SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2024, Vol. 67, Iss. 7, 172101:1–172101:12
https://doi.org/10.1007/s11432-022-3850-0

Re-quantization based binary graph neural networks

Kai-Lang YAO & Wu-Jun LI*

National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210046*, China*

Received 12 June 2022/Revised 19 October 2022/Accepted 29 April 2023/Published online 17 June 2024

Abstract Binary neural networks have become a promising research topic due to their advantages of fast inference speed and low energy consumption. However, most existing studies focus on binary convolutional neural networks, while less attention has been paid to binary graph neural networks. A common drawback of existing studies on binary graph neural networks is that they still include lots of inefficient full-precision operations in multiplying three matrices and are therefore not efficient enough. In this paper, we propose a novel method, called re-quantization-based binary graph neural networks (RQBGN), for binarizing graph neural networks. Specifically, re-quantization, a necessary procedure contributing to the further reduction of superfluous inefficient full-precision operations, quantizes the results of multiplication between any two matrices during the process of multiplying three matrices. To address the challenges introduced by re-quantization, in RQBGN we first study the impact of different computation orders to find an effective one and then introduce a mixture of experts to increase the model capacity. Experiments on five benchmark datasets show that performing re-quantization in different computation orders significantly impacts the performance of binary graph neural network models, and RQBGN can outperform other baselines to achieve state-of-the-art performance.

Keywords graph neural networks, binary neural networks, mixture of experts, computation-efficient algorithms

1 Introduction

Graphs widely exist in real applications, such as traffic flow forecasting, social network analysis, brain network analysis, knowledge graph completion, and molecular graph modeling. Complex relationships between objects are usually described by edges in graphs. With rich information contained in edges, effectively modeling and mining graph data can boost the performance of existing machine learning algorithms. Recently, graph neural networks (GNNs) [1–5] have emerged as one of the most successful and popular graph learning algorithms because of their powerful ability in modeling graph data.

Although GNNs have been successfully applied in various domains [6–10], they typically adopt full-precision models to achieve good performance. Full-precision models do not fulfill the specific purposes in some application scenarios. For example, in the interactive setting of recommender systems, intelligent customer service may demand fast inference speed for decisions. Algorithms integrated into APPs may require low energy consumption on mobile phones. International corporations may have a goal of carbon-neutral to achieve. Since binary operations can enjoy hardware support (e.g., xnor and popcount operations), binary neural networks (BNNs) [11–13] provide a feasible approach for efficiency by converting multiplication between full-precision matrices into the multiplication between binary matrices.

Unfortunately, most existing studies of BNNs focus on binary convolutional neural networks (CNNs) [14, 15], while only a few studies [13, 16, 17] paid attention to binary GNNs. One significant difference between GNNs and CNNs is that each layer of GNNs involves multiplying three matrices. In contrast, each layer of CNNs only involves the multiplication between two matrices. Such a difference poses new challenges for the research of binary GNNs. For example, representative binary GNNs like Bi-GCN [16] and Bi-GNN [17] adopt XNOR-Net and its variants [18, 19] to binarize the multiplication between feature

* Corresponding author (email: liwujun@nju.edu.cn)

matrix and weight matrix at each layer. In the absence of quantization of the normalized adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ where N is the number of nodes, Bi-GCN and Bi-GNN still perform the multiplication with $\boldsymbol{A}$ in full-precision mode. BGN [13] first binarizes the multiplication between the feature matrix and weight matrix at each layer and then converts the multiplication between $\boldsymbol{A}$ and a full-precision matrix into addition operations by quantizing the values of $\boldsymbol{A}$ to $\{+1, 0, -1\}$. However, the multiplication with $\boldsymbol{A}$ is essentially performed in full-precision mode. Moreover, quantizing the values of $\boldsymbol{A}$ to $\{+1, 0, -1\}$ not only violates the similarity assumption but also drops the important interaction weights between nodes which are essential guarantees for good performance of GNN models [6, 20]. We can find that existing studies on binary GNNs only study the binarization of the multiplication between two matrices. We take matrix multiplication of $\boldsymbol{AXW}$ as an example and perform the multiplication with the order of $\boldsymbol{A}\cdot(\boldsymbol{XW})$. We need to perform matrix multiplication twice. Suppose each matrix multiplication has the same order of computation complexity. In that case, it is easy to verify that existing studies can only reduce half of the full-precision operations, obtaining a limited speedup of a factor of about 2. With a limited speedup, existing studies on binary GNNs are impractical for inference acceleration in real applications. Hence, further reducing inefficient full-precision operations is of great significance.

Since each graph convolution layer involves multiplying three matrices, re-quantization is a necessary procedure if we want to further reduce superfluous full-precision operations in the binarization of the graph convolution layer. Specifically, inefficient full-precision multiplication between at least two matrices in each graph convolution layer is unavoidable without re-quantization. Here, re-quantization means that we need to further quantize the result of the multiplication between any two matrices before multiplying with the third matrix. The challenges posed by re-quantization are mainly twofold. First, re-quantization in different computation orders yields different results and subsequently results in different performance. Since none of the existing studies have investigated re-quantization in binary GNNs, how much computation orders affect model performance remains unknown. Second, model capacity is further reduced, leading to a further decrease in model accuracy. It is easy to verify that re-quantization reduces the model capacity. How to increase the model capacity without additional computation overhead poses a challenge to re-quantization. Overall, how to solve the above challenges posed by re-quantization remains unexplored.

In this paper, we propose a novel method, called re-quantization based binary graph neural networks (RQBGN), to construct effective and efficient binary graph neural networks. The contributions of this paper are outlined as follows.

• We are the first to identify and investigate the new problem, namely re-quantization, which is a necessary procedure contributing to further reduction of superfluous inefficient full-precision operations.

• We identify that different computation orders in re-quantization have a significant impact on the performance of binary models. Furthermore, we find that there exists an optimal computation order that performs consistently better than the other one on various tasks.

• We introduce a mixture of experts (MoE) [21, 22] into RQBGN to increase the capacity of binary graph neural networks and therefore increase model accuracy. We show that RQBGN has fewer floating operations than other methods.

• Experiments on five benchmark datasets verify that computation orders have a significant impact on the performance of binary models. Moreover, RQBGN can outperform other baselines to achieve state-of-the-art performance.

2 Notations and preliminaries

This section first introduces notations and then briefly reviews preliminaries of graph neural networks, binary neural networks, and a mixture of experts.

2.1 Notations

Let boldface uppercase letters, such as $\boldsymbol{C}$, denote matrices and boldface lowercase letters, such as $\boldsymbol{c}$, denote vectors. Let $\boldsymbol{C}_{i*}$ and $\boldsymbol{C}_{*j}$ denote the ith row and the jth column of a matrix $\boldsymbol{C}$, respectively. C_{ij} denotes the element at the ith row and the jth column in $\boldsymbol{C}$. $\|\boldsymbol{C}\|_F$ denotes the Frobenius norm of $\boldsymbol{C}$. $\|\boldsymbol{C}\|_0$ denotes the number of non-zero entries in $\boldsymbol{C}$. Let $\mathcal{S}(\cdot)$ denote a sign function and $\mathcal{Q}(\cdot)$ denote a function for low-bit quantization.

Let $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ denote the normalized weight matrix of a graph $\mathcal{G}$, where N denotes the number of nodes. $A_{ij} = 1$ iff there is an edge from node i to j, otherwise $A_{ij} = 0$. Let $\boldsymbol{X} \in \mathbb{R}^{N\times u}$ denote the node feature matrix, where u denotes the dimension of the node feature. Let L denote the layer number of GNNs.

2.2 Graph neural networks

Graph neural networks (GNNs) are developed for the representation learning of nodes and graphs, with the goal that the learned representation can capture the complex relationships contained in graphs. While lots of representative GNN models [6,7,23] have been proposed, most of them are developed based on the message passing framework [24]. For convenience, we take one of the most representative models namely SAGE [7] as an example for illustration. SAGE can be formulated as follows:

$$\boldsymbol{H}^{(\ell)} = f\left(\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)} + \boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_2^{(\ell)}\right), \tag{1}$$

where $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, $f(\cdot)$ denotes an activation function, $\boldsymbol{W}_1^{(\ell)}$ and $\boldsymbol{W}_2^{(\ell)} \in \mathbb{R}^{r\times r}$ are learnable parameters. The first term of the right-hand side, which refers to a graph convolution operation, encodes the structure information of graph $\mathcal{G}$ into $\boldsymbol{H}^{(\ell)}$. $\boldsymbol{A}$ can be obtained by preprocessing the original adjacency matrix of $\mathcal{G}$ or via a parameterized function. For example, in the attention-based GNN models [23,25], $\boldsymbol{A}$ is obtained via a parameterized function of the node representation at each layer.

2.3 Binary neural networks

With increased deep learning applications in various domains, it is urgent to construct efficient deep learning models. BNNs are developed to construct efficient deep learning models with fast inference speed, low energy consumption, and low storage overhead. The study of BNNs mainly includes how to perform binarization [11, 12, 18, 19] and how to train binary models [26–28]. We take one of the representative methods namely XNOR-Net++ [19] to illustrate how to perform binarization.

$$\boldsymbol{X}\boldsymbol{W} \approx (\mathcal{S}(\boldsymbol{X}) \cdot \mathcal{S}(\boldsymbol{W})) \odot \boldsymbol{\alpha}, \tag{2}$$

where $\boldsymbol{X}$ is a feature matrix, $\boldsymbol{W}$ is a learnable weight matrix, $\odot$ denotes element-wise multiplication, and $\boldsymbol{\alpha}$ is a learnable scaling vector. Since the gradient of $\mathcal{S}(\cdot)$ is almost zero everywhere, approximation like straight through estimator (STE) [26] is used to approximate the gradient of the full-precision variable with that of the quantized variable. Details are shown as follows:

$$\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}\right]_{ij} \approx \begin{cases} \left[\frac{\partial \mathcal{L}}{\partial \mathcal{S}(\boldsymbol{W})}\right]_{ij}, & \text{if } -1 < W_{ij} < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathcal{L}$ denotes the loss of models. BNNs can effectively learn model parameters with approximation techniques for estimating gradients.

2.4 Mixture of experts

Big data and large models are a dominant trend in machine learning research. However, naively increasing the number of parameters (e.g., increasing the depth and the width of models) poses challenges to training and experimental equipment. To this end, the mixture of experts (MoE) models [21,22,29–34] is developed to increase the model capacity without largely increasing computation cost. Specifically, different inputs will activate different parameters (expert modules) via a routing strategy. Let $\boldsymbol{x}$ denote the input, $f_g(\cdot)$ denote a gating function, and $f_e^k(\cdot)$ denote the kth expert module. Then the MoE unit is formulated as follows:

$$\boldsymbol{x}_o = \sum_{k=1}^{K} f_g(\boldsymbol{x})_k f_e^k(\boldsymbol{x}), \tag{4}$$

where K denotes the number of expert modules, and $f_g(\boldsymbol{x})_k$ denotes the kth element of $f_g(\boldsymbol{x})$. By introducing K expert modules, the model capacity is correspondingly increased by a factor of K, which facilitates the absorption of information from big data. Generally, outputs of $f_g(\cdot)$ are sparse. The increased computation overhead mainly depends on the sparsity of $f_g(\cdot)$ and the cost of computing $f_g(\cdot)$.

3 RQBGN

In this section, we first formulate re-quantization and analyze the challenges caused by it. Then, we present our proposed solutions for the corresponding challenges. After that, we present the objective function of RQBGN. Finally, we analyze the computation complexity of RQBGN.

3.1 Formulation and analysis of re-quantization

Here, we take the GNN model defined in (1) as an example to formulate re-quantization. And we analyze the challenges caused by re-quantization in terms of computation orders and model capacity.

3.1.1 *Formulation of re-quantization*

Unlike existing BNNs that only need to binarize the multiplication between two matrices, operations of $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ in (1) involve binarization of multiplying three matrices. This means $\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ or $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}$ needs to be further binarized before multiplying with the third matrix. We define the procedure of binarizing $\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ or $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}$ in $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ as re-quantization. Obviously, there are two different orders for re-quantization. We adopt a similar framework in XNOR-Net++ [19] to formulate these two computation orders, which are shown as follows:

$$\boldsymbol{H}^{(\ell)} = f\left(\underline{\mathcal{S}\left(\underline{\mathcal{Q}(\boldsymbol{A})\mathcal{S}(\boldsymbol{H}^{(\ell-1)})}\right)\mathcal{S}(\boldsymbol{W}_1^{(\ell)})} \odot \boldsymbol{\alpha}_1^{(\ell)} + \underline{\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\mathcal{S}(\boldsymbol{W}_2^{(\ell)})} \odot \boldsymbol{\alpha}_2^{(\ell)}\right), \tag{5}$$

$$\boldsymbol{H}^{(\ell)} = f\left(\underline{\mathcal{Q}(\boldsymbol{A})\mathcal{S}\left(\underline{\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\mathcal{S}(\boldsymbol{W}_1^{(\ell)})}\right)} \odot \boldsymbol{\alpha}_1^{(\ell)} + \underline{\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\mathcal{S}(\boldsymbol{W}_2^{(\ell)})} \odot \boldsymbol{\alpha}_2^{(\ell)}\right), \tag{6}$$

where $\boldsymbol{\alpha}_1^{(\ell)} \in \mathbb{R}^{1\times r}$ and $\boldsymbol{\alpha}_2^{(\ell)} \in \mathbb{R}^{1\times r}$ are learnable scaling vectors. Eq. (5) indicates that $\mathcal{Q}(\boldsymbol{A})\mathcal{S}(\boldsymbol{H}^{(\ell-1)})$ is calculated first, while Eq. (6) indicates that $\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\mathcal{S}(\boldsymbol{W}_1^{(\ell)})$ is calculated first. As we have explained in Section 1, quantizing the values of $\boldsymbol{A}$ to $\{+1, 0, -1\}$ is unreasonable and leads to poor performance, which we will verify in Section 4. Instead, we quantize the values of $\boldsymbol{A}$ to 4 bits of precision. Specifically, we replace $\mathcal{S}(\boldsymbol{A})$ with $\mathcal{Q}(\boldsymbol{A})$, which is a uniform quantizer defined in [35]. Formally, $\mathcal{Q}(\boldsymbol{A})$ is defined as follows:

$$\mathcal{Q}(\boldsymbol{A}) = \lfloor \mathrm{Clip}(\boldsymbol{A}/s, 0, 2^4 - 1) \rceil \cdot s, \tag{7}$$

where s is a learnable step size, $\lfloor \cdot \rceil$ is a rounding operation, and $\mathrm{Clip}(\cdot, 0, 2^4 - 1)$ is a function that clips the input variable to the range $[0, 2^4 - 1]$. A visual illustration of the re-quantization process is presented in Figure 1.

3.1.2 *Analysis of re-quantization*

It is easy to verify that Eqs. (5) and (6) give different results and may lead to different performance. Hence, one challenge is to answer whether there is an optimal computation order that performs consistently better than the other computation order on various tasks. Furthermore, it is known that binary operations will largely reduce the effective capacity of a model and further lead to a decrease in model performance. Because re-quantization will further reduce the model capacity, another challenge is to answer how to effectively increase the model capacity of binary GNN models after re-quantization.

3.2 Proposed solutions

We explore solutions for the challenges introduced in re-quantization. First, we present the findings about the impact of different computation orders. Then, we introduce a mixture of experts to increase the capacity of binary GNN models. The main idea is to replace the binary linear layer with multiple binary linear layers and only activate one for each input via a routing function. A visual illustration of RQBGN is presented in Figure 2.

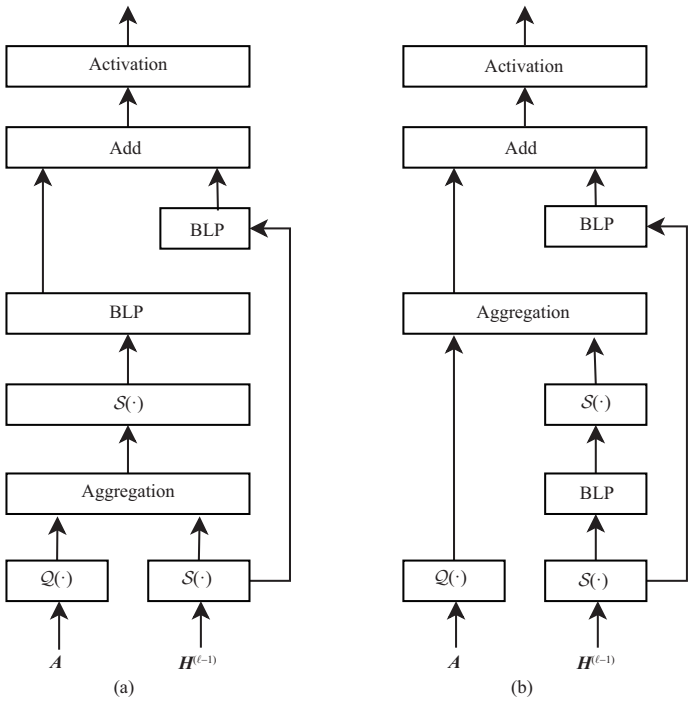

Figure 1 Visual illustration of the re-quantization process. $\boldsymbol{H}^{(\ell-1)}$ denotes the feature matrix of the ℓth layer. Aggregation denotes the operation of aggregating messages from neighbors for each center node. BLP denotes an operation of binary linear projection. (a) B1 denoting the process defined in (5). For B1, the first two matrices of $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ are multiplied first and then the obtained result is further binarized before multiplying the third matrix. (b) B2 denoting the process defined in (6). For B2, the latter two matrices of $\boldsymbol{A}\boldsymbol{H}^{(\ell-1)}\boldsymbol{W}_1^{(\ell)}$ are multiplied first and then the obtained result is further binarized before multiplying the first matrix.

3.2.1 *Impact of computation orders*

We perform extensive experiments to study the impact of computation orders. According to our experimental results in Section 4, we can draw two conclusions about the impact of computation orders. First, computation orders have a significant impact on the performance of binary GNN models. Second, the computation order defined in (5) is superior to the computation order defined in (6) in most cases. Taking ogbn-products as an example, we can find that the order in (5) outperforms the order in (6) by 4.3% when applied to SAGE and 3.2% when applied to GAT. Hence, RQBGN adopts computation order defined in (5).

3.2.2 *Mixture of experts in RQBGN*

Similar to [30], we extend $\boldsymbol{W}_1^{(\ell)}$ or $\boldsymbol{W}_2^{(\ell)}$ to a set of expert modules. We take a specific layer ℓ as an example for illustration. For convenience, let $\boldsymbol{W}$ denote $\boldsymbol{W}_1^{(\ell)}$ or $\boldsymbol{W}_2^{(\ell)}$ and $\boldsymbol{Z}$ denote $\mathcal{S}\left(\mathcal{Q}(\boldsymbol{A})\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\right)$ or $\mathcal{S}(\boldsymbol{H}^{(\ell-1)})$. Expert modules for $\boldsymbol{W}$ are formulated as follows. Similar to the definition in (4), we have

$$f_g(\boldsymbol{Z}_{i*})_k = \begin{cases} 1, & \text{if } k = \operatorname{argmax}\left(\operatorname{Softmax}\left((\boldsymbol{Z}_{i*}\mathcal{S}(\boldsymbol{\Theta})) \odot \boldsymbol{\alpha}_e\right)\right), \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

$$f_e^k(\boldsymbol{Z}_{i*}) = \boldsymbol{Z}_{i*}\mathcal{S}(\boldsymbol{W}_k), \tag{9}$$

$$\tilde{\boldsymbol{Z}}_{i*} = \sum_{k=1}^{K} f_g(\boldsymbol{Z}_{i*})_k f_e^k(\boldsymbol{Z}_{i*}), \quad i = 1, \ldots, N, \tag{10}$$

where $\mathcal{W} = \{\boldsymbol{W}_k \in \mathbb{R}^{r\times r}\}_{k=1}^K$, $\boldsymbol{\Theta} \in \mathbb{R}^{r\times K}$, and $\boldsymbol{\alpha}_e \in \mathbb{R}^{1\times K}$ are learnable parameters, K is the number of experts, $f_e^k(\cdot)$ denotes the kth expert module with $\boldsymbol{W}_k$ as its parameter, $f_g(\cdot)$ denotes a gating function with $\boldsymbol{\Theta}$ and $\boldsymbol{\alpha}_e$ as its parameters and it only activates one expert for an input, and $\tilde{\boldsymbol{Z}}_{i*}$ denotes the corresponding output of $\boldsymbol{Z}_{i*}$. The above forward process can be abstracted as follows when given $\boldsymbol{Z}$ as

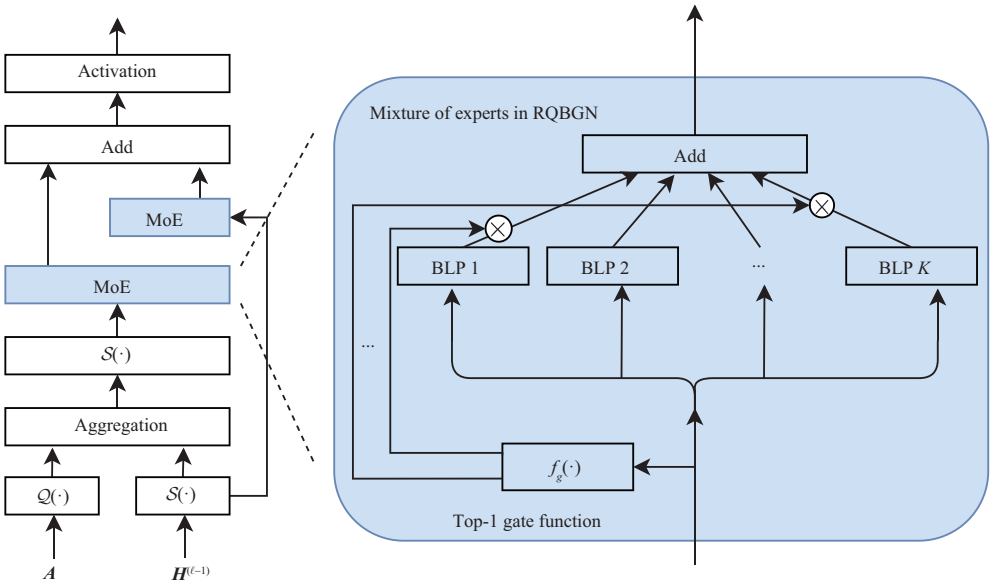

Figure 2 Visual illustration of RQBGN. MoE denotes the operation defined in (11). $f_g(\cdot)$ is defined in (8). Top-1 gate means that only the gate with a maximum score will be activated.

input:

$$\tilde{\boldsymbol{Z}} = \mathrm{MoE}(\boldsymbol{Z}; \mathcal{W}, \boldsymbol{\Theta}, \boldsymbol{\alpha}_e), \tag{11}$$

where $\boldsymbol{Z}$ denotes input, $\tilde{\boldsymbol{Z}}$ denotes the corresponding output of $\boldsymbol{Z}$, and $\mathcal{W}$, $\boldsymbol{\Theta}$, and $\boldsymbol{\alpha}_e$ denote the learnable parameters of MoE unit in RQBGN. Here, each expert in $\mathcal{W}$ has a different value, representing experts' different knowledge. As in [30, 33], random initialization for each $\boldsymbol{W}_k$ is the most common practice. Empirically, it is widely verified that the expert knowledge set $\mathcal{W}$ with random initialization is effective enough to increase the model capacity [29,32,33]. Hence, random initialization provides a practical expert knowledge set, which reaches MoE's primary goal of increasing the model capacity. The gating function $f_g(\cdot)$ defined in (8) is also binary, where matrix multiplication is implemented by binary operations. Since $K \ll r$, the computation cost introduced by the gating function is negligible. Moreover, since a node activates only one expert with the maximum gating value each time (see (8)), the computation cost of (11) is almost the same as that of (5). Similar to $\mathcal{S}(\cdot)$, we use STE to estimate the gradient of argmax$(\cdot)$. Then an RQBGN layer is defined as follows:

$$\begin{aligned}\boldsymbol{H}^{(\ell)} = f\Bigg(&\mathrm{MoE}\left(\mathcal{S}\big(\mathcal{Q}(\boldsymbol{A})\mathcal{S}(\boldsymbol{H}^{(\ell-1)})\big); \mathcal{W}_1^{(\ell)}, \boldsymbol{\Theta}_1^{(\ell)}, \boldsymbol{\alpha}_{1,e}^{(\ell)}\right) \odot \boldsymbol{\alpha}_1^{(\ell)} \\ &+ \mathrm{MoE}\left(\mathcal{S}(\boldsymbol{H}^{(\ell-1)}); \mathcal{W}_2^{(\ell)}, \boldsymbol{\Theta}_2^{(\ell)}, \boldsymbol{\alpha}_{2,e}^{(\ell)}\right) \odot \boldsymbol{\alpha}_2^{(\ell)}\Bigg).\end{aligned} \tag{12}$$

Since each node i activates only one expert each time (see (8)), Eq. (12) has the same order of computation complexity as (5).

Note that there are differences between RQBGN and the work in [34]. First, in RQBGN, the gate function in (8) is computed via binary matrix multiplication, while the gate function in [34] is computed via full-precision matrix multiplication. Therefore, the gate function in [34] introduces additional inefficient full-precision operations and the value of K cannot be too large. Second, RQBGN focuses on binarizing GNNs, while the work in [34] focuses on binarizing CNNs.

3.3 Objective function

Let $\hat{\boldsymbol{Y}} = \boldsymbol{H}^{(L)}$ denote the output of RQBGN. The objective function for RQBGN is formulated as follows:

$$\min_{\mathcal{P}} \sum_{i \in \mathcal{V}_{\mathrm{tr}}} \sum_{c} -Y_{ic} \log \hat{Y}_{ic} + \lambda/2 \cdot \sum_{\boldsymbol{P} \in \mathcal{P}} \|\boldsymbol{P}\|_F^2, \tag{13}$$

Table 1 Complexity analysis of operations. The analysis is mainly based on the GNN model in (1), and we only show the complexity of layer ℓ [a)]

Method	BOPs	FLOPs
GNN (FP)	0	$\mathcal{O}(2Nr^2 + 2\|\boldsymbol{A}\|_0 \cdot r + 3Nr)$
Bi-GNN [17]	$\mathcal{O}(2Nr^2)$	$\mathcal{O}(2\|\boldsymbol{A}\|_0 \cdot r + 3Nr)$
BGN [13]	$\mathcal{O}(2Nr^2)$	$\mathcal{O}(\|\boldsymbol{A}\|_0 \cdot r + 3Nr)$
RQBGN (ours)	$\mathcal{O}(2Nr^2 + 4\|\boldsymbol{A}\|_0 \cdot r + NrK)$	$\mathcal{O}(3Nr + NK)$

a) Numbers in column BOPs represent the numbers of binary operations. Numbers in column FLOPs indicate the numbers of floating operations. FP indicates full-precision. r is the dimension of node representation. K is the number of experts.

where $\mathcal{P} = \{\boldsymbol{W}_{1,1}^{(\ell)}, \ldots, \boldsymbol{W}_{1,K}^{(\ell)}, \boldsymbol{W}_{2,1}^{(\ell)}, \ldots, \boldsymbol{W}_{2,K}^{(\ell)}, \boldsymbol{\Theta}_1^{(\ell)}, \boldsymbol{\Theta}_2^{(\ell)}, \boldsymbol{\alpha}_1^{(\ell)}, \boldsymbol{\alpha}_2^{(\ell)}, \boldsymbol{\alpha}_{1,e}^{(\ell)}, \boldsymbol{\alpha}_{2,e}^{(\ell)}\}_{\ell=1}^{L}$ denote the learnable parameters in (12), λ is a hyper-parameter for the Frobenius norm regularization of $\mathcal{P}$, and $\mathcal{V}_{\text{tr}}$ denotes the training set.

3.4 Complexity analysis

This subsection compares the number of binary operations (BOPs) and floating operations (FLOPs) of different methods in the forward process. The comparison is mainly based on the GNN model defined in (1). The results are summarized in Table 1, from which we can draw the following conclusions. First, the sums of BOPs and FLOPs for different methods are approximately equal. Second, since NK is much smaller than $\|\boldsymbol{A}\|_0$ and $\|\boldsymbol{A}\|_0 \cdot r$ is much larger than Nr, we can find that RQBGN has much fewer FLOPs than other methods. In sum, RQBGN converts most of the inefficient FLOPs into BOPs. Consequently, RQBGN is more efficient than other methods.

4 Experiments

We evaluate RQBGN and other baselines on five benchmark datasets by binarizing two base GNN models, including SAGE [7] and GAT [23]. All methods are implemented with Pytorch [36] and Pytorch-Geometric Library [37]. All experiments are run on an NVIDIA RTX A6000 GPU server with 48 GB of graphics memory.

4.1 Datasets

Datasets for evaluation include ogbn-products, ogbn-papers100M, ogbn-proteins, ogbg-molhiv, and ogbg-molpcba [38][1)]. The first three datasets are used for node classification and the latter two are used for graph classification. The first three datasets are evaluated in a transductive setting and the latter two are evaluated in an inductive setting. ogbn-products is extracted from Amazon products co-purchasing network [39] and the task is to predict the category of a product. ogbn-papers100M is a paper citation network data extracted from Microsoft academic graph (MAG) [40] and the task is to predict the subject areas of papers that are published in arxiv. ogbn-proteins is a protein-protein association network data [41] in which edges indicate biologically meaningful associations between proteins, including physical interactions, homology, or co-expression. The task for ogbn-proteins is to predict the presence of protein functions. ogbg-molhiv and ogbg-molpcba are molecular property prediction datasets extracted from the MOLECULENET [42]. Each graph in ogbg-molhiv and ogbg-molpcba represents a molecule, in which nodes represent atoms and edges are chemical bonds. The statistics of datasets are summarized in Table 2.

4.2 Baselines

We choose SAGE [7] and GAT [23] as base GNN models for binarization. SAGE is one of the most representative non-attention-based GNN models and GAT is one of the most representative attention-based GNN models. Specifically, SAGE applies trainable aggregation functions to aggregate neighbor information. GAT adopts an attention mechanism to learn aggregation weights for neighboring nodes adaptively.

We mainly compare RQBGN with two state-of-the-art baselines, including BGN [13] and Bi-GNN [17]. Note that none of the existing studies have investigated the problem of re-quantization, and hence existing

1) https://ogb.stanford.edu/.

Table 2 Statistics of datasets

Node classification datasets				Graph classification datasets		
	ogbn-products	ogbn-papers100M	ogbn-proteins		ogbg-molhiv	ogbg-molpcba
#Nodes	2449029	111059956	132534	#Graphs	41127	437929
#Edges	61859140	1615685872	39561252	#Nodes per graph	25.5	26.0
#Features/node	100	128	8	#Edges per graph	27.5	28.1
#Classes	47	172	112	#Classes	2	2
#Training nodes	196615	1207179	86619	#Training graphs	32901	350343
#Validation nodes	39323	125265	21236	#Validation graphs	4113	43793
#Test nodes	2213091	214338	24679	#Test graphs	4113	43793
Task type	Multi-class	Multi-class	Multi-label	Task type	Binary-class	Binary-class
Metric	Accuracy	Accuracy	ROC-AUC	Metric	ROC-AUC	Average precision (AP)

Table 3 Hyper-parameters[a)]

	L	r	T	B	p	λ	η
ogbn-products	5	128	200	32×1024	0.15	5E−6	0.01
ogbn-papers100M	3	256	200	32×1024	0.1	1E−7	0.01
ogbn-proteins	7	128	1200	32×1024	0.1	0	0.01
ogbg-molhiv	5	256	100	512	0	1E−5	0.002
ogbg-molpcba	5	256	100	8×1024	0.2	0	0.01

a) L is layer number. r is a hidden dimension. T is the maximum number of epochs. B is mini-batch size. p is the probability of dropout. λ is the regularization coefficient for parameters. η is the learning rate. Note that values of p listed in the table are for binary GNN models, while different values of p will be set for full-precision GNN models.

studies include lots of inefficient full-precision operations. For a fair comparison, implementations of all methods, including RQBGN, only differ in the binarization process. To analyze the effectiveness of different binarization strategies, additional training techniques in Bi-GNN [17], like knowledge distillation and multi-stage training, are not adopted in our experiments.

The evaluation metrics for each dataset are presented in Table 2. Note that the evaluation metrics for all datasets are provided by the researchers in [38]. Since most recent advances in GNNs follow the same evaluation metrics in [38], we also adopt the same evaluation metrics.

4.3 Implementation

Since training GNN models on node classification datasets will incur exponential computation and memory complexity, we adopt BNS [43] as our neighbor sampling strategy to reduce computation and memory complexity. Furthermore, we apply GraphNorm [44] to normalize the hidden representation for accelerating the training process.

For most hyper-parameters, we first tune them with full-precision GNN models on the validation set and then set the same values for binary GNN models. Unless otherwise stated, the following settings are set to be the same for SAGE and GAT, the same for full-precision models and binary ones, and the same for RQBGN and other binarization strategies. For some common hyper-parameters, including layer number L, hidden dimension r, maximum epoch T, mini-batch size B, probability of dropout p, regularization coefficient for parameters λ, and learning rate η, we list their values in Table 3. Since we adopt BNS for training GNN models on node classification datasets, three additional hyper-parameters are needed to be set, i.e., $\tilde{s}_0$, $\tilde{s}_1$, δ. Specifically, $\tilde{s}_0$ is the number of sampled neighbors for nodes at the output layer, $\tilde{s}_1$ is the number of sampled neighbors for nodes at other layers, and δ is the ratio of blocked neighbors. For ogbn-products, $\tilde{s}_0 = 10$, $\tilde{s}_1 = 4$, $\delta = 1/2$. For ogbn-papers100M, $\tilde{s}_0 = 20$, $\tilde{s}_1 = 4$, $\delta = 1/2$. For ogbn-proteins, $\tilde{s}_0 = 48$, $\tilde{s}_1 = 24$, $\delta = 2/3$. For ogbn-products and ogbn-papers100M, we augment node features with masked label information as in [25]. For ogbn-proteins, edge features are utilized as in [25]. Since our goal is not to achieve high accuracy for the full-precision model but to verify the effectiveness of different binarization strategies, we set the number of heads in GAT to 1. K for RQBGN is selected from $\{2, 4, 6, 8, 16\}$ according to the performance on the validation set. We use Adam [45] for optimization. We adopt a cosine annealing schedule [46] on the learning rate to improve the convergence rate.

Table 4 Impact of computation orders[a)]

Methods	Accuracy (%) ↑ or ROC-AUC (%) ↑ or AP (%) ↑				
	ogbn-products	ogbn-papers100M	ogbn-proteins	ogbg-molhiv	ogbg-molpcba
SAGE (FP)	81.49 ± 0.19	65.53 ± 0.10	85.14 ± 0.25	78.78 ± 0.55	23.62 ± 0.20
SAGE-B2	74.81 ± 0.73	61.32 ± 0.17	81.60 ± 0.58	76.39 ± 1.27	16.23 ± 0.19
SAGE-B1	**79.10 ± 0.29**	**62.28 ± 0.10**	**81.74 ± 0.51**	**77.71 ± 1.41**	**18.73 ± 0.14**
GAT (FP)	82.07 ± 0.16	65.53 ± 0.15	85.21 ± 0.28	77.68 ± 0.88	21.86 ± 0.48
GAT-B2	76.81 ± 0.34	61.34 ± 0.10	81.23 ± 0.37	74.94 ± 1.09	14.62 ± 0.17
GAT-B1	**80.05 ± 0.24**	**62.01 ± 0.21**	**81.96 ± 0.39**	**76.19 ± 1.01**	**18.23 ± 0.05**

a) FP denotes the full-precision models. B1 denotes the binary models performing forward propagation with the computation order defined in (5). B2 denotes the binary models performing forward propagation with the computation order defined in (6). Boldface letters denote the best results between order B1 and order B2.

Table 5 Comparison with baselines[a)]

Methods	Accuracy (%) ↑ or ROC-AUC (%) ↑ or AP (%) ↑				
	ogbn-products	ogbn-papers100M	ogbn-proteins	ogbg-molhiv	ogbg-molpcba
SAGE (FP)	81.49 ± 0.19	65.53 ± 0.10	85.14 ± 0.25	78.78 ± 0.55	23.62 ± 0.20
BGN	73.18 ± 0.90	**63.10 ± 0.16**	76.81 ± 0.71	76.74 ± 0.92	19.74 ± 0.15
Bi-GNN	79.81 ± 0.45	63.03 ± 0.17	82.52 ± 0.40	**78.85 ± 0.84**	**20.08 ± 0.53**
RQBGN (ours)	**80.32 ± 0.16**	62.64 ± 0.08	**83.17 ± 0.26**	78.41 ± 0.76	19.48 ± 0.26
GAT (FP)	82.07 ± 0.16	65.53 ± 0.15	85.21 ± 0.28	77.68 ± 0.88	21.86 ± 0.48
BGN	78.15 ± 0.44	62.35 ± 0.11	68.36 ± 2.84	75.44 ± 1.31	17.83 ± 0.55
Bi-GNN	81.14 ± 0.10	62.79 ± 0.12	83.00 ± 0.25	77.06 ± 1.17	**18.84 ± 0.20**
RQBGN (ours)	**81.49 ± 0.20**	**62.91 ± 0.12**	**83.04 ± 0.10**	**77.27 ± 1.23**	18.74 ± 0.44

a) Boldface letters denote the best results among RQBGN and binary baselines.

4.4 Results

4.4.1 *Impact of computation orders*

Let B1 denote the computation order defined in (5), and B2 denote the computation order defined in (6). To compare B1 and B2, we conduct experiments with the base binary GNN models. The results are summarized in Table 4. We can draw the following conclusions. First, computation orders have a significant impact on the performance of binary GNN models. For example, different computation orders have an accuracy gap of 4.3% on ogbn-products, 1.0% on ogbn-papers100M, 0.7% on ogbn-proteins, 1.3% on ogbg-molhiv, and 3.6% on ogbg-molpcba. Second, computation order B1 can perform better than B2 on all datasets. Third, computation order B1 consistently outperforms B2 when applied to both SAGE and GAT. The above conclusions show that binarizing GNN models with computation order B1 is better.

4.4.2 *Comparison with baselines*

We compare our RQBGN with BGN [13] and Bi-GNN [17]. Results are summarized in Table 5. We can draw the following conclusions. First, RQBGN can achieve comparable performance to state-of-the-art baseline Bi-GNN on all datasets. For example, gaps between RQBGN and Bi-GNN on all datasets fluctuate in a small range of $[-0.60\%, 0.65\%]$ when all methods are applied to SAGE and in a range of $[-0.07\%, 0.35\%]$ when all methods are applied to GAT. Moreover, RQBGN outperforms Bi-GNN in 6 out of 10 cases. Since RQBGN has much fewer FLOPS than Bi-GNN (see Subsection 3.4), RQBGN demonstrates its effectiveness by achieving accuracy comparable to that of Bi-GNN. Specifically, RQBGN can effectively increase binary GNN models' capacity and alleviate the problem of reduced model capacity caused by re-quantization. Second, we observe that BGN performs worse than other methods in most cases and the gaps are relatively large. For example, gaps between BGN and other methods fluctuate in a large range of $[-7.14\%, 0.46\%]$ when all methods are applied to SAGE and in a range of $[-14.68\%, -0.56\%]$ when all methods are applied to GAT. In particular, gaps between BGN and RQBGN reach -7.1% on ogbn-products and -14.7% on ogbn-proteins. Moreover, BGN performs worse than Bi-GNN and RQBGN in 9 out of 10 cases. This point shows that quantizing the values of $\boldsymbol{A}$ to $\{+1, 0, -1\}$ leads to poor performance because a value of -1 violates the similarity assumption and values of $\{+1, 0, -1\}$ drop the important interaction weights between nodes. In sum, RQBGN can achieve comparable performance to

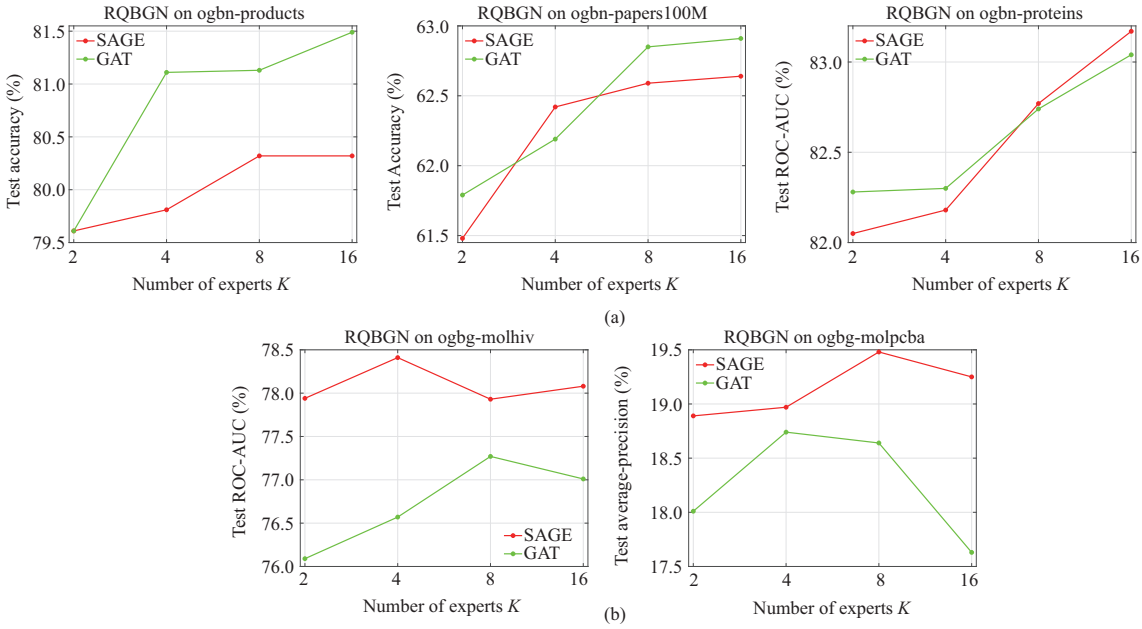

Figure 3 Effect of the number of experts K. SAGE and GAT are two base models for binarization. (a) Node classification datasets; (b) graph classification datasets

state-of-the-art baseline Bi-GNN when applied to both SAGE and GAT.

4.4.3 *Effect of the number of experts*

We perform experiments to analyze the effect of the number of experts. The results are summarized in Figure 3. We can find that the performance of binary GNN models improves in general as K increases in the range of $[2, 12]$. The results show that a mixture of experts can effectively increase the capacity of binary GNN models. As in [30, 33], this point also verifies that an expert knowledge set with random initialization is effective enough to increase the capacity of binary models.

5 Conclusion

In this paper, we propose a novel method, called RQBGN, to construct effective and efficient binary graph neural networks. To the best of our knowledge, we are the first to identify and investigate the new problem, namely re-quantization, which is a necessary procedure contributing to the further reduction of superfluous inefficient full-precision operations. In RQBGN, we identify that computation orders can significantly impact the performance of binary models. Furthermore, we find that there exists an optimal computation order that performs consistently better than the other one on various tasks. To ensure the effectiveness of RQBGN, we introduce a mixture of experts to boost the model capacity. We show that RQBGN has fewer FLOPs than other methods, and hence is more efficient than other methods. Experiments on five benchmark datasets demonstrate the effectiveness of RQBGN.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2020YFA0713901), National Natural Science Foundation of China (Grant Nos. 61921006, 62192783), and Fundamental Research Funds for the Central Universities (Grant No. 020214380108).

References

1 Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains. In: Proceedings of IEEE International Joint Conference on Neural Networks, Montreal, 2005. 729–734

2 Bruna J, Zaremba W, Szlam A, et al. Spectral networks and locally connected networks on graphs. In: Proceedings of International Conference on Learning Representations, Banff, 2014

3 Cao W M, Zheng C T, Yan Z Y, et al. Geometric deep learning: progress, applications and challenges. Sci China Inf Sci, 2022, 65: 126101

4 Qian X H. Graph processing and machine learning architectures with emerging memory technologies: a survey. Sci China Inf Sci, 2021, 64: 160401

5 Chen J X, Lin G Q, Chen J X, et al. Towards efficient allocation of graph convolutional networks on hybrid computation-in-memory architecture. Sci China Inf Sci, 2021, 64: 160409

6 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference on Learning Representations, Toulon, 2017

7 Hamilton W L, Ying Z T, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 1024–1034

8 Torng W, Altman R B. Graph convolutional neural networks for predicting drug-target interactions. J Chem Inf Model, 2019, 59: 4131–4149

9 Li M Z, Zhu Z X. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In: Proceedings of AAAI Conference on Artificial Intelligence, Virtual Conference, 2021. 4189–4196

10 Monti F, Bronstein M M, Bresson X. Geometric matrix completion with recurrent multi-graph neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 3697–3707

11 Courbariaux M, Bengio Y, David J P. BinaryConnect: training deep neural networks with binary weights during propagations. In: Proceedings of Advances in Neural Information Processing Systems, Montreal, 2015. 3123–3131

12 Hubara I, Courbariaux M, Soudry D, et al. Binarized neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Barcelona, 2016. 4107–4115

13 Wang H C, Lian D F, Zhang Y, et al. Binarized graph neural network. World Wide Web, 2021, 24: 825–848

14 LeCun Y, Boser B E, Denker J S, et al. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989, 1: 541–551

15 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Lake Tahoe, 2012. 1106–1114

16 Wang J F, Wang Y H, Yang Z, et al. Bi-GCN: binary graph convolutional network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Virtual Conference, 2021. 1561–1570

17 Bahri M, Bahl G, Zafeiriou S. Binary graph neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Virtual Conference, 2021. 9492–9501

18 Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Proceedings of European Conference on Computer Vision, Amsterdam, 2016. 525–542

19 Bulat A, Tzimiropoulos G. XNOR-Net++: improved binary neural networks. In: Proceedings of British Machine Vision Conference, Cardiff, 2019. 62

20 Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of Advances in Neural Information Processing Systems, Barcelona, 2016. 3837–3845

21 Jacobs R A, Jordan M I, Nowlan S J, et al. Adaptive mixtures of local experts. Neural Computation, 1991, 3: 79–87

22 Yuksel S E, Wilson J N, Gader P D. Twenty years of mixture of experts. IEEE Trans Neural Netw Learn Syst, 2012, 23: 1177–1193

23 Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of International Conference on Learning Representations, Vancouver, 2018

24 Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry. In: Proceedings of International Conference on Machine Learning, Sydney, 2017. 1263–1272

25 Shi Y S, Huang Z J, Feng S K, et al. Masked label prediction: unified message passing model for semi-supervised classification. In: Proceedings of International Joint Conference on Artificial Intelligence, Virtual Conference, 2021. 1548–1554

26 Bengio Y, Leonard N, Courville A C. Estimating or propagating gradients through stochastic neurons for conditional computation. 2013. ArXiv:1308.3432

27 Gong R H, Liu X L, Jiang S H, et al. Differentiable soft quantization: bridging full-precision and low-bit neural networks. In: Proceedings of IEEE/CVF International Conference on Computer Vision, Seoul, 2019. 4851–4860

28 Martinez B, Yang J, Bulat A, et al. Training binary neural networks with real-to-binary convolutions. In: Proceedings of International Conference on Learning Representations, Virtual Conference, 2020

29 Gross S, Ranzato M A, Szlam A. Hard mixtures of experts for large scale weakly supervised vision. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, 2017. 5085–5093

30 Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously large neural networks: the sparsely-gated mixture-of-experts layer. In: Proceedings of International Conference on Learning Representations, Toulon, 2017

31 Roller S, Sukhbaatar S, Szlam A, et al. Hash layers for large sparse models. In: Proceedings of Advances in Neural Information Processing Systems, Virtual Conference, 2021. 17555–17566

32 Fedus W, Zoph B, Shazeer M. Switch Transformers: scaling to trillion parameter models with simple and efficient sparsity. 2021. ArXiv:2101.03961

33 Lewis M, Bhosale S, Dettmers T, et al. BASE layers: simplifying training of large, sparse models. In: Proceedings of International Conference on Machine Learning, Virtual Conference, 2021. 6265–6274

34 Bulat A, Martinez B, Tzimiropoulos G. High-capacity expert binary networks. In: Proceedings of International Conference on Learning Representations, Virtual Conference, 2021

35 Esser S K, McKinstry J L, Bablani D, et al. Learned step size quantization. In: Proceedings of International Conference on Learning Representations, Virtual Conference, 2020

36 Paszke A, Gross S, Massa F, et al. PyTorch: an imperative style, high-performance deep learning library. In: Proceedings of Advances in Neural Information Processing Systems, Vancouver, 2019. 8024–8035

37 Fey M, Jan E L, Bablani D, et al. Fast graph representation learning with PyTorch Geometric. In: Proceedings of International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds, New Orleans, 2019

38 Hu W H, Fey M, Zitnik M, et al. Open graph benchmark: datasets for machine learning on graphs. In: Proceedings of Advances in Neural Information Processing Systems, Virtual Conference, 2020. 22118–22133

39 Chiang W L, Liu X Q, Si S, et al. Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, 2019. 257–266

40 Wang K S, Shen Z H, Huang C Y, et al. Microsoft academic graph: when experts are not enough. Quantitative Sci Studies, 2020, 1: 396–413

41 Szklarczyk D, Gable A L, Lyon D, et al. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Res, 2019, 47: 607–613

42 Wu Z Q, Ramsundar B, Feinberg E N, et al. MoleculeNet: a benchmark for molecular machine learning. Chem Sci, 2018, 9: 513–530

43 Yao K L, Li W J. Blocking-based neighbor sampling for large-scale graph neural networks. In: Proceedings of International Joint Conference on Artificial Intelligence, Virtual Conference, 2021. 3307–3313

44 Cai T L, Luo S J, Xu K, et al. GraphNorm: a principled approach to accelerating graph neural network training. In: Proceedings of International Conference on Machine Learning, Virtual Conference, 2021. 1204–1215

45 Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of International Conference on Learning Representations, San Diego, 2015

46 Loshchilov I, Hutter F. SGDR: stochastic gradient descent with warm restarts. In: Proceedings of International Conference on Learning Representations, Toulon, 2017