborhood size, which makes little contribution to and even impairs the performance on positional node classification tasks. However, *shared* feature plus *mean* aggregation gives the same embedding for every node, so the results are constantly poor with no variance.
- *Cross Feature Type Comparison*: For positional node classification tasks, most positional node feature initialization methods achieve much better performance than structural node feature ones. The advantage of position node features over structural node features is especially remarkable with *mean* aggregation.
- *Within Feature Type Comparison*: Among all positional node features: 1. *random* and *one-hot* initialization achieve comparable results. This is because they are essentially the same: after passing through the first layer of neural network where the parameters are randomly initialized, one-hot initialization is equivalent to random initialization except for possible differences in dimensions (e.g., on Pubmed). 2. among all positional features, *deepwalk* and *eigen* demonstrate the best performance across all the datasets, which owes to the higher-order positional information they can capture.

3.3 Structural node classification

Definition and Datasets. The tasks of structural node classification target at predicting the "structural role" of each node [14, 16, 17]. Here we choose three datasets for structural node classification, namely American air-traffic network, Brazilian air-traffic network and European air-traffic network [32]. Given an airport node in the air-traffic network, the target is to predict passenger flow level of that node solely based on the structure of air-traffic network. These three datasets are chosen because the node labels of them indicate the structural roles (vary in four levels from hubs to switches), rather than the traditional community identifiers of nodes [13, 22, 35].

Protocols. Following struc2vec [32], we use 80% of nodes for training. To highlight the performance of our novel *degree+* method, we adopt logistic regression with L2 regularization to train the classifier using the representation learned by struc2vec [32], which demonstrates SOTA results on these datasets.

Performances. Experiment results of different node initialization methods on structural node classification datasets are presented in Table 2. Note that there is no real feature in these datasets to compare with.

Observations.

- *Aggregation*: For structural node classification tasks, *sum* aggregation outperforms mean aggregation because it can capture the

Aggr.	Type	Initial.	**USA-air** *Acc.(%)*	**Brazil-air** *Acc.(%)*	**Europe-air** *Acc.(%)*
Mean	$\mathcal{P}$	random	59.3±1.8	45.7±5.9	44.9±5.8
		one-hot	59.2±2.6	48.6±7.4	44.0±0.7
		eigen	55.3±1.5	40.0±6.9	31.6±2.1
		deepwalk	58.1±2.8	42.1±9.6	41.5±3.3
	$\mathcal{S}$	shared	25.0±0.0	25.0±0.0	25.0±0.0
		degree	53.8±1.9	48.6±4.1	42.7±2.7
		degree+	59.2±2.7	60.0±3.0	50.6±3.9
		pagerank	39.7±2.9	47.9±7.4	25.9±0.0
Sum	$\mathcal{P}$	random	60.7±3.2	47.9±7.4	48.9±5.1
		one-hot	59.2±3.3	50.7±8.5	48.9±5.4
		eigen	67.8±2.5	57.8±5.3	49.4±4.5
		deepwalk	68.8±3.0	65.0±6.4	54.1±2.8
	$\mathcal{S}$	shared	55.7±2.0	61.4±4.7	45.4±1.0
		degree	63.6±3.0	70.0±4.1	58.0±3.6
		degree+	**69.1±2.6**	**76.4±4.1**	**61.2±3.8**
		pagerank	58.8±2.0	73.6±5.4	45.9±1.0
SOTA		struc2vec	63.8±1.6	73.6±9.6	58.8±3.0

Table 2: Structural node classification results

number of neighbors, which is an important structural feature in graphs.

- *Cross Feature Type Comparison*: For structural node classification tasks, in most cases structural node features demonstrate superiority compared with positional ones, and our proposed structural node feature *degree+* manifests the most distinct advantage over other positional features, reaching the new state-of-the-art.
- *Within Feature Type Comparison*: 1. among all four types of structural node features, *degree+* improves on *degree* by using a degree bucket, where nodes with degree values in a certain range are projected into one bucket. This alleviates the node degree sparsity and skewness problem. 2. *shared* can only capture the sizes of multi-hop neighborhoods, but loses track of specific neighborhood structures, thus performing rather poorly. 3. In contrast, *pagerank* can be viewed as a generalized higher-order node degree, and we conjecture that its performance deterioration arises from over-smoothing which in the worst cases renders it as similar to *shared*.

3.4 Graph classification

Definition and Datasets. For graph classification tasks, we consider two datasets with real node features, MUTAG [9] and PROTEINS [3] from chemical domain. In addition, we also consider two datasets without real node features, IMDB-BINARY and IMDB-MULTI [40] from the social domain.

Protocols. We take advantage of the GNN comparison framework proposed in [11]. On top of their experiment settings, we introduce the initialization methods, and use mean- and sum-pooling when applying GraphSAGE for graph classification.

Performances. Experiment results of different node initialization methods on graph classification datasets are presented in Table 3, where the *real feat.* is only available for MUTAG and PROTEINS.

Observations.

Aggr.	Typ.	Initial.	**MUTAG** *Acc.(%)*	**PROTEINS** *Acc.(%)*	**IMDB-B** *Acc.(%)*	**IMDB-M** *Acc.(%)*
Mean	$\mathcal{P}$	random	64.9±4.1	67.2±4.2	58.0±2.9	36.1±1.9
		one-hot	65.8±7.0	67.8±2.6	56.9±3.4	36.8±3.2
		eigen	63.8±2.1	60.4±1.0	50.2±1.3	33.4±0.7
		deepwalk	65.1±8.3	68.1±4.0	52.1±3.4	35.7±1.9
	$\mathcal{S}$	shared	66.7±0.0	59.6±0.0	50.0±0.0	33.3±0.0
		degree	**84.4±7.7**	69.5±2.6	69.7±5.1	45.1± 2.6
		pagerank	66.5±1.9	68.0±5.5	54.4±4.0	35.5±1.7
		real feat.	71.4±4.4	74.0±4.2	-	-
Sum	$\mathcal{P}$	random	66.9±7.1	67.5±4.1	54.0±3.6	36.2±2.1
		one-hot	65.1±3.8	66.8±3.8	52.8±2.7	33.4±2.6
		eigen	65.4±7.7	69.0±4.1	69.3±4.6	42.4±3.4
		deepwalk	64.2±8.6	66.2±4.2	51.9±2.8	35.3±3.0
	$\mathcal{S}$	shared	79.9±6.7	69.1±4.5	67.9±2.8	43.3±4.6
		degree	84.0±8.4	69.3±3.3	68.9±2.5	44.9±4.1
		pagerank	77.3±7.6	**69.9±3.1**	**70.3±2.9**	**48.2±3.2**
		real feat.	83.0±6.3	73.8±2.6	-	-

Table 3: Graph classification results

- *Aggregation*: Similar to structural classification tasks, *sum* aggregation outperforms mean aggregation on graph classification tasks, since the number of neighbors contributes as an important type of structural information for graph classification tasks.
- *Cross Feature Type Comparison*: For graph classification, though the best performance is not consistently achieved on a particular feature initialization method across four datasets, it always falls in the category of structural node features. This is because we do not care about positional features such as the specific position of each node in graph classification tasks. Instead, similar to structural node classification task, the overall structural information of the graph plays the important role.
- *Within Feature Type Comparison*: 1. Among the structural node features, *pagerank* demonstrates better performance in most of the cases. 2. Impressively, the performances of GNN on *degree* on MUATG and *pagerank* on PROTEIN with the *sum* aggregator even surpass those with real features. This further demonstrates the importance of choosing the appropriate artificial node features, sometimes even when natural node features are available.

4 CONCLUSION

Graphs in the real world do not always have natural node features available, due to the lack of task-specific node attributes, privacy concerns and/or difficulties in data collection. In this paper, we study the usage of artificial node features when applying GNNs on non-attributed graphs. We categorize commonly used artificial node features into two groups, positional node features and structural node features, based on what kind of information they can help GNNs capture. Extensive empirical experiments are conducted across three graph mining tasks, positional node classification, structural node classification and graph classification. The results validate our insights that positional node features are more suitable for positional node classification, while structural node features benefit more for structural node classification and graph classification tasks. We hope our empirical study can provide a generic and practical guideline for choosing the appropriate artificial node features and exploring more useful artificial features based on the needs of downstream tasks.

REFERENCES

[1] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. 2020. The Surprising Power of Graph Neural Networks with Random Node Initialization. *arXiv preprint arXiv:2010.01179* (2020).

[2] Davide Bacciu, Federico Errica, and Alessio Micheli. 2018. Contextual graph markov model: A deep and generative approach to graph processing. In *ICML*.

[3] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21 (2005), i47–i56.

[4] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput. Networks* 30 (1998), 107–117.

[5] Chen Cai and Yusu Wang. 2019. A simple yet effective baseline for non-attribute graph classification. *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).

[6] Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. 2012. Spectral clustering of graphs with general degrees in the extended planted partition model. In *Conference on Learning Theory*.

[7] Xu Chen, Siheng Chen, Jiangchao Yao, Huangjie Zheng, Ya Zhang, and Ivor W Tsang. 2020. Learning on Attribute-Missing Graphs. *IEEE transactions on pattern analysis and machine intelligence* (2020).

[8] Zhengdao Chen, Lisha Li, and Joan Bruna. 2018. Supervised Community Detection with Line Graph Neural Networks. In *ICLR*.

[9] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34 (1991), 786–797.

[10] Chi Thang Duong, Thanh Dat Hoang, Ha The Hien Dang, Quoc Viet Hung Nguyen, and Karl Aberer. 2019. On Node Features for Graph Neural Networks. *CoRR* (2019).

[11] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A fair comparison of graph neural networks for graph classification. In *ICLR*.

[12] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. 2020. Implicit Graph Neural Networks. In *NeurIPS*.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin* 40 (2017), 52–74.

[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

[16] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *SIGKDD*.

[17] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *SIGKDD*.

[18] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph Policy Network for Transferable Active Learning on Graphs. In *NeurIPS*.

[19] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. 2020. Combining Label Propagation and Simple Models Out-performs Graph Neural Networks. *CoRR* abs/2010.13993 (2020).

[20] Mohammad Rasool Izadi, Yihao Fang, Robert Stevenson, and Lizhen Lin. 2020. Optimization of Graph Neural Networks with Natural Gradient Descent. In *IEEE BigData*. 171–179.

[21] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*.

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[23] Maosen Li, Siheng Chen, Ya Zhang, and Ivor W. Tsang. 2020. Graph Cross Networks with Vertex Infomax Pooling. In *NeurIPS*.

[24] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. *NeurIPS* (2020).

[25] Gongxu Luo, Jianxin Li, Hao Peng, Carl Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. Graph Entropy Guided Node Embedding Dimension Selection for Graph Neural Networks. *CoRR* abs/2105.03178 (2021).

[26] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. 2020. Path Integral Based Convolution and Pooling for Graph Neural Networks. In *NeurIPS*.

[27] Diego P. P. Mesquita, Amauri H. Souza Jr., and Samuel Kaski. 2020. Rethinking pooling in graph neural networks. In *NeurIPS*.

[28] Galileo Mark Namata, Ben London, Lise Getoor, and Bert Huang. 2012. Query-driven Active Surveying for Collective Classification. In *Workshop on Mining and Learning with Graphs*.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

[30] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *WSDM*.

[31] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data* 1 (2014).

[32] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *SIGKDD*.

[33] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. 2012. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J. Chem. Inf. Model.* 52 (2012), 2864–2875.

[34] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. 2021. Random features strengthen graph neural networks. In *SIAM International Conference on Data Mining (SDM)*.

[35] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29 (2008), 93–93.

[36] Balasubramaniam Srinivasan and Bruno Ribeiro. 2020. On the Equivalence between Positional Node Embeddings and Structural Graph Representations. In *ICLR*.

[37] Hibiki Taguchi, Xin Liu, and Tsuyoshi Murata. 2021. Graph convolutional networks for graphs containing missing features. *Future Gener. Comput. Syst.* 117 (2021), 155–168.

[38] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph Information Bottleneck. In *NeurIPS*.

[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[40] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *SIGKDD*.

[41] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *KDD*.

[42] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *TKDE* (2020).

[43] Carl Yang, Jieyu Zhang, and Jiawei Han. 2020. Co-Embedding Network Nodes and Hierarchical Labels with Taxonomy Based Generative Adversarial Networks. In *ICDM*.

[44] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. 2020. Relation Learning on Social Networks with Multi-Modal Graph Edge Variational Autoencoders. In *WSDM*.

[45] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. 2019. Conditional Structure Generation through Graph Variational Generative Adversarial Nets. In *NeurIPS*.

[46] Haoteng Yin, Yanbang Wang, and Pan Li. 2020. Revisit graph neural networks and distance encoding in a practical view. *arXiv preprint arXiv:2011.12228* (2020).

[47] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-Aware Graph Neural Networks. In *AAAI*.

[48] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *ICML*.

[49] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design Space for Graph Neural Networks. In *NeurIPS*.

[50] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *NeurIPS*.

[51] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*.

[52] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2020. Revisiting Graph Neural Networks for Link Prediction. *arXiv preprint arXiv:2010.16103* (2020).

[53] Yilin Zhang and Karl Rohe. 2018. Understanding Regularized Spectral Clustering via Graph Conductance. In *NeurIPS*.

[54] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.