Knowledge-Based Systems 257 (2022) 109919

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Irregular message passing networks

Xue Li [a], Yuanzhi Cheng [a,b,*]

[a] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang, 150001, China
[b] School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong, 266061, China

ARTICLE INFO

Article history:
Received 27 February 2022
Received in revised form 15 September 2022
Accepted 16 September 2022
Available online 22 September 2022

Keywords:
Graph neural networks
Message passing
Power iteration
Random propagation
Random attention

ABSTRACT

The graph neural network (GNN) is a widely adopted technique to process graph-structured data. Despite its pervasiveness, the exact reasons for the message aggregator's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from optimizing edge weights to improve the local fusion of node information. In this study, we demonstrate that such propagation weight optimization has a limited contribution to the success of message passing. Instead, we find that any normalized random attention (or edge weights) can have a similar and, sometime, even stronger effect. We refer to these randomly initialized propagations as irregular message passing. Experiments conducted on our random edge weight and random attention models verified the positive impact of weight randomness, uncovering the importance of the topology itself in achieving superior results for message iterations. Our code is available at https://github.com/Eigenworld/RAN.

1. Introduction

Graph neural networks (GNNs) are becoming increasingly popular due to their revolutionary performance in representation learning of graph-structured data, with applications in the social sciences, physics, applied chemistry, biology, and linguistics. They are the first choice to obtain an impression of one's data in most research areas focusing on graph data. The inspiration of GNNs is primarily derived from convolutional neural networks (CNNs). The resulting migration models naturally inherit their deep learning lineage — coupled activation functions and layer weights. GNNs broadly follow a recursive message passing scheme, where all the nodes iteratively follow some well-designed rules (or edge weights) to produce their new representations. Many variants of GNNs, such as graph convolutional networks (GCNs) [1] and graph attention networks (GATs) [2], have demonstrated ground-breaking performance in many tasks, such as node classification, link prediction, and graph classification. However, the design of these GNNs is based on empirical intuition, heuristics, and experimental trial-and-error. The theoretical explanation and model evaluation of state-of-the-art GNNs are thus important for understanding their representational properties and limitations.

The common thread in the literature is that the effectiveness of message iterations originates from the well-designed edge weight. One of the most widely used kernels is the diagonalizable propagation matrix, including all symmetric matrices (e.g., symmetric Laplacian matrix and cosine similarity matrix) and some asymmetric matrices (e.g., random walk Laplacian matrix). In [3], we demonstrate that these diagonalizable kernel-based GNNs can be best understood by removing layer weights and activation functions as follows:

$$\begin{cases} X'_1 = ReLU\left[MX\Omega_1\right] \\ X'_2 = ReLU\left[MX'_1\Omega_2\right] \\ \cdots \\ X'_{k-1} = ReLU\left[MX'_{k-2}\Omega_{k-1}\right] \\ X'_k = MX'_{k-1} \\ X' = Softmax\left(X'_k\Omega\right) \end{cases} \rightarrow \begin{cases} X'_k = M^kX \\ X' = Softmax\left(X'_k\Omega\right) \end{cases} \tag{1}$$

where M is the aggregator, X is the graph feature, and Ω is the layer weight.

The expression $X'_k = M^kX$ is known as power iteration [4], which is used to calculate eigenvectors. When k is large enough, the constant multiplication sends the feature vectors X to the eigen-subspace, where the clusters are well-separated:

$$M^kX = c_1(\lambda_1^k v_1) + c_2(\lambda_2^k v_2) + \cdots + c_n(\lambda_n^k v_n) \tag{2}$$

where v and λ are eigenvectors and eigenvalues of M, respectively. c is the coefficient ($c_i \neq 0$).

With increasing k, the noisy eigenvectors shrink quickly, and the eigenvectors with large eigenvalues are preserved to form an informative subspace.

* Corresponding author at: School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong, 266061, China; School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang, 150001, China.
E-mail addresses: nefu_education@126.com (X. Li), yzcheng@hitwh.edu.cn (Y. Cheng).

https://doi.org/10.1016/j.knosys.2022.109919
0950-7051/

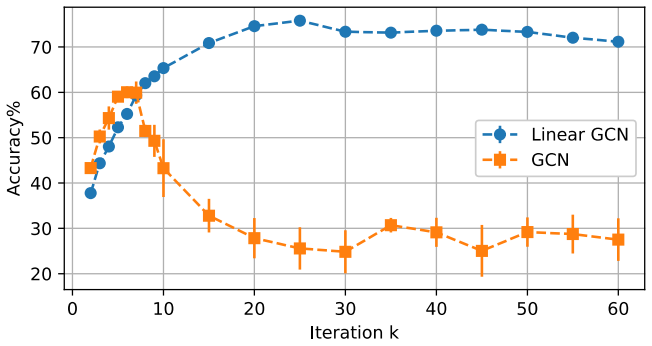

Fig. 1. Illustration of the eigen-subspace of Laplacian aggregator-based models on the Cora_300 dataset. The results are averaged over 5 runs and 100 epochs per run.

To obtain some understanding of the above eigen-subspace, we set $M = D^{-1/2}AD^{-1/2}$, and run Linear GCN [3,5] and GCN on Cora_300 (Cora with 300-dimensional random number features). As shown in Fig. 1, the accuracy of Linear GCN gradually increases with iterations and finally stabilizes at approximately 72%, forming an effective cluster subspace. Because there is no useful information in random features, the GCN learns nothing. The resulting noisy layer weights hinder the formation of an informative space.

It is more common to train shallow GNNs with meaningful graph features. When k is small, M^kX is the feature-dominated information fusion. When increasing k constantly, the feature information diminishes rapidly, and the structural representation (or eigen-subspace) gradually dominates the iteration. These are the two phases of power iteration. Section 4.4 provides a deeper analysis of the second phase.

One question that is easy to neglect but deserves exploration is: Is the effectiveness of message passing uniquely tied to the well-designed propagation weight? Or could a similar effect be achieved using some "bad" edge weights? Equipped with the above understanding of message passing, we propose irregular message passing to enrich the design of current GNNs. By conducting extensive experiments on our random edge weight networks and random attention networks, the new findings are as follows.

1. Message passing is not very sensitive to the edge weights. The graph topology itself is the key point for message aggregation.
2. We may not need a learnable attention mechanism. Any normalized random attention can achieve a satisfying performance.

Additionally, we also design generalized models of random attention by simply modifying the most popular algorithms on the OGB leaderboard. All experimental results confirmed the reliability and effectiveness of irregular message passing.

Our work makes the following contributions:

1. Irregular message passing is systematically presented for the first time.
2. Eigen-subspace analysis of GNNs is first given.
3. The structural defect of the graph attention mechanism is systematically presented.
4. The positive impact of randomness on neural learning systems is fully explored.
5. A unified geometric view of message propagation is proposed.

Table 1
Negative impact factors of attentions.

Data	Partial-neg	**Full-neg**	GAT result	**One degree**
Cora	29.4%	**58.3%**	82.9%	**18%**
CiteSeer	27.5%	**46.2%**	72.3%	**40%**
PubMed	43.6%	**36.9%**	79.7%	**46%**

The remainder of this paper is organized as follows. Section 2 analyzes the architecture development of graph neural networks and places much focus on the structural defects of the graph attention mechanism. Section 3 first presents a unified geometric view of message passing and then defines random weight networks, random attention networks, and random multi-attention networks. In Section 4, we conduct extensive experiments to verify the effectiveness of irregular message passing and explain why this is the case. Finally, Section 5 ends with a discussion and ideas for future research.

2. Related work

The earliest attempt at using neural networks for graphs can be traced back to the early 1990s, often referring to using recurrent neural networks to represent directed acyclic graphs [6,7]. The first proper construction of graph neural networks was proposed in the 2000s, and the well-designed propagation basically uses recursive aggregation until convergence [8,9]. What is generally known as graph neural networks starts by transferring convolutions from Euclidean to non-Euclidean space. Bruna et al. [10] and Defferrard et al. [11] used a Laplacian eigen-system to perform convolutions on the Fourier domain of graphs, which is known as spectral graph convolutional neural networks. Kipf and Welling [1] used a linear filter to approximate the above spectral method and reduced it to an efficient spatial architecture (GCN). After that, aggregating messages along the graph topology becomes the mainstream of the GNN design. Gilmer et al. [12] defined this iterative process as message passing, providing a unified view of GNNs.

The propagation matrix with excellent properties played an important role in the early development of GNNs. Similar to the kernel of GCN, the symmetric Laplacian matrix, its orthogonal eigenvectors can still well reflect the ideas of Fourier transform and convolution. To date, the consensus in the literature about the mechanism of message passing is still Laplacian smoothing. In the later spatial model design, however, spectral graph theory is almost absent in designing and explaining GNNs.

The new trend for designing spatial models is to incorporate the graph feature information into edge weights. The most well-known method is graph attention neural networks (GATs) [2], which has more than 4,000 citations from 2018 to 2022. The success of the attention mechanism has long been attributed to its adaptive information selection. Based on this common belief, many variants of attention models [13–15] have been developed to improve the local fusion of node information. Somewhat shockingly, however, this is not the case for graph attention.

Table 1 lists the distribution of nodes with partial or full negative attention scores. These scores are produced by one of GAT layer attentions. Li et al. [16] observed that the attention weights are distributed similarly on all citation networks, regardless of the heads and layers. Full negative attentions have little to do with the notion of focusing on informative neighbors, but GAT still achieves a good result using these degenerate attentions. The reason, as shown in Fig. 2, is that the softmax structure of the attention mechanism does not consider neighbor similarities. When all attention scores are negative, the softmax exponential function

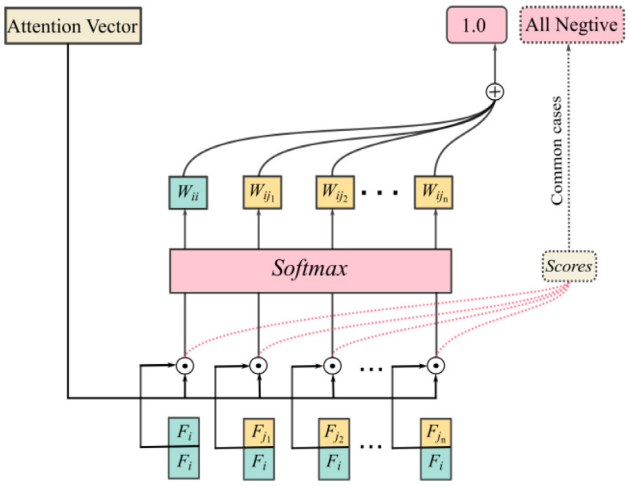

Fig. 2. The attention is dot-producted with each concatenated feature in the neighbor list to compute a score. This is done in parallel for all source nodes. Negative scores indicate the dissimilarity between the source node and its neighbors. The softmax structure transforms these scores into a list that sums to 1.0, even if they are all negative.

suppresses these negative inputs, resulting in local weights close to a uniform distribution $D^{-1}A$.

From the topology perspective, we do not need to control graph attention. Attentions are, in fact, constrained by the graph-affiliated structure. Affiliated nodes (e.g., one-degree nodes shown in Table 1 or nodes with many common neighbors) are forced to follow the label of their neighbors. These findings challenge the conventional wisdom about graph attention models.

Therefore, although knowledge from convolutional neural networks has enabled the graph learning community to make substantial gains in recent years, we anticipate that in the long term it is likely to impede progress. Our following explorations also confirm this claim.

3. Irregular message passing networks

In this section, we begin by explaining the mechanism of message passing in the context of spectral graph drawing [17] and present a unified geometric view of GNNs. Then, three irregular message passing networks are proposed. This is the first systematic study of the randomness in message propagation in the literature.

3.1. Geometric view of message passing

An attractive feature of power iteration is that it can be related to an intuitive geometric process. This study starts by exploring the optimization problem associated with Laplacian and defines the quadratic form of Laplacian as

$$x^T Lx = \sum_{(i,j)\in E} w_{ij}\left(x_i - x_j\right)^2, x \in R^{1\times n} \tag{3}$$

For each node i, the partial derivative of $x^T Lx$ with respect to x_i is

$$\frac{\partial x^T Lx}{\partial x_i} = 2\sum_{j\in N_i} w_{ij}\left(x_i - x_j\right) \tag{4}$$

Equating the Eq. (4) to zero and isolating the location of node i gives

$$x_i = \frac{\sum_{j\in N_i} w_{ij}x_j}{\deg(i)} \tag{5}$$

This new derivation suggests that placing each node at the weighted centroid of its neighbors could minimize $x^T Lx$. However, this solution is unwanted in that all the nodes will always be put at the same location. To prevent this collapse at the center of the drawing, we shift the neighbor centroid of node i outwards by the amount of $\mu|x_i|$, and $\mu > 0$. Then, we obtain

$$x_i - \frac{\sum_{j\in N_i} w_{ij}x_j}{\deg(i)} = \mu x_i \tag{6}$$

Stacking all the node vectors in the matrix yields

$$D^{-1}Lx = \mu x \tag{7}$$

We can rewrite it to obtain a more familiar eigen-equation,

$$\left(I + D^{-1}A\right)x = \lambda x, \lambda = 2 - \mu \tag{8}$$

According to power iteration, for any non-constant starting vector x_0, $\left(I + D^{-1}A\right)^k x_0$ will ultimately converge to the dominant eigenvector of $I + D^{-1}A$. By orthogonalizing against previously calculated eigenvectors at each iteration, we can calculate any desired eigenvector.

Fortunately, we do not need to calculate the exact eigen-basis, and a small number of matrix–vector(s) multiplications is enough to differentiate dissimilar nodes. Finally, we derive a linear GraphSAGE model [18] from the quadratic form of Laplacian.

$$\left(I + D^{-1}A\right)^k X \tag{9}$$

For each column of X, every node is iteratively put at the average between its previous location and the neighbor centroid.

By tuning the centroid and the deviation term in Eq. (6), we can construct any diagonalizable filters. However, for irregular filters, it is impossible to define such a perfect eigen-equation. Therefore, a more universal expression is

$$x_i \pm \sum_{j\in N_i} m_{ij}x_j \tag{10}$$

where m is the weight of the diffusion matrix and generally, it requires $0 < m_{ij} < 1$.

In light of graph drawing, we attribute the success of message passing to the local inward movements of nodes at each iteration. In the experiment, we will show that any irregular local inward movements could produce communities. Linear and nonlinear movements are both allowed.

3.2. Random weight networks

An interesting way of passing messages is random diffusion. The above geometric framework can be expressed as

$$x_i \pm \sum_{j\in N_i} m_{r_{ij}}x_j \tag{11}$$

where m_r is randomly generated in certain ways.

One natural construction of m_r is generating random neighbor weights directly. This model is referred to as a random weight network (RWN), and we define it in a power iteration style as follows, i.e., all the layers share the same diffusion matrix M.

$$\begin{aligned} &\text{Preprocessing:} \begin{cases} \text{Adjacency } A,\ \text{Random matrix } W \\ M = A * W + I \\ M = \text{Normalize}\,(M) \end{cases} \\ &\text{Model:} \quad \begin{cases} X' = M^k X \\ X_p = MLP\left(X'\right) \end{cases} \end{aligned} \tag{12}$$

where W obeys the uniform or normal distribution (in absolute value), *Normalize* is ℓ_p − row − normalization, for $p = 1, 2, \text{or}\infty$, and *MLP* is a linear classifier.

RWN, in fact, can be seen as a perturbation matrix of $I+D^{-1}A$, and the perturbations are the noisy matrix W and the methods of weight normalization. According to the random matrix theory [19], small perturbations do not change the eigen-system too much, but the high level of constrained perturbations (normalized random weights, in our case) are poorly studied. In Section 4, we explore the properties of the above irregular matrix in the graph node classification task.

3.3. Random attention networks

Another random weight is based on the attention mechanism. Existing attention-based models attach much importance to their learnable parameters. Similar to the expression below, Eq. (13), the kernel of GAT, is used at each GAT layer. The learnable parameters are the attention vector $\vec{\alpha}$ and the layer weight w. These parameters and the nonlinear activation *LeakyReLU* are thought of as the key factors of attention models. GAT also uses the multi-head mechanism to stabilize the performance, i.e., many attention components are trained in parallel at each layer.

$$M_{ij} = Softmax(LeakyReLU\left(\vec{\alpha}\left[h_i w \parallel h_j w\right]\right)) \tag{13}$$

Different from the above ideas, we believe we do not need to control graph attention. As discussed in Section 2, attention is constrained by the graph topology. Therefore, we could incorporate the feature information into the edge weights in a random way. We drop all learnable parameters, activation functions, and the multi-head mechanism and randomly generate the attention vector $\vec{\alpha}$ obeying a uniform or normal distribution. The proposed random attention network (RAN) is defined as the following PyTorch pseudocode.

$$\begin{aligned}&\text{Preprocessing}: \begin{cases}\text{Random attention vector } \vec{\alpha}\\ \vec{\alpha} = \text{Normalize}\,(\vec{\alpha})\\ M_{ij} = \text{Softmax}(\vec{\alpha}\cdot[\mathrm{X}_{[i,:]} \parallel \mathrm{X}_{[j,:]}])\end{cases}\\ &\text{Model}: \quad \begin{cases}X' = M^k X\\ X_p = MLP\left(X'\right)\end{cases}\end{aligned} \tag{14}$$

where $\parallel$ is the concatenation operation to concatenate two node representations, $\mathrm{X}_{[i,:]}$ and $\mathrm{X}_{[j,:]}$.

As shown in Eq. (14), there is only one attention vector, and the propagation weight M is calculated based on attention-feature cosine similarities. This random attention structure is hard to reconcile with the claim that the performance gain of attention-based models stems from their adaptive information selection.

3.4. Random multi-attention networks

We also construct a random multi-head attention network (RMN) that is very similar to GAT. The random weight for each head attention is calculated using Eq. (15). The resulting propagation matrices are then, as shown in Eq. (16), used to finish 1-order message passing, which is run in parallel per layer. The final step is concatenating or averaging the outputs of different heads to stabilize the node representation. By repeating the above random multi-head attention layer, we can build a linear, nonparametric, and random graph attention neural network.

$$\text{Head (X)} = \begin{cases}\text{Predefine}: \begin{cases}\vec{\alpha} \leftarrow \text{Random Initialization}\\ \vec{\alpha} = \text{Normalize}\,(\vec{\alpha})\end{cases}\\ M_{ij} = \text{Softmax}(\vec{\alpha}\cdot[\mathrm{X}_{[i,:]} \parallel \mathrm{X}_{[j,:]}])\end{cases} \tag{15}$$

$$\text{Repeat k Times} \begin{cases}M_{ith} = Head\,(X_{in})\\ X_{outs} = [M_0, M_1, \ldots, M_{\#}]\cdot X_{in}\\ X_{layer} = \text{Concat or Mean}\,(X_{outs})\\ X_{in} = X_{layer}\end{cases} \tag{16}$$

In the geometric view, all the nodes follow the same random rule to move to the centroid of their neighbors at each iteration.

3.5. Nonlinear architecture

The current popular GNNs prefer to insert the feature transformation and the activation function into their aggregation process. During the training, the weight update is coupled with message propagation, which increases the training complexity. We now seek to design nonlinear random models with Parameter-Free message passing. Here, we take RWN as an example and propose RWN+ as follows:

$$\begin{cases}\text{Random aggregator } M\\ X_1 = MX\\ X_2 = MX_1\\ \ldots\\ X_k = MX_{k-1}\\ X' = \frac{1}{k}\sum_i^k ReLU\,(W_i X_i)\\ X_p = MLP\left(X'\right)\end{cases} \tag{17}$$

To keep passing messages only once, we use the nonlinear skip connection to feed all previous outputs to the MLP classifier. The aggregation process still discards activation functions and layer weights. We further propose RAN+ and RMN+ in the same way.

3.6. Randomness analysis

The randomness in our models refers to the edge weight initialization, and it is just an unparameterized way to define aggregators. Random attention (or edge weights) is produced before the training and kept constant all the time. Previous studies using random processes are different from ours. Of particular interest are Dropout [20], DropEdge [21], FastGCN [22], and GraphSAGE [18]. They randomly remove some of the elements of graph attributes to boost the performance or save memory. Randomness is very important for them. If the binary operation (keep or remove) constantly removes very important input features, edges, or nodes, the model performance will drop drastically. However, for irregular message passing, randomness is not as important. Experiments in Sections 4.3 and 4.5 show that RAN is not sensitive to its attention values. The learnable attention vector can be replaced with a random vector to reduce unnecessary parameter learning.

4. Experiments

How much improvement is provided by optimizing the propagation weight, compared to just using random aggregators? To address this issue, we performed a comparative evaluation of irregular message passing networks against traditional representation learning methods and various GNN variants in node classification tasks.

4.1. Experimental setup

Dataset. We conduct exploratory experiments using six node classification benchmark datasets across domains: citation networks (Cora, CiteSeer, PubMed [23], DBLP [24]), air traffic (Air-USA [25]), and protein–protein interactions (PPI [26]). All the citation networks are PyTorch built-in data, which are split well

Table 2
Dataset statistics of benchmark networks.

Dataset	#Nodes	#Edges	#Features	Train/val/test
Cora	2,708	5,429	1,433	140/500/1, 000
CiteSeer	3,327	4,732	3,703	120/500/1, 000
PubMed	19,717	44,338	500	60/500/1, 000
Air-USA	1,190	13,599	238	119/238/833
PPI	2,599	27,189	50	2, 050/297/252
DBLP	17,716	105,734	1,639	1, 772/1, 772/14, 172

for training. The Cora data in DGL have a different test set from that of PyTorch, and we also present the test on Cora_dgl. For PPI, we choose two of its twenty-four networks and treat them as one large network. Summary statistics for each dataset are shown in Table 2.

Baseline. We evaluate the performance of our proposed irregular networks by comparing them with several baselines. (1) Traditional graph representation learning methods: Deep Walk [27], LP [28], and LNet [29]. (2) State-of-the-art GNNs: GCN [1], GAT [2], AGNN [13], and GraphSAGE [18]. (3) Power GNNs: SGC [5] or DAD [3] (their structures are almost the same, but theories are different).

Settings. All experiments on GNNs are performed based on the code and the parameters released by the corresponding published paper. The results are averaged over 10 runs and 100 epochs per run. For stability analysis, we repeat the above setting experiments 10 times for our random models and GCN, recording the worst and best results. **Unless stated otherwise, random attentions follow the normal distribution; RWN follows the uniform distribution; and all initial values are scaled via L2 normalization.** Additionally, the table cells of baseline records are intentionally left empty if there is no relative report in the original paper.

4.2. Analysis and evaluation

Analysis of regular and irregular message passing. One crucial question for message passing is, do we only need to study the diffusion matrix of having good mathematical properties? Traditionally, that is what we do. Many Laplacian-based models (e.g., GCN and SAGE) are proposed as standard structures of GNNs. Our work takes an opposite approach to design neural message passing. The edge weights of RWN are completely random, but as shown in Table 3, its best results are very close to those of regular GNNs; its worst results still

outperform most of the traditional methods (RWN vs. LP, LNet and DeepWalk), showing the robustness and superiority of the framework of message aggregation. Our original motivation for RWN is just to present the tolerance of GNN for irregular weights, but its behavior is surprising, and it achieves the best result on Air-USA.

Compared with RWN, random attention can better show the advantages of irregular propagation. They integrate graph features into edge weights in an unparameterized way, further reducing the chaos of node movements. RAN and RMN achieve similar and, sometime, even better results than regular GNNs. Compared with GCN, the best and worst results show the stability of these two random attention models. It is noteworthy that random attentions perform slightly better than learnable attentions. If considering the structure and training complexity, random attention is no doubt a better choice. This also indicates that researchers may attach too much importance to the parameter learning of message passing.

When further comparing RAN and RMN, we find that stacking random attention layers and increasing model randomness does not degrade the RMN performance; it still achieves a similar result as RAN. Clearly, these findings are hard to reconcile with the claim that the performance gain of attention-based models stems from their adaptive information selection. The graph topology itself is, in fact, a very important but less studied factor in message passing.

Nonlinear test. We also explore the importance of nonlinearity in GNN predictions. To keep passing messages only once, we propose RWN+, RAN+, and RMN+ based on nonlinear skip connections. As shown in Table 4, linear random models do not work on PPI data. Adding activation functions and learnable weights can greatly improve the model performance. RMN+ even outperforms GCN and GAT.

Thus far, the effectiveness of random weights has been fully verified. It is reasonable to conclude that the success of message passing is not uniquely tied to these perfect aggregations. We also view these results as an opportunity to encourage the graph learning community to pursue a more generalized explanation of message passing.

4.3. Ablation studies

Problem of data preprocessing. To the best of our knowledge, almost all the experiments on PubMed in the literature are performed without normalizing the data properly. The value of PubMed ranges from 0.0 to 1.2633, and it should be treated differently from one-hot or multi-hot data. Here, we normalize PubMed features by min–max normalization and retest some models in Table 5.

Compared with the performance record presented in Table 3, all the models are obviously improved, especially SGC, GCN, and RAN. The GNN example code in PyTorch uses L_1 normalization. Therefore, test L_1, L_2, and min–max feature normalization with RAN. Table 6 shows that min–max normalization outperforms all the other methods. The importance of data preprocessing to models cannot be overstressed.

Distribution test. The distribution of initial random weights could be N(0,1) or U(0,1) . Table 7 lists the results of RWN and RAN on the CiteSeer network. The normal distribution seriously degrades the performance of the RWN because of negative weights. When taking absolute values, RWN behaves much better. However, RAN is not sensitive to the initial weight distribution because the softmax structure of the attention mechanism keeps the edge weight between 0 and 1.

Normalization test. Thus far, all experiments are done under L_2 weight normalization, and it is time to check the effectiveness of L_1 and L_∞. Table 8 shows that all L_p normalizations could improve the model performance on the CiteSeer network compared with raw random weights. L_1 and L_2 perform better than L_∞ normalization.

Attention norm vs. Edge weight norm. In RAN, we use attention normalization, and what if we change it to final edge weight normalization, similar to RWN? Table 9 shows that this approach is reasonable. However, from the view of the computation complexity, normalizing attentions is much more economic.

Extreme attention test. Here, we generate random attention with varying variances to see the performance change of RAN. Table 10 shows that random attentions are not sensitive to initial values. This once again confirms our claim that we do not need to control graph attention.

4.4. Why do random weights work?

Singular value visualization. The positive impact of normalized random weights on message passing is not serendipitous.

Table 3
Irregular propagation performance across previous methods on benchmark networks.

Item	Method	Cora_dgl	Cora	CiteSeer	PubMed	Air-USA	DBLP
Records from Literature	GCN	81.5%		70.3%	79.0%	-	-
	AGNN	83.1 ± 0.1%		71.7 ± 0.1%	79.9 ± 0.1%	-	-
	GAT	83.0 ± 0.7%		72.5 ± 0.7%	79.0 ± 0.3%	-	-
	LP	68.0%		45.3%	63.0%	-	-
	LNet	79.5 ± 1.8%		66.2± 1.9%	78.3± 0.3%	-	-
	DeepWalk	70.7 ± 0.6%		51.4 ± 0.5%	76.8 ± 0.6%	-	-
GNNs	GCN	81.7± 0.8%	81.5± 0.6%	71.7± 0.7%	78.8 ± 0.6%	58.2±1.0%	83.9±0.3%
		82.3± 0.4%	82.2 ± 0.6%	**72.3 ± 0.6%**	79.1± 0.2%	**58.9±1.0%**	84.0±0.2%
	SAGE	82.3±0.9%	81.8 ± 0.8%	71.4 ± 1.0%	78.5 ± 0.5%	54.6±0.5%	82.9±0.3%
	AGNN	83.3± 0.9%	81.6 ± 0.7%	71.5 ± 0.7%	78.9 ± 0.7%	56.1±0.6%	**84.1±0.1%**
	GAT	83.1± 1.4%	**82.4 ± 0.8%**	71.7 ± 0.8%	78.1 ± 0.6%	55.8±1.0%	83.6±0.2%
	SGC	**83.8±0.0%**	82.3 ± 0.5%	72.0 ± 0.4%	**79.2 ± 0.4%**	58.4±0.2%	84.0±0.1%
Random Aggregator (RA)	RWN	78.8±0.1%	79.1±0.6%	70.0±0.5%	77.4±0.2%	57.9±0.7%	82.4±0.1%
		80.2±0.5%	80.1±0.5%	71.9±0.4%	78.8±0.3%	**60.5±0.8%**	82.7±0.1%
	RAN	83.0±0.4%	82.1±0.8%	72.1±0.2%	78.9±0.3%	58.9±0.9%	83.5±0.1%
		83.1±0.2%	**82.5±0.8%**	72.5±0.2%	79.2±0.4%	59.5±0.9%	83.5±0.1%
	RMN	83.2± 0.0%	81.5±0.7%	72.1±0.7%	79.2±0.1%	56.1±0.3%	83.9±0.1%
		83.3 ± 0.0%	82.1±0.3%	**72.6±0.5%**	**79.4±0.2%**	56.7±0.3%	**84.0±0.1%**
Best GNN – Best RA		**+0.5%**	**-0.1%**	**-0.3%**	**-0.2%**	**-1.6%**	**+0.1%**

Table 4
Nonlinear test on PPI.

GNN Arch.	Method	PPI
GNNs	**GCN**	**62.1 ± 0.6%**
	GAT	**65.5 ± 0.5%**
RWNs	RWN	53.7 ± 0.3%
	RWN+	58.3 ± 0.6%
		59.4 ± 0.4%
RANs	RAN	49.9 ± 0.9%
	RAN+	60.5 ± 0.4%
		61.2 ± 0.9%
RMNs	RMN	56.4 ± 0.2%
	RMN+	64.7 ± 0.3%
		65.6 ± 0.4%
Best GNN – Best RA		**-0.1%**

Table 5
Results on normalized PubMed.

GCN	GAT	RMN
81.3 ± 0.4%	79.8 ± 0.4%	80.4 ± 0.4%
SGC	RWN	**RAN**
82.0 ± 0.5%	79.6 ± 0.1%	**80.9 ± 0.3%**

Table 6
RAN feature normalization test on PubMed.

Pub_None	Pub_L1	Pub_L2	**Pub_min–max**
78.9 ± 0.3%	79.3 ± 0.3%	79.5 ± 0.2%	**80.6 ± 0.4%**
79.2 ± 0.4%	79.5 ± 0.3%	79.7 ± 0.1%	**80.9 ± 0.3%**

Table 7
Distribution test of irregular weight.

Model	Norm	**Uniform**	**|Norm|**
RWN	30.5 ± 1.3%	**71.3 ± 0.4%**	**70.6 ± 0.9%**
RAN	**72.3 ± 0.4%**	**72.2 ± 0.4%**	**72.3 ± 0.4%**

Table 8
Normalization test of irregular weight.

Model	Raw	L_1	$\boldsymbol{L_2}$	L_∞
RWN	68.3 ± 0.7%	**71.4 ± 0.3%**	**71.5 ± 0.3%**	70.0 ± 0.4%
RAN	67.0 ± 0.9%	**72.3 ± 0.4%**	**72.4 ± 0.3%**	67.0 ± 1.2%

Table 9
Attention norm vs edge weight norm.

Model	Cora	CiteSeer	PubMed	AirUSA
Att_L2	**82.1 ± 0.8%**	72.4 ± 0.3%	**81.0 ± 0.4%**	**59.3 ± 0.4%**
Weight_L2	81.9 ± 0.8%	**72.6 ± 0.3%**	79.9 ± 0.3%	56.0 ± 0.9%

To demonstrate this, we design the following four propagation matrices of RWN:

$$\begin{cases} R_1 = I + D^{-1}A. \\ R_2 = L_2\_\text{Norm}\left(\ddot{A} * W_u\right) \\ R_3 = L_2\_\text{Norm}\left(\ddot{A} * W_n\right) \\ R_4 = L_2\_\text{Norm}(\ddot{A} * |W_n|) \end{cases} \tag{18}$$

where $\ddot{A} = \text{A}+\text{I}$, W_u and W_n are uniform and normal distribution weights, respectively.

For visualization convenience, we use the small Karate Club network, which represents friendship between 34 members of a university-based karate club. For each distribution, W_u and W_n, we generate 1,000 random matrices and draw their singular values in Fig. 3.

Observing the difference between Fig. 3(a) and (b), we find that R_2 is just the perturbation matrix of R_1. (Normalized) random

Table 10
Accuracy of extreme attention test (%).

Data	N(0,0)	N(0,1)	N(0,10)	N(0,100)	N(0,1000)
Cora	**82**.2 ± 0.6	**82**.4 ± 0.6	**82**.1 ± 0.7	**82**.4 ± 0.8	**82**.2 ± 0.7
CiteSeer	**72**.5 ± 0.2	**72**.4 ± 0.4	**72**.4 ± 0.4	**72**.5 ± 0.2	**72**.6 ± 0.5
PubMed	**80**.6 ± 0.4	**80**.5 ± 0.5	**80**.7 ± 0.5	**80**.6 ± 0.5	**80**.5 ± 0.6

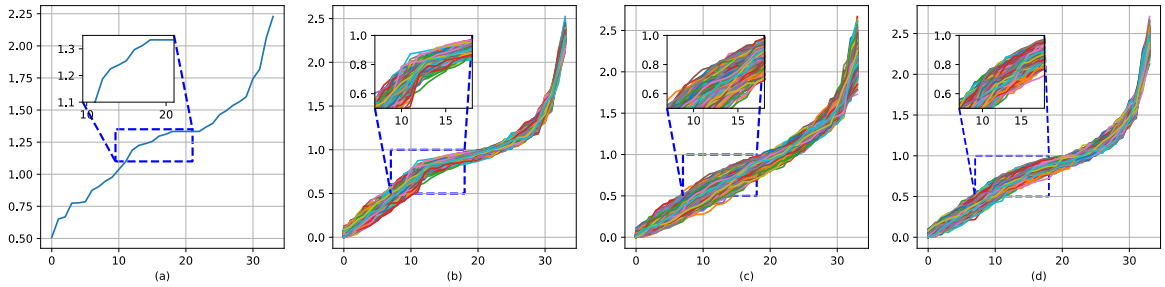

Fig. 3. (a), (b), (c), and (d) correspond to the ascending singular values of $1 \times R_1$, $1000 \times R_2$, $1000 \times R_3$, and $1000 \times R_4$. Fig. 3(a) and (b) have highly similar fluctuations, showing the stable similarities between R_1 and R_2. However, the curves in (c) tend to be straight lines, resulting in severe information loss. After taking the absolute value of R_3, the R_4 curves in (d) present a similar bending as that in (a).

Table 11
Accuracy of random feature iteration.

Model	Cora_300	CiteSeer_500	PubMed_1000
DA	**75.7 ± 0.4%** **k=20**	**52.7 ± 0.2%** **k=40**	**76.8 ± 0.3%** **k=50**
RWN	**62.4 ± 1.5%** **k=15**	**42.6 ± 1.1%** **k=15**	**76.1 ± 0.6%** **k=15**
SAGE	57.1 ± 2.9% k=5	35.5 ± 2.4% k=5	52.6 ± 4.0% k=10
RAN	**74.8 ± 0.4%** **k=25**	**52.3 ± 0.7%** **k=60**	**75.8 ± 0.7%** K=40

uniform perturbations do not change singular values too much. However, the raw normal distribution weight R_3 destroys the original structure properties as shown in Fig. 3(c). In Section 4.3, we have shown that taking absolute values of the normal distribution can help improve the performance of RWN, which is consistent with the trend in Fig. 3(d).

Analysis of iteration subspace. We perform the random feature test to explore the eigen-subspace of GNNs and list the results for the R_1-based model (DA), RWN, and nonlinear R_1-based model (SAGE) as well as RAN in Table 11. The optimal number of iterations k is chosen from 1 to 60.

For diagonalizable kernel R_1, the increasing iterations send the random feature vectors X to its eigen-space, where the clusters are well-separated. Therefore, deep DA is highly performant among all citation networks. For RWN, its subspace can be regarded as a rough version of DA eigen-space. These two linear models perform much better than SAGE. During the training of SAGE, the random features mislead the parameter learning and aggravate the over-smoothing.

For RAN, we first use the meaningful features to compute the aggregation weight and then run it with the random features. It is difficult to judge whether the RAN aggregator is diagonalizable, but it indeed forms an effective eigen-subspace. When setting attentions to be zero vectors, RAN is equivalent to the model DA. Looking again at the Extreme Attention Test in Section 4.3, it suggests that there is no obvious difference between N(0,0) and other distributions. Therefore, the RAN subspace is similar to the DA eigen-subspace.

Taking all the results in Section 4.4 together, we conclude that the effectiveness of message passing is not uniquely tied to the well-designed propagation weight. Randomly passing messages is also an effective aggregative process.

Table 12
Dataset statistics of OGB networks.

Dataset	#Nodes	#Edges	#Features	Train/val/test
ogbn-arxiv	169,343	1,166,243	128	54%/17%/29%
ogbn-products	2,449,029	61,859,140	100	8%/2%/90%

4.5. Generalization test of random attentions

The proposed concepts in this paper challenge the widely used attention mechanisms in the research community. It may not be convincing to test random attention only in our proposed models, so we further design random attention models for another six popular attention models ranked at the top of the Open Graph Benchmark (OGB) leaderboard and test them on the ogbn-arxiv and ogbn-products datasets [30]. Summary statistics for each dataset are shown in Table 12. All experiments in this subsection are performed based on the code and the parameters released on the OGB leaderboard. The results are averaged over 5 runs and 2,000 epochs per run.

Generalization test on ogbn-arxiv. We choose the three most popular attention-based models on the OGB leaderboard: GAT+Norm [31], DRGAT [OGB Leaderboard], and RevGAT [32]. GAT+Norm uses the symmetric Laplacian matrix in GCN to normalize the attention edge weights produced by GAT; DRGAT uses RNN to model the residual evolving pattern between layers; and RevGAT generalizes reversible residual connections to grouped reversible residual connections for GAT. Their corresponding random models R-GAT+Norm, R-DRGAT, and R-RevGAT are simply constructed by replacing their learnable attentions with normalized random attentions. Moreover, some of the above models use GIANT+XRT [33] to augment the input graph features.

As shown in Table 13, the performance of random attention-based models is similar to that of learnable attention models. The added randomness has no hits to stability. Random attention works well with complex components such as RNN residual connections (R-DRGAT) and reversible residual connections (R-RevGAT)

Generalization test on ogbn-products. SAGN [34] and its variants [35] are currently the best models on the OGB leaderboard for ogbn-products. SAGN keeps the message passing linear and hypothesizes that the intermediate representations are informative. Diagonal attentions are used to integrate these multi-hop node features into the final prediction.

Table 13
Generalization test of random attention on ogbn-arxiv.

Model		Val acc(%)	Test acc(%)
Records from OGB Leaderboards	GAT+Norm	75.16 ± 0.08%	73.91 ± 0.12%
	GIANT-XRT+DRGAT	77.16 ± 0.08%	76.11 ± 0.09%
	GIANT-XRT+RevGAT	77.01 ± 0.09%	75.90 ± 0.19%
Our experiments	GAT+Norm	75.08 ± 0.08%	73.98 ± 0.16%
	R-GAT+Norm	**75.06 ± 0.06%**	**73.93 ± 0.07%**
	GIANT-XRT+DRGAT	77.09 ± 0.05%	76.00 ± 0.24%
	GIANT-XRT+R-DRGAT	**77.14 ± 0.05%**	**76.04 ± 0.08%**
	GIANT-XRT+RevGAT	76.92 ± 0.10%	76.17 ± 0.17%
	GIANT-XRT+R-RevGAT	**76.89 ± 0.12%**	**76.10 ± 0.10%**

Table 14
Generalization test of random attention on ogbn-products.

Model		Val Acc(%)	Test Acc(%)
Records from OGB Leaderboards	SAGN	93.11 ± 0.05%	81.28 ± 0.12%
	SAGN+SLE	93.09 ± 0.07%	84.68 ± 0.12%
	GIANT-XRT+ SAGN+MCR	93.89 ± 0.02%	86.51 ± 0.09%
Our experiments	SAGN	92.40 ± 0.02%	82.52 ± 0.23%
	R-SAGN	**92.30 ± 0.04%**	**82.73 ± 0.18%**
	SAGN+SLE	93.0 ± 0.06%	84.55 ± 0.08%
	R-SAGN+SLE	**92.87 ± 0.10%**	**84.53 ± 0.04%**
	GIANT-XRT+ SAGN+MCR	93.90 ± 0.05%	86.50 ± 0.06%
	GIANT-XRT+ R-SAGN+MCR	**93.87 ± 0.02%**	**86.55 ± 0.05%**

Table 15
Dynamic randomness test on ogbn-arxiv.

Model	Input	N(0,0)	N(0,1)
FR-RevGAT	76.40%	76.36 ± 0.0%	76.22 ± 0.15%
Model	**N(0,10)**	**N(0,100)**	**N(0,1000)**
FR-RevGAT	**76.23 ± 0.15%**	**76.22 ± 0.16%**	**76.23 ± 0.15%**

Table 16
Bad attention test.

Model	**Input1**	**Input2**	Input3
FR-RevGAT	**75.87%**	**76.16%**	76.23%
R-RevGAT	**76.14 ± 0.08%**	76.14 ± 0.22%	76.10 ± 0.08%
Best run	**76.23%**	**76.50%**	76.24%

SAGN+SLE, and GIANT-XRT+R-SAGN+MCR are still highly performant. This no doubt improves our understanding about the attention mechanism.

Additionally, most experiments in this paper study the attentions constrained by the graph topology. The difference here is that the attention in R-SAGN is used as a feature selection tool, which enlarges the application of random attentions.

Dynamic random attention test. We vary the variance of random attentions and hold learned parameters constant, running 1,000 times, to observe the performance change. Table 15 shows that the well-chosen pretrained R-RevGAT achieves an accuracy of 76.40%. After replacing the initial random attention with extreme values, the performance of R-RevGAT (or FR-RevGAT) is still very stable and close to that of SOTA—76.33%. Therefore, the randomness of attention has a negligible effect on performance as a whole. The next logical question is: there must be some "bad attentions", how can we filter them out?

First, looking again at Tables 13 and 14, the results of learnable attention models are not constant, meaning that the so-called "bad attentions" are normal to some extent. Second, the results of "bad attentions" are not as poor as expected. As shown in Table 16, we choose some random attention from the above Frozen R-RevGAT (or FR-RevGAT) test as the attention values of standard R-RevGAT. The performance of R-RevGAT is still satisfactory compared with the results reported in Table 13. Third, if attention must be determined, trying different attentions with the trained models is not truly time-consuming. Training models with different initial random attentions is also feasible.

5. Conclusion

We conduct extensive experiments to show the high tolerance of GNNs for irregular edge weights. From the geometric view, nodes could move irregularly along the graph topology toward their neighbor centroids, producing recognizable communities. The exact nature of the data and the form of random weights will affect the quality of communities. We dedicate a significant portion of this paper discussing random attentions, trying to prove that graph nodes, constrained by the topology, do not need to control attentions. Moreover, we propose the concept of an eigen-subspace for GNNs to better analyze the structural property of their aggregators.

For future work, it is worth exploring the effectiveness of random attentions in more complex models such as Transformers [36] and ViT [37] as well as more practical applications such as human–object interactions [38] and drug property analysis [39].

CRediT authorship contribution statement

Xue Li: Conceptualization, Data curation, Methodology, Writing – original draft. **Yuanzhi Cheng:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions that greatly improved the paper's quality. This work was supported in part by the National Natural Science Foundation of China under Grant No.61702135.

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proc. of ICLR, 2017, Toulon, France.
[2] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proc. of ICLR, Vancouver, Canada, 2018.
[3] X. Li, Y.Z. Cheng, Understanding the message passing in graph neural networks via power iteration, Neural Netw. 140 (2021) 130–135.
[4] G.H. Golub, C.F. Van Loan, Symmetric eigenvalue problems, in: Matrix Computations, Vol. 3, JHU Press, 2012, chap. 8.

[5] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: Proc. of ICLR, 2019, pp. 6861–6871.
[6] A. Sperduti, A. Starita, Supervised neural networks for the classification of structures, IEEE Trans. Neural Netw. Learn. Syst. 8 (1997) 714–735.
[7] P. Frasconi, M. Gori, A. Sperduti, A general framework for adaptive processing of data structures, IEEE Trans. Neural Netw. Learn. Syst. 9 (1998) 768–786.
[8] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proc. of IJCNN, 2005, pp. 729–734.
[9] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, in: Proc. of IJCNN, 2009, pp. 61–80.
[10] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: Proc. of ICLR, 2014.
[11] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proc. of NIPs, 2016, pp. 3844–3852.
[12] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proc. of ICML, 2017, pp. 1263–1272.
[13] K.K. Thekumparampil, S. Oh, C. Wang, L.J. Li, Attention-based graph neural network for semi-supervised learning, 2018, arXiv:1803.03735.
[14] Y. Zhang, X. Wang, C. Shi, X. Jiang, Y.F. Ye, Hyperbolic graph attention network, 2021, arXiv:abs/1912.03046.
[15] D. Kim, Alice H. Oh, How to find your friendly neighborhood: Graph attention design with self-supervision, paper presented at ICLR, 2021.
[16] M. Li, H. Zhang, X. Shi, et al., A statistical characterization of attentions in graph neural networks, paper presented at ICLR, 2019.
[17] Y. Koren, Drawing graphs by eigenvectors: theory and practice, Comput. Math. Appl. 49 (2004) 1867–1888.
[18] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proc. of NIPS, 2017.
[19] G.W. Stewart, Error and perturbation bounds for subspaces associated with certain eigenvalue problems, SIAM Rev. 15 (4) (1973) 727–764.
[20] N. Srivastava, G.E. Hinton, A. Krizhevsky, et al., Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[21] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: Towards deep graph convolutional networks on node classification, paper presented at ICLR, 2020.
[22] J. Chen, T. Ma, C. Xiao, FastGCN: Fast learning with graph convolutional networks via importance sampling, 2018, arXiv:1801.10247.
[23] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, Ai Mag. 29 (2008) 93–106.
[24] A. Bojchevski, S. Gunnemann, Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking, in: Proc. of ICLR, 2018.
[25] J. Wu, J. He, J. Xu, Demo-net: Degree-specific graph neural networks for node and graph classification, in: Proc. of ICLR, 2019, pp. 406–415.
[26] M. Zitnik, J. Leskovec, Predicting multicellular function through multi-layer tissue networks, Bioinformatics 33 (2017) i190–i198.
[27] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proc. of KDD. ACM, 2014, pp. 701–710.
[28] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proc. of ICLR, 2003, pp. 912–919.
[29] R. Liao, Z. Zhao, R. Urtasun, R. Zemel, Lanczos-net: Multi-scale deep graph convolutional networks, in: Proc. of ICLR, 2019.
[30] W. Hu, M. Fey, M. Zitnik, et al., Open graph benchmark: Datasets for machine learning on graphs, in: NIPs. Vol. 33, 2020, pp. 22118–22133.
[31] Y. Wang, J. Jin, W. Zhang, et al., Bag of tricks for node classification with graph neural networks, 2021, arXiv:2103.13355.
[32] G. Li, M. Muller, B. Ghanem, et al., Training graph neural networks with 1000 layers, in: ICML, 2021, pp. 6437–6449.
[33] E. Chen, C. Chang, J. Hsieh, et al., Node feature extraction by self-supervised multi-scale neighborhood prediction, 2021, arXiv:2111.00064.
[34] C. Sun, H. Gu, J. Hu, Scalable and adaptive graph neural networks with self-label-enhanced training, 2021, arXiv:2104.09376.
[35] C. Zhang, Y. He, Y. Chen, et al., Improving the training of graph neural networks with consistency regularization, 2021, arXiv:2112.04319.
[36] A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need, in: NIPs, Vol. 30, 2017.
[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv:2010.11929.
[38] S. Qi, W. Wang, B. Jia, J. Shen, S. Zhu, Learning human-object interactions by graph parsing neural networks, in: ECCV, 2018, pp. 401–417.
[39] K. Hsieh, Y. Wang, L. Chen, et al., Drug repurposing for COVID-19 using graph neural network and harmonizing multiple evidence, Sci. Rep. 11 (1) (2021) 1–13.