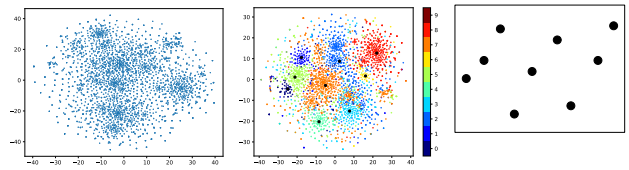
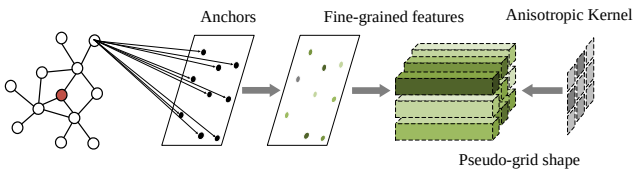
The Sumpplementary material of Graph Deformer Network

Wenting Zhao , **Yuan Fang** , **Zhen Cui*** , **Tong Zhang** and **Jian Yang**
Key Lab of Intelligent Perception and Systems for
High-Dimensional Information of Ministry of Education,
Jiangsu Key Lab of Image and Video Understanding for Social Security,
School of Computer Science and Engineering, Nanjing University of Science and Technology
{wtingzhao, fangyuan, zhen.cui, tong.zhang, csjyang}@njust.edu.cn

A Appendix



(a) Original data (b) K-means Cluster (c) Cluster centers



(d) The procress of projection and convolution on one node

Figure 1: A toy example: visualization of (a) original data and (b) its K-means cluster for Cora dataset. T-SNE is used to project high-dimensional data into 2-D space. (c) 9 cluster centers can be regarded as a pseudo-grid in the 2-D space. Next, (d) a neighbor is transformed into the anchor space, and anisotropic convolution is implemented.

Fig. 1 shows the main idea of Graph Deformer Network, K-means Cluster on the original data produces 9 cluster centers, pseudo-grid shape, indicating several implicit directions like 3×3 receptive field (top left, top right, etc.) in images. Initially anchor nodes are generated by imposing an non-linear transformation on cluster centers in order to match the updated features. Then, the irregular neighbors are deformed into the anchor space. In this deformer process, these irregular local neighborhoods are successfully transformed into a unified space without loss of information, instead, this way refines the features of each neighbor into 9 directions represented by anchor nodes. Finally anisotropic convolution can be implemented by using different filters on these fine-grained features. The anisotopic convolution realizes weight sharing for different local neighborhoods, which not only maintains computationl efficiency, but also captures some common property and helps improve expressive power of model.

*Contact Author

A.1 Proof of Proposition 1

(a) Mean aggregator fails (b) Sum aggregator fails

Figure 2: Examples of graph structures on which mean and sum aggregators fail. The neighborhood regions of v and v' have the same representation even though their graph structures are different.

Proof. Let us first illustrate the cases that cannot be distinguished by mean/sum aggregation. There exist two non-isomorphic graphs whose response outputs are consistent on mean and sum aggregation. As shown in Fig. 2, the gray node is a reference node, denoting features of the green and yellow nodes as $\mathbf{x}_g$, $\mathbf{x}_y$. Then, (a) $\frac{1}{4}(2\mathbf{x}_g + 2\mathbf{x}_y) = \frac{1}{2}(\mathbf{x}_g + \mathbf{x}_y)$, and the mean aggregator cannot distinguish them. (b) $(3\mathbf{x}_g + \mathbf{x}_y) = (\mathbf{x}_g + 2\mathbf{x}_y)$ if $2\mathbf{x}_g = \mathbf{x}_y$, and the sum aggregator cannot distinguish them. In practice, the sum aggregator is relatively better than the mean aggregator because the sum operation reflects the accumulation number (i.e., node degree).

Next, we will prove the proposed graph deformer process (not including the anisotropic convolution part) can distinguish these two cases. Projecting the different neighborhoods into anchor space can keep the energy constant, i.e., the sum of neighbors in the local neighborhood is the same as the sum on each anchor node after deforming.

(1) Thus, for two non-isomorphic graphs, if their sum on a local neighborhood is different such as Fig. 2(a), then it must be different after deforming into anchor space.

(2) If two non-isomorphic graphs have the same sum on the local neighborhood as shown in Fig. 2(b), then the sum is still same after deforming. But their representations corresponding to the same anchor node are different with a high probability. If two local neighborhoods have the same representation on each anchor after being deformed, taking Fig. 2(b) as an example, setting $m = 3$, i.e., 3 anchor nodes, then we have

$$\alpha_{0,0}\mathbf{x}_0 + \alpha_{1,0}\mathbf{x}_1 + \alpha_{2,0}\mathbf{x}_2 + \alpha_{3,0}\mathbf{x}_3 + \alpha_{4,0}\mathbf{x}_4$$

$$
\begin{aligned}
&= \alpha_{5,0}\mathbf{x}_5 + \alpha_{6,0}\mathbf{x}_6 + \alpha_{7,0}\mathbf{x}_7 + \alpha_{8,0}\mathbf{x}_8,\\
&\alpha_{0,1}\mathbf{x}_0 + \alpha_{1,1}\mathbf{x}_1 + \alpha_{2,1}\mathbf{x}_2 + \alpha_{3,1}\mathbf{x}_3 + \alpha_{4,1}\mathbf{x}_4\\
&= \alpha_{5,1}\mathbf{x}_5 + \alpha_{6,1}\mathbf{x}_6 + \alpha_{7,1}\mathbf{x}_7 + \alpha_{8,1}\mathbf{x}_8,\\
&\alpha_{0,2}\mathbf{x}_0 + \alpha_{1,2}\mathbf{x}_1 + \alpha_{2,2}\mathbf{x}_2 + \alpha_{3,2}\mathbf{x}_3 + \alpha_{4,2}\mathbf{x}_4\\
&= \alpha_{5,2}\mathbf{x}_5 + \alpha_{6,2}\mathbf{x}_6 + \alpha_{7,2}\mathbf{x}_7 + \alpha_{8,2}\mathbf{x}_8,\\
\text{s.t.}\ &\begin{cases}
\mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4 + \mathbf{x}_5 = \mathbf{x}_6 + \mathbf{x}_7 + \mathbf{x}_8,\\
\alpha_{j,k} = \dfrac{\exp(\langle \mathbf{x}_j, \overline{\mathbf{a}}_k\rangle/\sqrt{d})}{\sum_{k'}\exp(\langle \mathbf{x}_j, \overline{\mathbf{a}}_{k'}\rangle/\sqrt{d})}, \quad k' = 0, 1, \cdots, m-1,
\end{cases}
\end{aligned}
\tag{1}
$$

$$
\Longrightarrow
$$

$$
\begin{aligned}
&\beta_1 = \alpha_{0,0} = \alpha_{1,0} = \alpha_{2,0} = \cdots = \alpha_{7,0} = \alpha_{8,0},\\
&\beta_2 = \alpha_{0,1} = \alpha_{1,1} = \alpha_{2,1} = \cdots = \alpha_{7,1} = \alpha_{8,1},\\
&\beta_3 = \alpha_{0,2} = \alpha_{1,2} = \alpha_{2,2} = \cdots = \alpha_{7,2} = \alpha_{8,2},\\
&\beta_1 + \beta_2 + \beta_2 = 1,\\
&\beta_i = \frac{\exp(\langle \mathbf{x}_j, \overline{\mathbf{a}}_k\rangle/\sqrt{d})}{\sum_{k'}\exp(\langle \mathbf{x}_j, \overline{\mathbf{a}}_{k'}\rangle/\sqrt{d})}, \quad k' = 0, 1, \cdots, m-1.
\end{aligned}
\tag{2}
$$

Because these two graphs are non-isomorphic but the sum is the same, then the features $\mathbf{x}_i$ is different. Moreover, the derived normalized attention score for all nodes should be the same. Thus, the inner product between these nodes and anchors should satisfy,

$$
\begin{aligned}
&\langle \mathbf{x}_0, \overline{\mathbf{a}}_k\rangle = \langle \mathbf{x}_1, \overline{\mathbf{a}}_k\rangle + C_1 = \cdots = \langle \mathbf{x}_8, \overline{\mathbf{a}}_k\rangle + C_8,\\
&k' = 0, 1, \cdots, m-1,
\end{aligned}
\tag{3}
$$

which means that the inner product between any two nodes and all anchor nodes differs by a same constant C_i. In this case, it is equivalent to a linear normalization of the inner product. Both $\mathbf{x}_i$ and $\overline{\mathbf{a}}_{k'}$ are optimized by the neural network. And the formula of Eqn. (3) difficultly holds. Thus, the proposed graph deformer process usually is an injective, stronger than sum/mean aggregation. □

A.2 Proof of Proposition 2

Before giving the Proof of Proposition 1, we first simply elaborate on the Weisfeiler-Lehman (WL) graph isomorphism test.

WL test [Shervashidze *et al.*, 2011] is a state-of-the-art graph kernel method, which concatenates each vertex of a labeled graph and its neighbors to create a sorted multiset label, and then assigns a new label/color to this multiset by hashing. Vertices with the same multiset are assigned the same label. This can be formulated as

$$
c^{(t)}(v) = \text{HASH}\Big(\big(c^{(t-1)}(v), \{\!\{c^{(t-1)}(u) | u \in \mathcal{N}_v^1\}\!\}\big)\Big), \tag{4}
$$

where the $c^{(t)}(v)$ means the color/label of node v in iteration t, $\mathcal{N}_v^1$ is the set including the 1-hop neighbors of node v, and $\{\!\{\cdot\}\!\}$ is a multiset.

Proof. The graph deformer process has been analyzed in A.1, which usually is injective. For a single-scale neighborhood, the anisotropic convolution described in Eqn.(9) is equal to the next form

$$
\widetilde{\mathbf{x}}_{v_r} = \text{MLP}\left([\widetilde{\mathbf{u}}_0, \cdots, \widetilde{\mathbf{u}}_{m-1}]\right), \tag{5}
$$

which is a combination of concatenation on anchor nodes and a multilayer perceptron.

For a multiset $\{\!\{\widetilde{\mathbf{u}}_i\}\!\}$, $i = 0, 1, \cdots, m-1$, where $\widetilde{\mathbf{u}}_i$ represents a d-dimensional vector. There exists some order making the concatenation on the $\{\!\{\widetilde{\mathbf{u}}_i\}\!\}$ unique. That means the concatenation can obtain different results for different local neighborhoods. Thus, the concatenation aggregation can satisfy an injective, which also is implemented in the WL test. Additionally, according to the universal approximation theorem [Hornik, 1991], multilayer feedforward networks are, under very general conditions on the hidden unit activation function, universal approximators provided that sufficiently many hidden units are available. Thus, we can reach the conclusion.

□

A.3 Datasets

The global properties of datasets for node and graph classifications have been summarized in Table 1 and Table 2, and details are as follows:

Table 1: Summary of graph datasets for node classification.

Dataset	Nodes	Edges	Features	Classes	Label rate
Cora	2708	5429	1433	7	0.052
Citeseer	3327	4732	3703	6	0.036
Pubmed	19717	44338	500	3	0.003

- Citation graph. The Cora dataset is constructed by 2708 machine learning papers belonging to 7 classes. Each node represents a paper, and two papers are connected if one cites another paper. There are a total of 5429 edges. Node features are bag-of-words representation indicating the absence/presence of the corresponding word from the dictionary that consists of 1433 unique words. Similarly, Citeseer contains 3327 papers divided into 6 classes. Node features are also bag-of-words representation with 3037 unique words. There exist 4732 edges between the nodes. Pubmed is a larger dataset containing 19717 papers and 44338 edges. The node features are real-valued entries indicating Term Frequency-Inverse Document Frequency (TF-IDF) of the corresponding word from a dictionary.
- Bioinformatics datasets. MUTAG [Debnath *et al.*, 1991] is a nitro compounds dataset including 188 samples and is divided into 2 classes. PTC [Toivonen *et al.*, 2003] consists of compounds labeled according to carcinogenicity on rodents with 19 node labels. NCI1 [Wale *et al.*, 2008] is a balanced dataset of chemical compounds collected by the National Cancer Institute (NCI), which contains 4110 chemical compounds with 37 discrete node labels about the human tumor cell. ENZYMES [Borgwardt *et al.*, 2005] consists of 600 protein tertiary structures divided into 6 classes and obtained from the

Table 2: Summary of graph datasets for graph classification.

Dataset	Graphs	Classes	Node labels	Max nodes	Avg.nodes	Avg.edges
MUTAG	188	2	7	28	17.93	19.79
PTC	344	2	19	109	25.56	14.69
NCI1	4110	2	37	111	29.87	32.3
ENZYMES	600	6	3	126	32.63	62.14
PROTEINS	1113	2	3	620	39.06	72.82
IMDB-BINARY	1000	2	-	136	19.77	96.53
IMDB-MULTI	1500	3	-	89	13.0	65.94
REDDIT-MULTI-12K	11929	11	-	3782	391	456

BRENDA enzyme database. PROTEINS [Borgwardt *et al.*, 2005] consists of 1113 proteins in which nodes are secondary structure elements (SSEs). There exists an edge between two nodes if they are contiguous in the amino acid sequence.

- Social network datasets. IMDB-BINARY and IMDB-MULTI are movie collaboration datasets derived from the work [Yanardag and Vishwanathan, 2015], where every graph belongs to a kind of movie genre which is also to be predicted. The IMDB-BINARY dataset contains two types: Action and Romance. While the IMDB-MULTI dataset is constructed from Comedy, Romance, and Sci-Fi genres. Each node represents an actor/actress. If two actors appear in the same movie, then they are connected by an edge. REDDIT-MULTI-12K is a social interaction dataset, where graphs represent online discussion threads crawled from Reddit. Each node represents a user, and two users are connected if one of them responded to at least one comment from the other. The task is to identify the graph belongs to which question/answer-based community.

A.4 Related Work

In this section, we briefly retrospect the graph convolutional neural networks related to our work. Recently, numerous graph convolution methods accrue in the field of artificial intelligence. These works roughly fall into two categories: spectral based methods [Bruna *et al.*, 2014; Susnjara *et al.*, 2015; Defferrard *et al.*, 2016; Levie *et al.*, 2018] and spatial based methods [Gilmer *et al.*, 2017; Such *et al.*, 2017; Simonovsky and Komodakis, 2017; Gao *et al.*, 2018; Huang *et al.*, 2018; Liu *et al.*, 2019]. Based on the Spectral Graph Theory [Chung and Graham, 1997], the work [Shuman *et al.*, 2013] presents a basic framework to process graph data via filtering. Bruna et al. firstly generalized convolutional neural networks to graphs through the decomposition of the graph Laplacian matrix [Bruna *et al.*, 2014]. The work [Henaff *et al.*, 2015] seeks to express spatial localization of filters with smoothing coefficients. Then, ChebyNet [Defferrard *et al.*, 2016] and GCN [Kipf and Welling, 2017] take advantage of recursive Chebyshev polynomials to approximate parameterized filters, where the computing efficiency is significantly improved. Ever since, increasing work [Li *et al.*, 2018a; 2019; Chen *et al.*, 2018; Zhuang and Ma, 2018] is dedicated to designing, improving, and optimizing convolution operators on graphs. On the other hand, the spatial based approaches perform convolution directly on graphs by aggregating node features over a spatial neighborhood. DCNN [Atwood and Towsley, 2016] proposes a diffusion-convolution method. Niepert et al. [Niepert *et al.*, 2016] normalized a graph to a grid-shaped structure and performed traditional convolution operations. DGCNN [Wu *et al.*, 2018] adds a disordered graph convolutional layer(DGCL) to avoid the loss of information. GraphSAGE [Hamilton *et al.*, 2017] generates embeddings by sampling and aggregating node features in a local neighborhood. Velickovic et al. [Velickovic *et al.*, 2018] adopted an attention mechanism into graph learning. MoNet [Monti *et al.*, 2017] utilized a Gaussian mixture model to encode the weights between nodes and aggregate node features. JK-Net [Xu *et al.*, 2018] aggregates features of different layers by max-pooling, concatenation, or LSTM-attention. GIN [Xu *et al.*, 2019] presents a theoretical framework to analyze the expressive power of GNNs. GIC method [Jiang *et al.*, 2019] attempts to discover the direction of variations by Gaussian mixture models.

In contrast to previous methods [Kipf and Welling, 2017; Xu *et al.*, 2019; Zhuang and Ma, 2018], etc., the way of aggregation is obviously different. GCNs usually bypass the irregularity of graphs by utilizing a weighted-sum aggregation over the neighborhoods, which is an isotropic filter. Our GDN instead firstly transforms the local neighborhoods into a regular anchor space and then performs anisotropic filters on the regular anchor space.

In contrast to Transformer (multi-head attention mechanism) [Vaswani *et al.*, 2017] and its a generalization, graph attention network [Velickovic *et al.*, 2018], we give the following differences. In the Transformer, all center nodes are anchor nodes, and the attention coefficient is computed between each central node and its neighbor nodes. In our GDN, several anchor nodes are initially generated by K-means Cluster from global nodes which implicitly represents several different directions like a $k \times k$ patch (upper left, upper right, etc.) in images, and the attention coefficient is computed between local neighbors and anchor nodes. Because the anchor nodes are generated from the global graph, all local neighborhoods are projected into a common anchor space, some common property for all local neighborhood can be captured by imposing an anisotropic filter on anchors. In contrast, though multi-head mechanism is used, the Transformer is still locally aggregated. Another weakness of the Transformer is computationally intensive, especially for large graphs.

Our proposed GDN is different form other peer works.

P-GNN [You *et al.*, 2019] selects several anchor nodes as position characterization, and concatenates the feature of node with features of anchor nodes, then mean aggregation is employed on different anchor sets followed by a fully-connected transform, which is an isotropic filter. LGCN [Gao *et al.*, 2018] performs feature selection by sorting the top k-largest values on each feature dimension, and produces k new nodes (fixed size) for the next filtering process. Caps-GNN [Xinyi and Chen, 2018] presents a capsule graph network to learn graph capsules of different aspects from node capsules by utilizing an attention module and routing mechanism. In contrast, our proposed method transforms local neighborhood into an anchor space spanned by several anchor nodes, the filtering is operated in the anchor space. Transforming irregular structures into a regular space also be studied in 3D domain. PointCNN [Li *et al.*, 2018b] learns an X-transformation from the input points and then applies convolution operator on the X-transformed features. KP-Conv [Thomas *et al.*, 2019] takes radius neighborhoods as input and processes them with weights spatially located by a small set of kernel points. In essence, PointCNN [Li *et al.*, 2018b] leverages the self-attention mechanism to produce fixed-size output for different-size local neighbor regions. KPConv [Thomas *et al.*, 2019] projects neighbor 3D points into 3D kernel points, and this projection is only operated in 3D point space. In contrast, our proposed method focus on the more general graph domain.

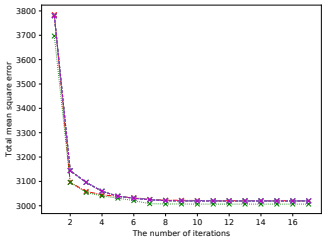
A.5 Other Experimental results

Empirical comparison, time cost. We further compare the time cost between our GDN and widely-used GCN and GAT in Table 3. As it is shown, our GDN's time cost is less than GAT.

Table 3: Comparison of running time over testing

Method	Cora	Citeseer	Pubmed
GCN	0.011	0.021	0.116
GAT	0.038	0.155	OOM
GDN(2L)	0.037	0.098	0.19

The influence of random sampling anchors to clustering convergence. We additionally visualize the convergence curves of the kmeans clustering under different random samplings in Figure 3. As it is shown, the clustering can always converge.



(a) Cora

Figure 3: The convergence of kmeans clustering under different node samplings.

A.6 The Notations & GDN Algorithm

We summarize the used notations of our paper as Table 4 and algotithm description as Alg. 1.

Table 4: The notations used in the paper.

Notation	Representation
$\mathcal{G}$	graph
$\mathcal{V}$	vertex set
$\mathcal{E}$	edge set
$\mathbf{A}, \mathbf{A}^{'}$	adjacency matrix
$\mathbf{X}, \mathbf{X}^{'}$	feature matrix
$\mathbf{x}_i, \mathbf{X}_i$	feature vector of node in graph
$\mathcal{N}_{v_i}^{s}$	set of s-hop neighbors for node v_i
$\overline{\mathbb{V}}$	anchor set
$\overline{\mathbf{v}}$	one anchor node
$\overline{\mathbf{x}}$	feature vector of anchor node
f	filters
$\mathcal{D}$	deformer function
$\mathcal{C}$	anisotropic convolution operator
$\mathcal{K}$	convolution kernel
$\mathcal{P}$	plooing / graph coarsening operator
$\mathbf{F}^{(r)}$	deformed multi-granularity feature matrix for node v_r
$\mathcal{V}_{\text{sampling}}$	sampled node set from graph
$\overline{\mathbf{a}}_k$	feature vector of anchor node k after updating
$\mathbf{q}$.	anchor-related feature vector (query feature vector)
$\mathbf{u}$.	value feature vector
$\widetilde{\mathbf{u}}_k$	deformed feature vector on anchor node k
α	weight matrix
$\widetilde{\mathbf{x}}_{v_r}$	output of graph deformer convolution for node v_r
$\mathbf{Z}$	binary cluster matrix for graph coarsening

Algorithm 1 Graph Deformer Algorithm

Input: node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$; adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$; the scale of neighborhoods S; the number of key nodes m; the deformer function $\mathcal{D}$; the deformer parameter Θ; the convolution operation $\mathcal{C}$; the convolution kernel $\mathcal{K}$; fully-connected function g;.
Output: node embedding $\mathbf{z}$.

$\overline{\mathcal{V}}^0 = \{(\overline{\mathbf{v}}_k^0, \overline{\mathbf{x}}_k^0)\}|_{k=0}^{m-1} \leftarrow$ Clustering $\{(v_i, \mathbf{x}_i)|v_i \in \mathcal{V}_{\text{sampling}}\}$;
$\overline{\mathbf{x}}_k \leftarrow \text{ReLU}(\mathbf{W}_A \overline{\mathbf{x}}_k^0 + \mathbf{b}_A), k = 1, \cdots, m$;
for $v_r \in \mathcal{V}$ **do**
  Initialized input feature $\mathbf{x}_r$
  for $s = 1, \cdots, S$ **do**
    $\mathbf{F}^{(r,s)} \leftarrow \sum_{v_t \in \mathcal{N}_{v_r}^s} \mathcal{D}_{v_t \to \overline{\mathbf{v}}_i}(\overline{\mathbf{x}}_i, \mathbf{x}_t, \Theta)$;
    $\widetilde{\mathbf{x}}_r^{(s)} \leftarrow \mathcal{C}(\mathbf{F}^{(r,s)}, \mathcal{K}^l)$;
  end for
  $\widetilde{\mathbf{x}}_r \leftarrow [\mathbf{x}_r; \widetilde{\mathbf{x}}_r^{(1)}; \cdots; \widetilde{\mathbf{x}}_r^{(S)}]$;
  $\mathbf{z}_r = g(\widetilde{\mathbf{x}}_r)$;
end for

References

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NeurIPS*, pages 1993–2001, 2016.

[Borgwardt *et al.*, 2005] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[Bruna *et al.*, 2014] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *ICLR*, 2014.

[Chen *et al.*, 2018] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

[Chung and Graham, 1997] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. 1997.

[Debnath *et al.*, 1991] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pages 3844–3852, 2016.

[Gao *et al.*, 2018] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *SIGKDD*, pages 1416–1424, 2018.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, volume 70, pages 1263–1272, 2017.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034, 2017.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Hornik, 1991] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[Huang *et al.*, 2018] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *NeurIPS*, pages 4558–4567, 2018.

[Jiang *et al.*, 2019] Jiatao Jiang, Zhen Cui, Chunyan Xu, and Jian Yang. Gaussian-induced convolution for graphs. In *AAAI*, 2019.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Levie *et al.*, 2018] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[Li *et al.*, 2018a] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

[Li *et al.*, 2018b] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, pages 820–830, 2018.

[Li *et al.*, 2019] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label efficient semi-supervised learning via graph filtering. In *CVPR*, 2019.

[Liu *et al.*, 2019] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, volume 33, pages 4424–4431, 2019.

[Monti *et al.*, 2017] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

[Niepert *et al.*, 2016] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, pages 2014–2023, 2016.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(23):2539–2561, 2011.

[Shuman *et al.*, 2013] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[Simonovsky and Komodakis, 2017] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, pages 3693–3702, 2017.

[Such *et al.*, 2017] Felipe Petroski Such, Shagan Sah, Miguel Alexander Dominguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D Cahill, and Raymond Ptucha. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):884–896, 2017.

[Susnjara *et al.*, 2015] Ana Susnjara, Nathanael Perraudin, Daniel Kressner, and Pierre Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

[Thomas *et al.*, 2019] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J Guibas. Kpconv: Flexible and

deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[Toivonen *et al.*, 2003] Hannu Toivonen, Ashwin Srinivasan, Ross D King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, 19(10):1183–1193, 2003.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[Wale *et al.*, 2008] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[Wu *et al.*, 2018] Bo Wu, Yang Liu, Bo Lang, and Lei Huang. Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model. *Neurocomputing*, 321:346–356, 2018.

[Xinyi and Chen, 2018] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *ICLR*, 2018.

[Xu *et al.*, 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5453–5462, 2018.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[Yanardag and Vishwanathan, 2015] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *SIGKDD*, pages 1365–1374, 2015.

[You *et al.*, 2019] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *ICML*, pages 7134–7143, 2019.

[Zhuang and Ma, 2018] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*, pages 499–508, 2018.