FULL-PRECISION FREE BINARY GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Binary neural networks have become a promising research topic due to their fast inference speed and low energy consumption advantages. However, most existing works focus on binary convolutional neural networks, while less attention has been paid to binary graph neural networks. A common drawback of existing works on binary graph neural networks is that they still include lots of inefficient full-precision operations and hence are not efficient enough. In this paper, we propose a novel method, called full-precision free binary graph neural networks (FFBGN), to avoid full-precision operations for binarizing graph neural networks. To address the challenges introduced by *re-quantization* which is a necessary procedure for avoiding full-precision operations, in FFBGN we first study the impact of different computation orders to find an effective computation order and then introduce mixture of experts to increase the model capacity. Experiments on three large-scale datasets show that performing re-quantization in different computation orders significantly impacts the performance of binary graph neural network models, and FFBGN can outperform other baselines to achieve state-of-the-art performance.

1 INTRODUCTION

Graphs widely exist in real applications, such as traffic networks, social networks, brain networks, knowledge graphs, and molecular graphs. 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) (Gori et al., 2005; Bruna et al., 2014) 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 (Kipf & Welling, 2017; Hamilton et al., 2017; Torng & Altman, 2019; Li & Zhu, 2021; Monti et al., 2017), 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 in 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) (Courbariaux et al., 2015; Hubara et al., 2016; Martinez et al., 2020) provide a feasible approach for efficiency by converting multiplication of full-precision matrices into multiplication of binary matrices.

Unfortunately, most existing works of BNNs focus on binary convolutional neural networks (CNNs) (LeCun et al., 1989; Krizhevsky et al., 2012), while only a few works (Wang et al., 2021b; Bahri et al., 2021; Wang et al., 2021a) pay attention to binary GNNs. Specifically, Bi-GCN (Wang et al., 2021b) and Bi-GNN (Bahri et al., 2021) adopt XNOR-Net and its variants (Rastegari et al., 2016; Bulat & Tzimiropoulos, 2019) to binarize the multiplication between feature matrix and weight matrix at each layer. In the absence of quantization of the normalized adjacency matrix ($\mathbf{A} \in \mathbb{R}^{N \times N}$, N is the number of nodes), multiplication with $\mathbf{A}$ is still performed in full-precision mode. BGN (Wang et al., 2021a) first binarizes the multiplication between feature matrix and weight matrix at each layer and then convert the multiplication between $\mathbf{A}$ and a full-precision matrix into addition operations by quantizing the values of $\mathbf{A}$ to $\{+1, 0, -1\}$. However,

the multiplication with $\mathbf{A}$ is essentially performed in full-precision mode. Moreover, quantizing the values of $\mathbf{A}$ to $\{+1, 0, -1\}$ not only violates both the similarity assumption and also drops the important interaction weights between nodes, which are essential guarantees for good performance of GNN models (Defferrard et al., 2016; Kipf & Welling, 2017)). We can find that existing works on binary GNNs only study the binarization of the multiplication of two matrices. Consequently, they still include lots of inefficient full-precision operations and hence are not efficient enough.

Since each graph convolution layer involves the multiplication of three matrices, *re-quantization* is a necessary procedure if we want to avoid full-precision operations in the binarization of graph convolution layer. Here, *re-quantization* means that we need to further quantize the result of the multiplication of 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 yield different results and, as a result, different performance. Second, model capacity is further reduced, leading to a further decrease in model accuracy. How to address the challenges introduced by re-quantization remains unexplored.

In this paper, we propose a novel method, called full-precision free binary graph neural networks (FFBGN) [1], 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 for avoiding full-precision operations in binary graph neural networks.
- We conduct an extensive experimental analysis of computation orders in re-quantization and identify that computation orders have a significant impact on the performance of binary models.
- We introduce mixture of experts (MoE) (Jacobs et al., 1991; Yüksel et al., 2012) into FFBGN to increase the capacity of binary graph neural networks, and hence increase the model accuracy.
- Experiments on three large-scale datasets show that FFBGN can outperform other baselines to achieve state-of-the-art performance.

2 Notations and Preliminaries

In this section, we first introduce notations, and then briefly review preliminaries of graph neural networks, binary neural networks, and mixture of experts.

2.1 Notations

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

We use $\mathbf{A} \in \mathbb{R}^{N \times N}$ to denote the normalized weight matrix of a graph $\mathcal{G}$, where N denotes the number of nodes. $A_{ij} = 0$ denotes no edge between node i and node j, otherwise there is an edge between them. We use $\mathbf{X} \in \mathbb{R}^{N \times u}$ to denote the node feature matrix, where u denotes the dimension of node feature. We use L to 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

[1] Full-precision free means that no full-precision operations exist in the matrix multiplication of the graph convolution operation. Please note that the whole model still has few full-precision operations related to the scaling factors, which can almostly be omitted compared with other operations.

graphs. While lots of advanced GNN models (Kipf & Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018) have been proposed, most of them are developed based on the message passing framework (Gilmer et al., 2017). In brief, GNNs mainly consist of two kind of operations, message passing and message updating. We take the model in (Hamilton et al., 2017) as an example for introduction:

$$\mathbf{H}^{(\ell)} = f(\mathbf{A}\mathbf{H}^{(\ell-1)}\mathbf{W}_g^{(\ell)} + \mathbf{H}^{(\ell-1)}\mathbf{W}_s^{(\ell)}), \tag{1}$$

where $\mathbf{H}^{(0)} = \mathbf{X}$, $f(\cdot)$ denotes the activation function, $\mathbf{W}_g^{(\ell)}$ and $\mathbf{W}_s^{(\ell)} \in \mathbb{R}^{r\times r}$ are learnable parameters. The first term of the right-hand side, which refers to a graph convolution operation, ensures that the structure information of graph $\mathcal{G}$ can be encoded in $\mathbf{H}^{(\ell)}$, while the second term of the right-hand side balances information between neighbors and center nodes. $\mathbf{A}$ can be obtained by preprocessing the original adjacency matrix of $\mathcal{G}$, or it can be obtained via a parameterized function. For example, in the attention-based GNN models (Velickovic et al., 2018; Shi et al., 2021), $\mathbf{A}$ is calculated 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 concerns of BNNs mainly include how to perform binarization (Courbariaux et al., 2015; Hubara et al., 2016; Rastegari et al., 2016; Bulat & Tzimiropoulos, 2019) and how to train binary models (Bengio et al., 2013; Gong et al., 2019; Martinez et al., 2020). We take one of the representative methods, namely XNOR-Net++ (Bulat & Tzimiropoulos, 2019), to illustrate how to perform binarization.

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

where $\mathbf{X}$ is a feature matrix, $\mathbf{W}$ is a learnable weight matrix. $\odot$ denotes element-wise multiplication. $\boldsymbol{\alpha}$ is a learnable scaling factor. Since the gradient of $\mathcal{S}(\cdot)$ is almost zero everywhere, approximation like straight through estimator (STE) (Bengio et al., 2013) is used to approximate the gradient of the full-precision variable with that of quantized variable. Details are as follows.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \approx \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathcal{S}(\mathbf{W})}, & \text{if } -1 < \mathbf{W} < 1 \\ 0, & \text{otherwise} \end{cases}$$

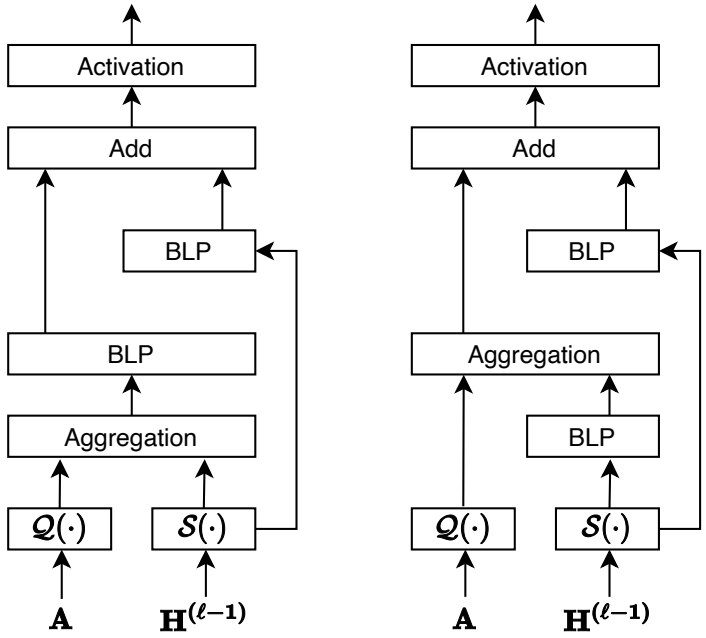
where $\mathcal{L}$ denotes the loss of models. BNNs can effectively learn model parameters with approximation techniques for estimating gradients.

2.4 Mixture of Experts

Big data and large models are a dominant trend of machine learning research. However, naively increasing the number of parameters (e.g., increasing the depth and the width of models) poses great challenges to training and experimental equipment. Instead, mixture of experts (MoE) models (Jacobs et al., 1991; Yüksel et al., 2012; Gross et al., 2017; Shazeer et al., 2017; Roller et al., 2021; Fedus et al., 2021; Lewis et al., 2021; Bulat et al., 2021) are 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 $\mathbf{x}$ denote the input, $f_g(\cdot)$ denote a gating function, and $f_e^k(\cdot)$ denote the k^{th} expert module. Then the MoE unit is formulated as follows:

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

where K denotes the number of expert modules, $f_g(\mathbf{x})_k$ denotes the k^{th} element of $f_g(\mathbf{x})$. 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 in computing $f_g(\cdot)$.



(a) B1 (b) B2

Figure 1: A visual illustration of re-quantization process. $\mathbf{H}^{(\ell-1)}$ denotes the feature matrix of the ℓ^{th} layer. 'Aggregation' denotes the operation of aggregating message from neighbors for each center node. 'BLP' denotes an operation of binary linear projection. (a) 'B1' denotes the process defined in (2). (b) 'B2' denotes the process defined in (3).

3 FFBGN

In this section, we present the details of our FFBGN. First, we present the challenges posed by re-quantization. Second, we present the techniques to solve the challenges in FFBGN.

3.1 Challenges of Re-Quantization

To formulate the binarization of GNNs, we take the GNN model defined in (1) as an example. The first step is to binarize the matrices involved in forward propagation in each graph convolution layer. The second step is to perform binary matrix multiplication.

Re-Quantization Different from existing BNNs which only need to binarize the multiplication of two matrices, operations $\mathbf{A}\mathbf{H}^{(\ell-1)}\mathbf{W}_g^{(\ell)}$ in (1) involve binarization of the multiplication of three binary matrices, which means $\mathbf{H}^{(\ell-1)}\mathbf{W}_g^{(\ell)}$ or $\mathbf{A}\mathbf{H}^{(\ell-1)}$ need to be further binarized (*re-quantized*) before multiplying with the third matrix. Obviously, there are two different computation orders for re-quantization. We adopt a similar framework as that in XNOR-Net++ (Bulat & Tzimiropoulos, 2019) to formulate the process.

$$\mathbf{H}^{(\ell)} = f\left(\underline{\mathcal{S}\big(\mathcal{Q}(\mathbf{A})\mathcal{S}(\mathbf{H}^{(\ell-1)})\big)\mathcal{S}(\mathbf{W}_g^{(\ell)})} \odot \boldsymbol{\alpha}_g + \underline{\mathcal{S}(\mathbf{H}^{(\ell-1)})\mathcal{S}(\mathbf{W}_s^{(\ell)})} \odot \boldsymbol{\alpha}_s\right) \tag{2}$$

$$\mathbf{H}^{(\ell)} = f\left(\underline{\mathcal{Q}(\mathbf{A})\mathcal{S}\big(\mathcal{S}(\mathbf{H}^{(\ell-1)})\mathcal{S}(\mathbf{W}_g^{(\ell)})\big)} \odot \boldsymbol{\alpha}_g + \underline{\mathcal{S}(\mathbf{H}^{(\ell-1)})\mathcal{S}(\mathbf{W}_s^{(\ell)})} \odot \boldsymbol{\alpha}_s\right), \tag{3}$$

where $\boldsymbol{\alpha}_g$, $\boldsymbol{\alpha}_s \in \mathbb{R}^{1\times r}$ are learnable scaling factors. Equation (2) indicates that $\mathbf{A}\mathbf{H}^{(\ell-1)}$ is calculated first, while Equation (3) indicates that $\mathbf{H}^{(\ell-1)}\mathbf{W}_g^{(\ell)}$ is calculated first. As we have explained in Section 1, quantizing the values of $\mathbf{A}$ to $\{+1, 0, -1\}$ is unreasonable and leads to poor performance. Instead, we quantize the values of $\mathbf{A}$ to 4 bits of precision. Specifically, we replace $\mathcal{S}(\mathbf{A})$

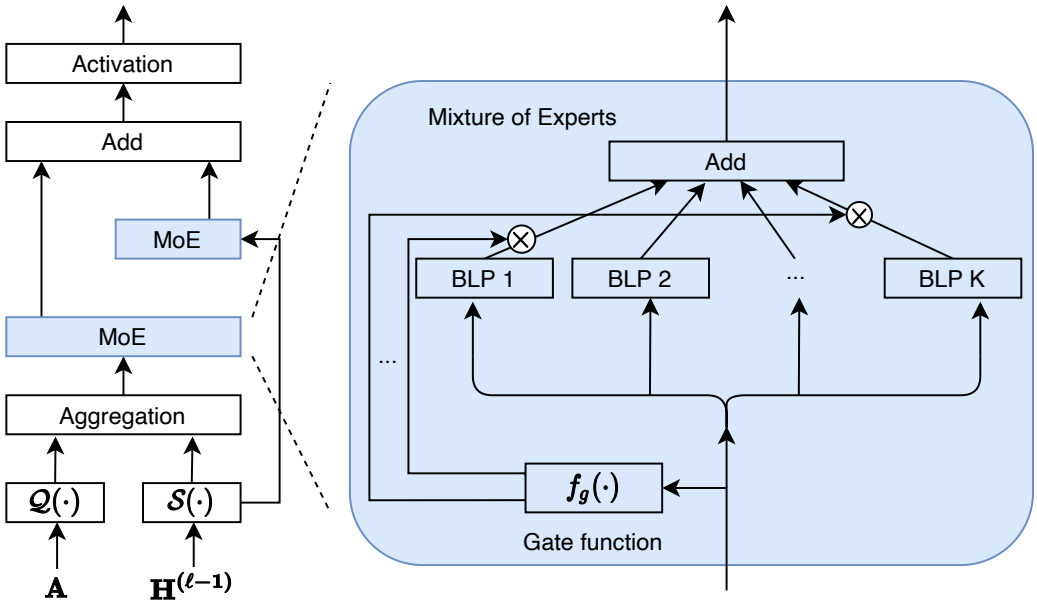


Figure 2: A visual illustration of FFBGN. 'MoE' denotes the operation defined in (7). $f_g(\cdot)$ summarizes Equation (4) and Equation (5).

with $\mathcal{Q}(\mathbf{A})$, which is a uniform quantizer defined in (Esser et al., 2020). Formally, $\mathcal{Q}(\mathbf{A})$ is defined as follows:

$$\mathcal{Q}(\mathbf{A}) = \lfloor \text{Clip}(\mathbf{A}/s, 0, 2^4 - 1) \rceil \cdot s,$$

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

Challenges It is easy to verify that Equation (2) and Equation (3) give different results and may lead to different performance. Hence, one challenge is to answer the question whether there exists 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 the question how to effectively increase the model capacity of binary GNN models after re-quantization.

3.2 Techniques for Solving the Challenges

We explore solutions for the challenges introduced in re-quantization. First, we present the conclusions about the impact of different computation orders. Then, we introduce 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 FFBGN is presented in Figure 2.

Impact of Computation Orders We perform extensive experiments to study the impact of computation orders. According to our experimental results which will be presented 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, computation order defined in (2) is superior to computation order defined in (3) in most cases. Hence, FFBGN adopts computation order defined in (2).

Mixture of Experts Similar to (Shazeer et al., 2017), we extend $\mathbf{W}_g^{(\ell)}$ or $\mathbf{W}_s^{(\ell)}$ to a set of expert modules. Let $\mathbf{Z}^{(\ell)} = \mathcal{Q}(\mathbf{A})\mathcal{S}(\mathbf{H}^{(\ell-1)})$. Expert modules for $\mathbf{W}_g^{(\ell)}$ are formulated as follows. For node i, we have:

$$(\mathbf{p}_i^{(\ell)})^\top = \mathrm{Softmax}(\mathcal{S}(\mathbf{Z}_{i*}^{(\ell)})\mathcal{S}(\mathbf{\Theta}_g^{(\ell)})), \tag{4}$$

$$q_{i,j}^{(\ell)} = \begin{cases} 1, & \text{if } j = \mathrm{argmax}(\mathbf{p}_i^{(\ell)}) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbf{Z}_{o,i*}^{(\ell)} = \sum_{k=1}^{K} q_{i,k}^{(\ell)} \cdot \mathcal{S}(\mathbf{Z}_{i*}^{(\ell)})\mathcal{S}(\mathbf{W}_{g,k}^{(\ell)}), \tag{6}$$

$$\mathbf{Z}_o^{(\ell)} = \mathrm{MoE}(\mathbf{Z}^{(\ell)}; \mathcal{W}_g^{(\ell)}, \mathbf{\Theta}_g^{(\ell)}), \tag{7}$$

where $\mathcal{W}_g^{(\ell)} = \{\mathbf{W}_{g,k}^{(\ell)} \in \mathbb{R}^{r\times r}\}_{k=1}^K$ and $\mathbf{\Theta}_g^{(\ell)} \in \mathbb{R}^{r\times K}$ are learnable parameters, K is the number of experts. $\mathrm{MoE}(\cdot)$ is a function summarizing the forward process of mixture of experts. The gating function defined in (4) and (5) is also binary, where matrix multiplication is implemented by binary operations. Since $K \ll r$, computation cost introduced by gating function is negligible. Similar to $\mathcal{S}(\cdot)$, we use STE to estimate the gradient of $\mathbf{p}_i^{(\ell)}$. Then an FFBGN layer is defined as follows:

$$\mathbf{H}^{(\ell)} = f\bigg(\mathrm{MoE}\big(\mathcal{Q}(\mathbf{A})\mathcal{S}(\mathbf{H}^{(\ell-1)}); \mathcal{W}_g^{(\ell)}, \mathbf{\Theta}_g^{(\ell)}\big) + \mathrm{MoE}\big(\mathcal{S}(\mathbf{H}^{(\ell-1)}); \mathcal{W}_s^{(\ell)}, \mathbf{\Theta}_s^{(\ell)}\big)\bigg). \tag{8}$$

3.3 Objective Function

Let $\mathcal{W} = \{\mathbf{W}_{g,1}^{(\ell)}, \cdots, \mathbf{W}_{g,K}^{(\ell)}\}_{\ell=1}^L \cup \{\mathbf{W}_{s,1}^{(\ell)}, \cdots, \mathbf{W}_{s,K}^{(\ell)}\}_{\ell=1}^L \cup \{\mathbf{\Theta}_g^{(\ell)}, \mathbf{\Theta}_s^{(\ell)}\}_{\ell=1}^L$ denote the learnable parameters in Equation (4)-Equation (7). $\hat{\mathbf{Y}} = \mathbf{H}^{(L)}$ denotes the output of FFBGN. For multi-class classification, $f(\cdot)$ is the softmax function, while it is a sigmoid function for multi-label classification. The objective function for FFBGN is formulated as follows:

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

where λ is a hyper-parameter for the Frobenius norm regularization of $\mathcal{W}$. $\mathcal{V}_{tr}$ denotes the training set.

3.4 Complexity Analysis of Operations

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 less than $\|\mathbf{A}\|_0$ and $\|\mathbf{A}\|_0 \cdot r$ is much larger than Nr, we can observe that FFBGN has much fewer FLOPs than other methods [2]. In sum, FFBGN converts most of the inefficient FLOPs into BOPs. Consequently, FFBGN is more efficient than other methods.

Table 1: Complexity analysis of operations. Numbers in column 'BOPs' represent the numbers of binary operations. Numbers in column 'FLOPs' indicate the numbers of floating operations. The analysis is mainly based on the GNN model defined in (1), and we only show the complexity of layer ℓ.

Method	BOPs	FLOPs
Bi-GNN (Bahri et al., 2021)	$\mathcal{O}\big(2Nr^2\big)$	$\mathcal{O}(2\|\mathbf{A}\|_0 \cdot r + 3Nr)$
BGN (Wang et al., 2021a)	$\mathcal{O}\big(2Nr^2\big)$	$\mathcal{O}(\|\mathbf{A}\|_0 \cdot r + 3Nr)$
FFBGN (ours)	$\mathcal{O}\big(2Nr^2 + 4\|\mathbf{A}\|_0 \cdot r + NrK\big)$	$\mathcal{O}(3Nr + NK)$

[2] No FLOPs exist in the matrix multiplication of the graph convolution operation.

4 EXPERIMENTS

In this section, we perform experiments to verify the effectiveness of FFBGN on three node classification datasets, namely ogbn-products, ogbn-papers100M, and ogbn-proteins[3] (Hu et al., 2020). The statistics of datasets are summarized in Table 2. FFBGN is implemented with Pytorch-Geometric Library (Fey & Lenssen, 2019). All experiments are run on an NVIDIA TitanXP GPU server with 48 GB graphics memory.

Table 2: Statistics of datasets.

Datasets	ogbn-products	ogbn-papers100M	ogbn-proteins
#Nodes	2,449,029	111,059,956	132,534
#Edges	61,859,140	1,615,685,872	39,561,252
Features/Node	100	128	8
#Classes	47	172	112
#Training Nodes	196,615	1,207,179	86,619
#Validation Nodes	39,323	125,265	21,236
#Test Nodes	2,213,091	214,338	24,679
Task Type	Multi-class	Multi-class	Multi-label
Metric	Accuracy	Accuracy	ROC-AUC

4.1 BASELINES AND SETTINGS

To verify the universality of FFBGN, we binarize three base GNN models, namely SAGE (Hamilton et al., 2017), PNA (Corso et al., 2020), and UniMP (Shi et al., 2021). Due to the large size of the datasets, we adopt BNS (Yao & Li, 2021) to speed up training. Furthermore, we also apply GraphNorm (Cai et al., 2021) to normalize the hidden representation for accelerating the training process.

Although none of the existing works have investigated the problem of re-quantization and consequently include inefficient full-precision operations, we can still demonstrate the effectiveness of FFBGN by comparing FFBGN with some of them. We mainly compare FFBGN with two state-of-the-art baselines, namely BGN (Wang et al., 2021a) and Bi-GNN (Bahri et al., 2021). For a fair comparison, implementations of all methods, including FFBGN, only differ in the binarization process. To analyze the effectiveness of different binarization strategies, additional training strategies in Bi-GNN (Bahri et al., 2021), like knowledge distillation and multi-stage training, are not adopted in our experiments.

The maximum epoch T, probability of dropout p, mini-batch size b, and other hyper-parameters r, L and λ are independent of binarization methods, and hence they are set to be the same for different binarization methods on one specific dataset. Concretely, for ogbn-products, $r = 128$, $L = 5$, $p = 0.1$, $T = 200$, $\lambda = 5 \times 10^{-6}$ and $b = 16 \times 1024$. For ogbn-papers100M, $r = 128$, $L = 3$, $p = 0.1$, $T = 200$, $\lambda = 5 \times 10^{-7}$ and $b = 16 \times 1024$. For ogbn-proteins, $r = 128$, $L = 5$, $p = 0.0$, $T = 1000$, $\lambda = 0.0$ and $b = 16 \times 1024$. For full-precision UniMP model, number of heads $n_h = 4$, $p = 0.3$ for ogbn-products, $p = 0.1$ for ogbn-papers100M and $p = 0.2$ for ogbn-proteins. The number of experts K is selected from $\{2, 4, 8, 16\}$ via validation sets. We use Adam (Kingma & Ba, 2015) for optimization and learning rate η is set to 0.01. For each setting, the mean results of 10 runs with different initialization each time are reported.

4.2 RESULTS

Effect of Computation Orders Let 'B1' denote the computation order defined in (2), and 'B2' denote the computation order defined in (3). To compare 'B1' and 'B2', we conduct experiments with the base binary GNN models. The results are summarized in Table 3. We can draw the following conclusions from Table 3. First, computation orders have a significant impact on the performance of

[3] https://ogb.stanford.edu/docs/nodeprop/

Table 3: Impact of computation orders. 'FP' denotes the full-precision models. 'B1' denotes the binary models performing forward propagation with the computation order defined in (2). 'B2' denotes the binary models performing forward propagation with the computation order defined in (3).

Methods	Accuracy (%) ↑ or ROC-AUC (%) ↑		
	ogbn-products	ogbn-papers100M	ogbn-proteins
SAGE (FP)	80.91 ± 0.18	65.25 ± 0.05	80.12 ± 0.26
SAGE-B2	72.16 ± 0.16	57.93 ± 0.20	76.48 ± 0.31
SAGE-B1	77.47 ± 0.21	60.36 ± 0.19	77.04 ± 0.32
PNA (FP)	80.36 ± 0.32	65.14 ± 0.03	80.29 ± 0.23
PNA-B2	74.04 ± 0.15	58.79 ± 0.12	76.48 ± 0.31
PNA-B1	76.66 ± 0.18	60.57 ± 0.08	77.04 ± 0.38
UniMP (FP)	82.43 ± 0.18	65.56 ± 0.11	80.80 ± 0.19
UniMP-B2	74.85 ± 0.38	60.23 ± 0.20	78.53 ± 0.30
UniMP-B1	79.34 ± 0.13	61.04 ± 0.24	78.54 ± 0.38

Table 4: Comparison with baselines.

Methods	Accuracy (%) ↑ or ROC-AUC (%) ↑		
	ogbn-products	ogbn-papers100M	ogbn-proteins
SAGE (FP)	80.91 ± 0.18	65.25 ± 0.05	80.12 ± 0.26
BGN	76.21 ± 0.70	61.07 ± 0.25	75.88 ± 0.39
Bi-GNN	78.59 ± 0.47	61.23 ± 0.21	77.21 ± 0.41
FFBGN (ours)	78.86 ± 0.22	61.86 ± 0.09	77.05 ± 0.30
PNA (FP)	80.36 ± 0.32	65.14 ± 0.03	80.29 ± 0.23
BGN	73.05 ± 0.31	59.79 ± 0.22	76.07 ± 0.24
Bi-GNN	77.26 ± 0.35	62.66 ± 0.09	77.84 ± 0.24
FFBGN (ours)	78.43 ± 0.23	62.12 ± 0.17	77.11 ± 0.40
UniMP (FP)	82.43 ± 0.18	65.56 ± 0.11	80.80 ± 0.19
BGN	75.08 ± 0.44	62.84 ± 0.10	72.91 ± 0.45
Bi-GNN	79.39 ± 0.20	62.50 ± 0.21	78.05 ± 0.19
FFBGN (ours)	79.90 ± 0.14	62.78 ± 0.05	78.97 ± 0.20

binary GNN models. Second, computation order 'B1' is superior to computation order 'B2' in most cases. Hence, it is better to binarize GNN models with computation order 'B1'.

Comparison with Baselines We compare our FFBGN with BGN (Wang et al., 2021a) and Bi-GNN (Bahri et al., 2021). Results are summarized in Table 4. We can draw the following conclusions. First, BGN performs worse than other methods in most cases, and the gaps are relatively large. This means that although BGN can avoid the re-quantization problem by quantizing the values of $\mathbf{A}$ to $\{+1, 0, -1\}$, it leads to worse performance. Moreover, compared to FFBGN, there still exist inefficient full-precision operations in BGN. Second, FFBGN can achieve comparable performance to Bi-GNN in all cases. This means that FFBGN can effectively increase the capacity of binary GNN models and somewhat alleviates the problem of reduced model capacity caused by re-quantization. Considering there exist lots of inefficient full-precision operations in BGN and Bi-GNN, the above results demonstrate the effectiveness of our proposed techniques in FFBGN.

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 see that the performance of binary GNN models improves in general as K increases in the range $[2, 16]$. The results show that mixture of experts can effectively increase the capacity of binary GNN models.

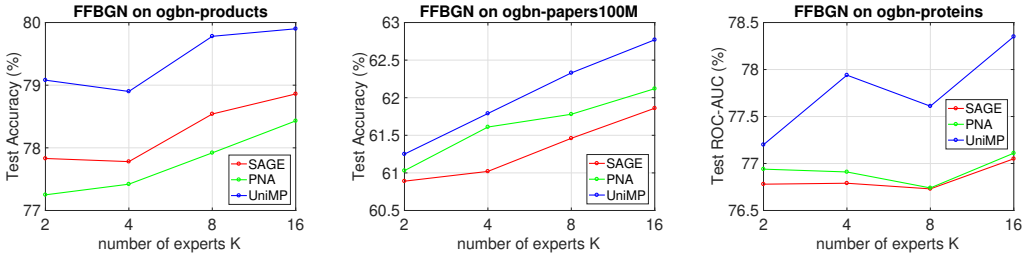


Figure 3: Effect of the number of experts K. SAGE, PNA and UniMP are three base models for binarization.

5 Conclusion

In this paper, we propose a novel method, called FFBGN, to avoid full-precision operations for binarizing 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 for avoiding full-precision operations. Experiments on real datasets demonstrate the effectiveness of FFBGN.

References

Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. Binary graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Adrian Bulat and Georgios Tzimiropoulos. XNOR-Net++: improved binary neural networks. In *British Machine Vision Conference*, pp. 62, 2019.

Adrian Bulat, Brais Mart´inez, and Georgios Tzimiropoulos. High-capacity expert binary networks. In *International Conference on Learning Representations*, 2021.

Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. GraphNorm: a principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, 2021.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 2015.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Adavances in Neural Information Processing Systems*, 2016.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: bridging full-precision and low-bit neural networks. In *IEEE/CVF International Conference on Computer Vision*, 2019.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, 2005.

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Adavances in Neural Information Processing Systems*, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: simplifying training of large, sparse models. In *International Conference on Machine Learning*, 2021.

Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.

Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2020.

Federico Monti, Michael M. Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 2016.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. *CoRR*, abs/2106.04426, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: the sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: unified message passing model for semi-supervised classification. In *International Joint Conference on Artificial Intelligence*, 2021.

Wen Torng and Russ B. Altman. Graph convolutional neural networks for predicting drug-target interactions. *Journal of Chemical Information and Modeling*, 59(10):4131–4149, 2019.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, Xiangjian He, Yiguang Lin, and Xuemin Lin. Binarized graph neural network. *World Wide Web*, 24(3):825–848, 2021a.

Junfu Wang, Yunhong Wang, Zhen Yang, Liang Yang, and Yuanfang Guo. Bi-GCN: binary graph convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021b.

Kai-Lang Yao and Wu-Jun Li. Blocking-based neighbor sampling for large-scale graph neural networks. In *International Joint Conference on Artificial Intelligence*, 2021.

Seniha Esen Yüksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Transactions Neural Networks Learning Systems*, 23(8):1177–1193, 2012.