1. ERRATA

- In the section 5.1, Baselines and experimental setting, of the main paper, I wrote that "We set the k of SortPooling such that 60% graphs have nodes more than k.", which should be "... such that 60% graphs have nodes **less** than k.". So is the the section 5.2, Baselines and experimental setting, "we set the k of SortPooling such that 90% graphs have nodes more than k" should be changed to "we set the k of SortPooling such that 90% graphs have nodes **less** than k".

2. CONNECTION WITH SPECTRAL GRAPH CONVOLUTION

The proposed graph convolution model in this paper also has a spectral formulation. A **spectral convolution** of a filter $g_\theta(\Lambda)$ with a graph signal $\mathbf{x} \in \mathbb{R}^n$ is defined as:

$$g_\theta(\Lambda) * \mathbf{x} = \mathbf{U} g_\theta(\Lambda) \mathbf{U}^\top \mathbf{x}, \tag{S1}$$

where $\mathbf{U}$ is the matrix of eigenvectors of the graph's normalized Laplacian $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$, $\Lambda = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{n-1})$ is the diagonal eigenvalue matrix of $\mathbf{L}$. According to [1], $\mathbf{U}^\top \mathbf{x}$ is defined as the graph Fourier transform of $\mathbf{x}$ to the frequency modes $\lambda_0, \ldots, \lambda_{n-1}$ by a set of orthonormal eigenvectors in $\mathbf{U}$. Thus, (S1) can be understood as first transforming $\mathbf{x}$ into the spectral domain by $\mathbf{U}^\top \mathbf{x}$, and then applying a filter g_θ on the frequency modes, followed by an inverse Fourier transform by left multiplying $\mathbf{U}$. The symmetrically normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is conventionally used, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the degree matrix of the graph.

Since a diagonal filter $g_\theta = \text{diag}(\theta)$ whose parameters are all free has $\mathcal{O}(n)$ complexity which is the size of the entire graph, and is not spatially localized, there are various approximations to (S1) by reducing the number of parameters and restricting the convolution to be spatially localized [2–4]. [4] proposed a renormalization trick defined as follows:

$$g_{\mathbf{W}} * \mathbf{X} := \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X}\mathbf{W}, \tag{S2}$$

where $\mathbf{X} \in \mathbb{R}^{n \times c}$ generalizes $\mathbf{x}$ to multiple channels, $\mathbf{W} \in \mathbb{R}^{c \times c'}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{D}}$ have the same definitions as before. (S2) is shown to be a first-order approximation of the polynomially parameterized spectral graph convolution [1]. In (S2), filters are localized to be within nodes' 1-hop neighbors. In addition, it has only one parameter to learn for each ⟨input channel, output channel⟩ pair, largely reducing the number of parameters. It is shown that stacking multiple such first-order approximations outperforms the original polynomial version on a node classification task.

Our proposed graph convolution is similar to (S2) except for using a random-walk normalized propagation model instead of the symmetrical one. Recall that $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ is exactly the transition matrix of a random walk on the self-loop-included graph. In fact, we can show that (S1) is also a first-order approximation of the polynomially parameterized spectral graph convolution, given that we generalize graph Fourier transform to the random-walk normalized Laplacian $\mathbf{L}^{\text{rw}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$. Note that, however, the generalized graph Fourier basis is no longer orthogonal due to the nonsymmetric Laplacian matrix.

To show the differences between our graph convolution model and (S2), We define $\mathbf{Y} := \mathbf{X}\mathbf{W}$, and write down the feature vector of vertex i after the two graph convolution models, respectively. The output of our proposed graph convolution model is

$$\mathbf{Z}_i = f\left(\frac{1}{\tilde{\mathbf{D}}_{ii}}\left(\mathbf{Y}_i + \textstyle\sum_{j \in \Gamma(i)} \mathbf{Y}_j\right)\right), \tag{S3}$$

while the output of (S2) is

$$\mathbf{Z}_i = f\left(\frac{1}{\tilde{\mathbf{D}}_{ii}}\mathbf{Y}_i + \textstyle\sum_{j \in \Gamma(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{ii}\tilde{\mathbf{D}}_{jj}}} Y_j\right). \tag{S4}$$

As we can see, the random walk normalized propagation model (S3) has the following property: if i and i' have the same continuous vector color ($\mathbf{Y}_i = \mathbf{Y}_{i'}$) and their neighbor colors $\{\mathbf{Y}_j | j \in \Gamma(i)\}$ and $\{\mathbf{Y}_{j'} | j' \in \Gamma(i')\}$ match each other, then $\mathbf{Z}_i = \mathbf{Z}_{i'}$ after propagation. The key is that if i and i' have the same multiset of neighboring colors, they must have the same degree ($\tilde{\mathbf{D}}_{ii} = \tilde{\mathbf{D}}_{i'i'}$) too. Therefore, it exactly maps identical subtrees to identical colors, matching the behavior of WL.

On the other hand, the symmetrically normalized propagation model (S4) does not hold this property. This is because of the additional term $\tilde{\mathbf{D}}_{jj}$ inside the summation, which requires not only $\mathbf{Y}_j$ being matched, but also their degrees being equal in both multisets. Thus the new colors $\mathbf{Z}_i$ and $\mathbf{Z}_{i'}$ may be different after propagation even if they have identical rooted subtrees. This means that our graph convolution (S3) is a closer approximation to WL than (S4). Despite being a theoretically closer approximation, we empirically found (S3) and (S4) have similar performance, and chose (S3) as the default graph convolution model in DGCNN.

3. TRAINING THROUGH BACKPROPAGATION

The whole network can be trained efficiently through backpropagation. Let $\mathcal{L}$ denote the loss of a graph sample. It is standard to compute the gradients of $\mathcal{L}$ w.r.t. the traditional layers' parameters and inputs. Here we show how to do it for graph convolution and SortPooling layers.

Let $\mathbf{P} \in \{0,1\}^{k \times n}$ and $\mathbf{Z}^{\text{sp}} \in \mathbb{R}^{k \times \sum_t^h c_t}$ be the permutation matrix and output, respectively in the forward propagation of SortPooling, where $\mathbf{P}_{ij} = 1$ if the j^{th} row of $\mathbf{Z}^{1:h}$ is ranked i^{th} in $\mathbf{Z}^{\text{sp}}$ and 0 otherwise. We have

$$\mathbf{Z}^{\text{sp}} = \mathbf{P}\mathbf{Z}^{1:h}, \text{ and } \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{1:h}} = \mathbf{P}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{\text{sp}}}. \tag{S5}$$

For the first graph convolution layer, we let $\mathbf{V} := \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}$. Thus, $\mathbf{Z} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}) = f(\mathbf{V})$. We have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\frac{\partial \mathcal{L}}{\partial \mathbf{V}}\mathbf{W}^\top, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{X}^\top\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\frac{\partial \mathcal{L}}{\partial \mathbf{V}}, \text{ and } \quad \frac{\partial \mathcal{L}}{\partial \mathbf{V}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Z}} \odot f'(\mathbf{V}). \tag{S6}$$

For the stacked graph convolution layers, we also let $\mathbf{V}^t := \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{Z}^t\mathbf{W}^t$. Thus $\mathbf{Z}^{t+1} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{Z}^t\mathbf{W}^t) = f(\mathbf{V}^t)$. Here we need to further consider the gradients from the direct connection between $\mathbf{Z}^t$ and $\mathbf{Z}^{1:h}$. The complete loss gradient w.r.t. $\mathbf{Z}^t$ is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^t} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\frac{\partial \mathcal{L}}{\partial \mathbf{V}^t}\mathbf{W}^{t\top} + \left[\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{1:h}}\right]_{\{t\}}, \text{ and } \quad \frac{\partial \mathcal{L}}{\partial \mathbf{V}^t} = \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{t+1}} \odot f'(\mathbf{V}^t), \tag{S7}$$

where we use $[\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{1:h}}]_{\{t\}}$ to denote the c_t columns in $\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{1:h}}$ corresponding to $\mathbf{Z}^t$.

4. SUPPLEMENT FOR EXPERIMENTS

A. Detailed dataset information

We used five bioinformatics datasets. MUTAG [5] is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds classified according to their mutagenic effects on a bacterium. PTC [6] is a dataset of 344 chemical compounds classified according to their carcinogenicity for male and female rats. NCI1 [7] contains anti-cancer screens for cell lung cancer and ovarian cancer cell lines. PROTEINS and D&D are graph collections of chemical compounds classified into two classes: enzyme and non-enzyme [8]. Graphs in these five bioinformatics datasets all have vertex labels. Only MUTAG and PTC contain edge labels, which are not used in this paper. We used three social network datasets. COLLAB is a scientific collaboration dataset where ego-networks are generated for researchers and are classified into three research fields. IMDB-B is a movie collaboration dataset where ego-networks for actors/actresses are classified into Action or Romance genres. IMDB-M is a multi-class version of IMDB-B containing genres Comedy, Romance, and Sci-Fi. The COLLAB, IMDB-B, and IMDB-M are from [9]. Graphs in these social network datasets contain neither vertex labels nor edge labels, thus are pure structures.

B. Hyperparameter selection

We manually tuned the two hyperparameters, learning rate and number of training epochs, for each dataset. We did not tune hyperparameters independently for all 10 series cross validation experiments, but used one random splitting of training (90%) and validation (10%) data to select one pair of hyperparameters for each dataset to use consistently in all 10 series of cross validations. Concretely, the learning rates are selected from {0.01, 0.001, 0.0001, 0.00001}. The numbers of training epochs are selected from {100, 200, 300, 400, 500}. The combination which both shows convergence of optimization and small overfitting is selected. The adopted hyperparameters for each dataset are shown in Table S1.

Table S1. Hyperparameters for each dataset.

Dataset	MUTAG	PTC	NCI1	PROTEINS	D&D	COLLAB	IMDB-B	IMDB-M
learning rate	0.0001	0.0001	0.0001	0.00001	0.00001	0.0001	0.0001	0.0001
# training epochs	300	200	200	100	200	300	300	500

C. Supplementary experiments to test the effect of SortPooling

In order to understand SortPooling and graph convolutions deeper, we conducted four supplementary experiments on MUTAG and compared their performance. In the first experiment, we turned off SortPooling and fed graphs into DGCNN with random vertex orders. In the second experiment, we turned off SortPooling, and fed into DGCNN graphs with vertex orders calculated by the Weisfeiler-Lehman algorithm in a preprocessing step. In the third experiment, we replaced SortPooling with the summed node features as the input to the traditional layers. In the fourth experiment, we turned on SortPooling, but did not update the weights of the previous graph convolution layers (the weights are fixed all-one matrices).

We show the training curves of the four experiments compared to using the original SortPooling in Figure S1. As we can see, the validation loss curve of using the original SortPooling is always lower than those without SortPooling. In the first experiment, we can see that using random vertex orders resulted in easily overfitting the training data. This is because training and validation graphs are fed into neural networks without a consistent vertex order. In the second experiment, the training curve of SortPooling coincides well with that of using precomputed WL orders. This meets our expectation since SortPooling is designed to use the continuous WL colors for sorting. Interestingly, we find that the validation curve of SortPooling is lower than that of using the precomputed WL orders. This is because SortPooling has an additional regularization effect by dynamically sorting vertices. By dynamically sorting vertices during training, vertices are less sensitive to the precomputed fixed orders, which leads to a lower risk of overfitting. In the third experiment, we can see that using summed node features have consistently higher training and validation loss than SortPooling, meaning that it is much more difficult for summing to fit the data than for sorting. By summing, individual vertex features are averaged out instead

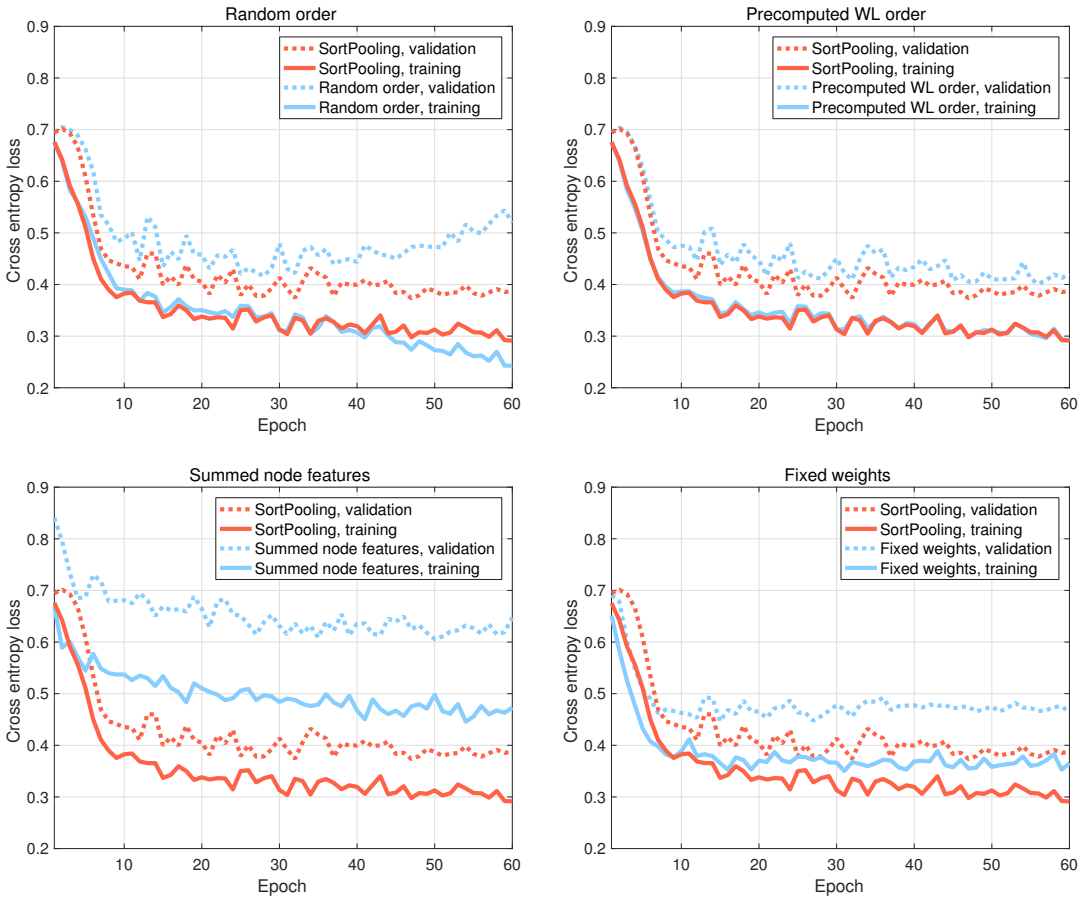

Fig. S1. Training curves of SortPooling compared with 1) random order, 2) precomputed WL order, 3) summed node features, and 4) fixed weights.

Table S2. Accuracy results on MUTAG for the supplementary experiments.

	Random order	Precomputed WL order	Summed node features	Fixed weights	SortPooling
Accuracy	78.72±3.04	82.56±2.71	78.50±2.43	82.50±2.74	85.83±1.66

of being kept, leading to a great loss of information. We used the fourth experiment to verify the usefulness of training the graph convolution parameters. As we can see, the inclusion of trainable graph convolution parameters leads to better expressing ability.

With the same experimental settings as the main paper, we further conducted 10 times 10-fold cross validations to test their performance. The accuracies and standard deviations are shown in Table S2. As we can see, using random vertex orders or summed node features have much lower accuracies than other networks. Compared to precomputed WL orders, the network with SortPooling increases the accuracy by 3%. Compared to using fixed weights in graph convolutions, SortPooling which backpropagates loss gradients and updates weights also shows about 3% accuracy improvement.

REFERENCES

1. D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis **30**, 129–150 (2011).
2. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv:1312.6203 (2013).
3. M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in "Advances in Neural Information Processing Systems," (2016), pp. 3837–3845.
4. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907 (2016).
5. A. K. Debnath, d. C. R. Lopez, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity." Journal of medicinal chemistry **34**, 786–797 (1991).
6. H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation of the predictive toxicology challenge 2000–2001," Bioinformatics **19**, 1183–1193 (2003).
7. N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," Knowledge and Information Systems **14**, 347–375 (2008).

8. P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," Journal of molecular biology **330**, 771–783 (2003).
9. P. Yanardag and S. Vishwanathan, "Deep graph kernels," in "Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining," (ACM, 2015), pp. 1365–1374.