





Fully Quantized Graph Convolutional Networks for Embedded Applications

Habib Taha Kose ¹, Jose Nunez-Yanez ²,
Robert Piechocki ¹ James Pope ¹

¹University of Bristol, Bristol, UK

²University of Linköping, SE

Abstract—Graph Neural Networks (GNNs) demonstrate remarkable proficiency across diverse tasks involving graph data, such as communication networks, physical systems, and chemical bonds. However, GNNs require significant computational resources compared to other machine learning approaches. Significant memory is necessary to store graph information. The problem is that GNNs are not well suited for resource constrained devices that have limited memory, computation, and energy. Most current solutions quantize Convolutional Neural Networks (CNNs), such the Dorefa-Net algorithm. There has not been a comprehensive quantization technique proposed for GNNs that fully quantizes all network parameters. In this paper we propose and evaluate using the Dorefa-Net algorithm integrated with the Graph Convolutional Network (GCN). We compare the accuracy performance of several quantization techniques and three commonly used datasets. Our findings reveal that the Dorefa-Net method quantizes network parameters down to 4-bit integers with an acceptable accuracy loss (up to 2%) compared to the base model. However, we find that Dorefa-Net does not perform well for aggressive quantization levels (e.g. 1 and 2 bit), which are necessary for specialized hardware, such as FPGAs. To address this, we propose a modified version denoted *Dorefa-Graph*. We show that *Dorefa-Graph* performs better than the other quantization techniques, particularly when aggressively quantized, making it better suited for bespoke hardware.

Index Terms—Graph Neural Networks, Network Quantization, Dorefa-Net

I. INTRODUCTION

GNNs have emerged as an effective machine learning approach integrating graph data structures with Neural Network (NN) architectures. For complex graph relationships, GNNs have been shown to perform well for various inference tasks. [1]. GNNs have been proposed for numerous application domains that include social networks, molecular bonding, e-commerce, product recommendations, particle physics, natural language processing, traffic applications, anomaly detection, and numerous academic studies [2]–[4]. However, the complex nature of graph data poses computational challenges. Quantization techniques emerge as a strategy to enhance computational efficiency and reduce memory consumption by representing activations and model weights with fewer bits. While low-number of bits can accelerate matrix multiplications [5], a delicate balance is required between model efficiency and accuracy. Despite the well-established exploration of quantization in NNs and CNNs [6], GNNs remain insufficiently investigated in this regard.

This study focuses on quantized Graph Convolutional Networks (GCNs) using the well known Dorefa-Net algorithm [7], commonly used in CNN applications. Our approach underscores the advantages of matrix quantization in GCN schemes, enabling swift and efficient computations in matrix multiplication operations. Quantizing the GCN model allows the efficient utilization of limited computational resources, particularly in embedded devices. This research proposes quantized GCN models that can be implemented in efficient, high-speed, and energy-efficient hardware. In this paper, we focus on the accuracy loss of Post-Training Quantization (PTQ) with the Dorefa-Net algorithm. Additionally, we examine the impact of quantization on accuracy using common datasets in core GCN layers. Our contributions include:

- We present inference accuracy results of the quantized GCN model with the Dorefa-Net algorithm. To the best of our knowledge, this is the first work showing the impact of the Dorefa-Net algorithm for fully quantized GCNs.
- We explore a modified version of the Dorefa-Net algorithm for GCNs, *Dorefa-Graph*, that performs better, particularly at lower bit numbers.
- We compare our results with another common method SGQuant [8]. We use a public available version of the method and denote this version as *SGQuant'* [9].

The source code for our technique and experiments is publicly available here https://github.com/TahaKose/Quantized_GCIN. The paper begins by providing context and discussing prior research in Section 2. Section 3 presents the specifics of the Dorefa-Net and Dorefa-Graph algorithms. Performance evaluation and comparisons are conducted in Section 4. Finally, Section 5 concludes with a summary and outlines directions for future research.

II. BACKGROUND AND RELATED WORK

In this section, we provide an overview of Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and quantization approaches. We also discuss relevant literature.

A. Graph Neural Networks

GNNs prove to be effective instruments for learning graph data by taking into account the graphical structure along with the properties of nodes and edges. A graph is denoted in terms of edges and vertices as $G = (V, E)$, Here, V signifies

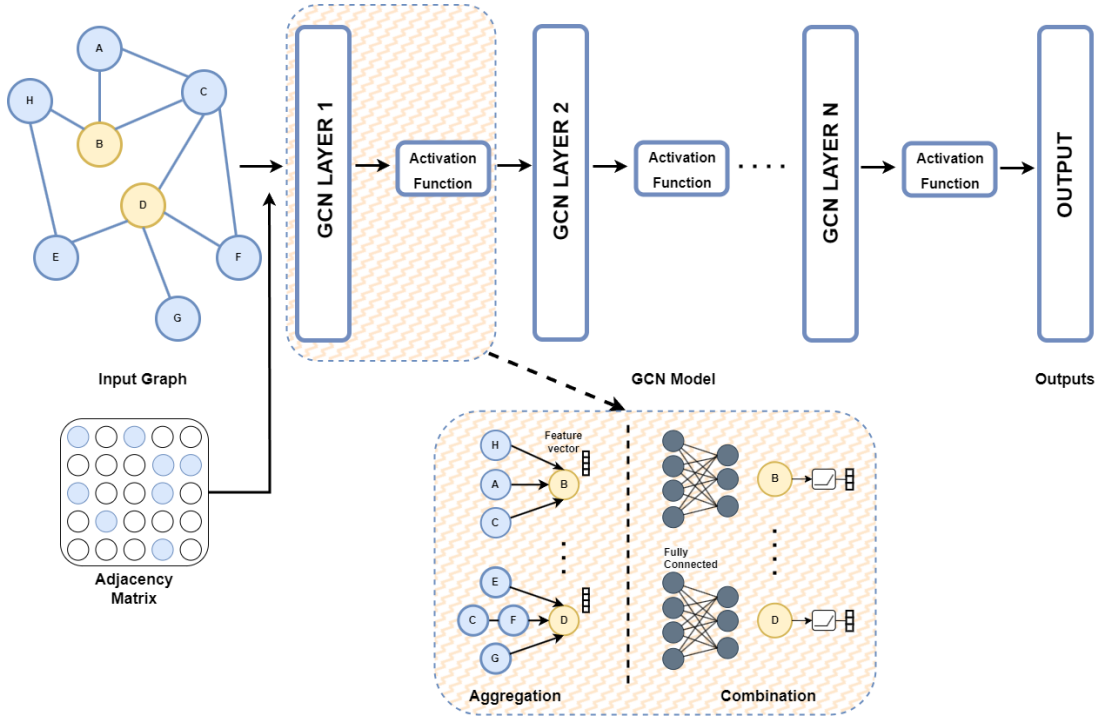


Fig. 1: N-Layer Graph Convolutional Neural Network Overview

vertices (e.g., users, molecules, citations), and E represents edges, capturing the intricate relationships between vertices. As depicted in Figure 1, GNN models typically encompass two phases: the Aggregation and Combination (Update) Phases. The aggregation phase computes a new feature vector by summing the features of neighboring nodes, as illustrated in Equation 1. Methods such as weighted average, maximum, or summation are employed in this phase to generate a new feature vector. These feature vectors are then combined in the combination phase, as depicted in Equation 2, forming a high-level feature matrix. In this phase, the new feature vector of each node merge with the original feature matrix. The resultant feature matrix encapsulates high-level features and finds applications in classification or regression tasks.

$$\mathbf{a}_v^l = \text{Aggregation} \left(\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v) \right) \quad (1)$$

$$\mathbf{h}_v^l = \text{Combination} \left(\mathbf{a}_v^l \right) \quad (2)$$

In Equations 1 and 2, \mathbf{h}_v^l represents the feature vector of a node, and $\mathcal{N}(v)$ denotes the set of nodes that are neighbors of a given node. The arrangement of the aggregation and combination phases has been a focal point in studies investigating system efficiency. The alteration in the size of node feature vectors after the combination stage has a direct impact on computational efficiency.

B. Graph Convolutional Networks

GCNs are GNN models that apply the convolution process to graph data. GCNs can be expressed locally for a single edge

or vertex, or globally for all edges and vertices. Equation 3 shows the local expression of GCNs.

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\mathbf{W}^{(l)} \times \left(\sum_{j \in \hat{\mathcal{N}}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{h}_j^{(l)} \right) \right) \quad (3)$$

In Equation 3, the feature vector of node i is denoted by $\mathbf{h}_i^{(l)}$ and \mathbf{W}^l is the trainable weight value. d_i is the degree of node i and $\hat{\mathcal{N}}(i)$ are its neighbouring nodes. The global representation of GCNs is given in equation 4.

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \quad (4)$$

Where $\mathbf{H}^{(l+1)}$ is the input feature matrix for layer l , \mathbf{A} is the adjacency matrix and σ is the non-linear activation function. $\mathbf{A}\mathbf{H}$ refers to the aggregation phase and $\mathbf{H}\mathbf{W}$ to the combination phase. However, in this equation, only the features of the neighbour nodes are aggregated and the node itself is not included. In addition, multiplication by the adjacency matrix changes the scale of the feature vector. In this case, higher-degree nodes will have more impact on the aggregation because they have more neighbours, and the effect of low-degree nodes will become insignificant. To overcome these problems, equation 5 is proposed.

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(l)} \mathbf{W}^{(l)}) \quad (5)$$

To obtain the normalized matrix $\tilde{\mathbf{A}}$, the identity matrix \mathbf{I} is added to the adjacency matrix \mathbf{A} . $\tilde{\mathbf{D}}$ is the diagonal node degree matrix.

C. Quantization

Quantization stands out as a widely adopted technique for creating scalable neural networks, aiming to reduce memory demands and accelerate computations. Two primary approaches, scalar quantization, and vector quantization, are employed. Scalar quantization reduces the number of bits used to store network parameters, while vector quantization represents these parameters through codebooks. Both approaches yield positive outcomes in terms of shrinking network size. Although scalar quantization exhibits a limited compression ratio, it proves to be a cost-effective alternative to vector quantization, avoiding complex and expensive additional computations. This makes it a powerful choice for implementing scalable GNNs on energy-constrained devices. Quantizing the feature matrix, a substantial component of the GNN model significantly reduces the network’s size. Various quantization methods, including binary [10], vector [11], scalar [8], and hybrid approaches [12], effectively minimize the size of the feature matrix. Degree-based quantization emerges as an efficient strategy for compressing features, requiring specific preprocessing steps. Degree Quant [13] employs a Quantization-Aware Training (QAT) model, utilizing a stochastic quantization mask to emphasize the importance of nodes, ensuring nodes with crucial information (high degree) retain full precision, while others are quantized. This will be difficult to map to hardware, as different nodes will have different levels of precision, and the hardware will have little ability to adapt at the fine granularity level. QAT updates network weights during training, considering the quantization process. Compared to Post-Training Quantization (PTQ), which quantizes a trained model after training, QAT exhibits superior accuracy performance but demands additional computational power, making it less suitable. While quantizing only the feature matrix can significantly reduce network size, more comprehensive approaches are necessary for embedded devices. SGQuant [8] introduces a scalar quantization method for PTQ working at various levels, quantizing features and attention with automatically selected bits. Quantizing weights and activations in GNNs is a common practice. LPGNAS [14] enables quantization of weights and activations throughout training and inference, employing the Network Architecture Search (NAS) algorithm. Recent techniques aim to enhance quantization strategies with additional methods. EXACT [15] utilizes Random Projection for activation quantization. Recognizing that the multiplication of adjacency, weight, and feature matrices underlies the aggregation and combination stages in GNNs, quantizing these matrices is imperative for enhanced applicability and structural flexibility. For example, GCINT [16] suggests a dynamic structure allowing the quantization of weights, activations, error, and loss functions during both forward and backward passes. Wang et al. [17] propose a scheme quantizing weights, activations, messages, inputs, and outputs during aggregation and combination phases, omitting the quantization of the adjacency matrix. QGTC [18] treats the adjacency matrix as binary, allowing node embeddings and

weights to be presented in any bit number.

III. METHODOLOGY

In this section, we explain the Dorefa-Net and Dorefa-Graph techniques for quantizing GCN parameters. We provide a thorough explanation of the formulas that describe the quantization algorithm and the methods of implementation.

A. Dorefa-Net Quantization

The Dorefa-Net algorithm, a well-established quantization technique for Convolutional Neural Networks (CNN), has garnered widespread recognition. While extensively applied in CNN, its implications for GCN necessitate exploration. Our study employs the Dorefa-Net quantization method to create a lightweight GCN model. Leveraging Dorefa-Net enables the creation of a network model that proves both time and memory-efficient during the inference phase. This efficiency is achieved by utilizing a reduced number of bits, accompanied by an acceptable trade-off in accuracy. This paper specifically delves into the investigation of the impact of quantization during the inference stage, with no implementation of the quantization method during the training phase. Our experiments showed that the Dorefa-Net scheme is not useful for QAT in GCN models due to the activation distribution. Figure 2 visually outlines how the effects of PTQ on the GCN are replicated in our study.

In the DoReFa-Net algorithm, the k -bit ($k > 1$) weight and the activation quantizations are shown in Equation 6 and Equation 7, respectively.

$$f_{\omega}^k(r_i) = 2 \text{ quant}_k \left(\frac{\tanh(r_i)}{2 \max(|\tanh(r_i)|)} + \frac{1}{2} \right) - 1 \quad (6)$$

$$f_{\alpha}^k(r) = \text{quant}_k(r) \quad (7)$$

In equation 6, \tanh is used to express the weights in the range $[-1, 1]$. After normalization, the integer output r_o is obtained. In equation 7, the input activation r is quantized as k -bits, so that r can take 2^k different values. The resulting

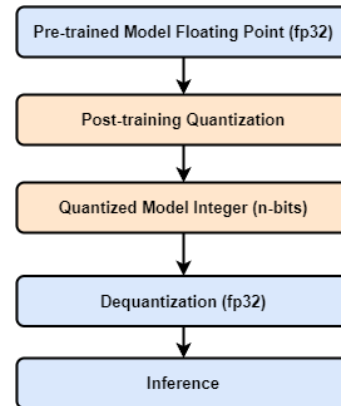


Fig. 2: Quantization Effect Simulation Scheme

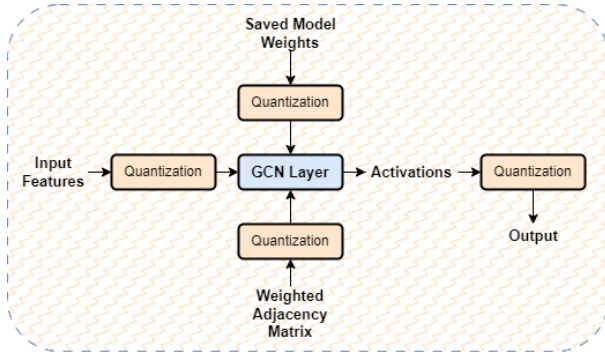


Fig. 3: Inputs/Outputs of Quantized Layer

values are used to update the weights. This technique implements Equation 8 to achieve quantization of both weights and activation

$$\text{quantize}(\text{num_bits}) = \frac{\text{round}(x \cdot (2^{\text{num_bits}} - 1))}{2^{\text{num_bits}} - 1} \quad (8)$$

The intensive of GCNs poses challenges in deploying them on edge devices with constrained capacities [19]. Hence, an important solution lies in the adoption of a fully quantized network. The Dorefa-Net algorithm stands out as a uniform quantization method for network parameters. Algorithm 1 provides a comprehensive illustration of implementing the Dorefa-Net scheme on GCN. In this scheme, the Dorefa-Net technique is applied to quantize the final model weights, which are initially trained with full precision and subsequently fed back into the model, as delineated by Equation 6. Correspondingly, the feature vector and layer activation outputs undergo quantization using the activation quantization method specified in Equation 7. The comprehensive impact of a fully quantized network is assessed by scrutinizing structural alterations in the adjacency matrix, which is quantized according to Equation 7. The adjacency matrix, portraying interconnections among nodes in GCNs, solely consists of binary values, 0 or 1. However, during the aggregation phase, the matrix A in AH multiplication aligns with matrix AD multiplication. To account for this, a weighted adjacency matrix is introduced, derived from the outcome of AD multiplication upon quantizing the adjacency matrix. Consequently, this matrix incorporates floating-point weight expressions obtained from the diagonal degree matrix D instead of only 0 and 1 values. Figure 3 visually explains the utilization of quantized values in a GCN layer during the inference phase. Furthermore, a flexible model structure is adopted to facilitate mixed-precision quantization at layer level with varying bit numbers. This strategic approach not only addresses resource constraints but also provides adaptability to different precision requirements, making it conducive to the deployment of GCNs in embedded systems.

Algorithm 1 Training and Testing with Dorefa-Net Quantization. Feature matrix and edge information are used as input.

```

1: for epoch ← 1 to args.epochs do
2:   #Train the model:
3:   model.train()
4:   output ← model(features, adj)
5:   loss ← compute_loss(output, labels)
6:   loss.backward()
7:   Optimize the model parameters:
8:   optimizer.step()
9:   Save the model after training:
10:  torch.save(model.state_dict(), "model.pth")
11: end for
12: #Test the quantized model:
13: model.eval()
14: model.load_state_dict(quantize_weights("model.pth"))
15: output ← model(features, adj)
16: #Quantize activations and perform inference:
17: q_features ← quantize_activations(features)
18: q_adj ← quantize_activations(adj)
19: quantized_output ← model(q_features, q_adj)

```

B. Dorefa-Graph

Dorefa-Net was designed for use with CNNs, but in some cases, it may not be suitable for GCNs. Our investigations have shown that the distribution of activation outputs of GCN layer 2 changes gradually with the number of epochs. This gradual shift hinders the Dorefa-Net system, which quantizes the activation outputs between 0 and 1, leading to accuracy losses when bit values are low and optimal quantization levels are difficult to determine. We suggest using Dorefa-Graph to address these issues. This method differs from the initial quantization algorithm as it uses a dynamic approach to activation quantization. The modified technique computes the quantization parameters for every activation input tensor and establishes the best levels. The quantization parameters of Dorefa-Graph are recalculated for each input's activation using Equations 9, 10, 11.

$$\text{Scale} = \frac{f_{\max} - f_{\min}}{q_{\max} - q_{\min}} \quad (9)$$

$$q_{\min} = -2^{(\text{num_bits}-1)} \quad (10)$$

$$q_{\max} = q_{\min} + (2^{\text{num_bits}} - 1) \quad (11)$$

In the Equation 9, f_{\min} and f_{\max} are the smallest and largest values of the input respectively, whilst q_{\min} and q_{\max} determine the boundaries of the quantization range. Scale refers to the equal intervals between quantization levels. Input tensors may include both positive and negative values. It is crucial to locate the point where the numbers in the coordinate system change from negative to positive, known as the zero point, for proper comprehension and handling of the distribution. The

Equation 12 indicates the position of the zero point in the quantized range.

$$Zero\textit{point} = q_{min} - \frac{f_{min}}{Scale} \quad (12)$$

This enables Dorefa-Graph to adapt more effectively to different activation ranges and data distribution. As a result, it can achieve better accuracy, especially in aggressive quantization operations, which are essential for embedded systems. In our research, we investigate the effect of quantization for accuracy. We convert the quantized values ($qval$) back to floating-point values ($dqval$) for use with conventional hardware (as shown in Equation 14) and then apply the remaining parameter quantization using the original Dorefa-Net scheme described in Section III.

$$qval = \frac{input}{Scale} + Zero\textit{point} \quad (13)$$

$$dqval = Scale(qval + Zero\textit{point}) \quad (14)$$

IV. PERFORMANCE EVALUATION

In this section, we present examinations of various GCN models with commonly used data sets. Our aim is to demonstrate the quantization effects of the Dorefa-Net and Dorefa-Graph quantization techniques.

A. Datasets

This study derives its results from the exploration of commonly used datasets, with a focus on assessing the impacts of the Dorefa-Net and Dorefa-Graph schemes on GCNs. The investigation employs datasets from the citation dataset family, encompassing the Cora, PubMed, and Citeseer datasets. The key characteristics of these datasets are summarized in Table II, delineating variations in node and edge numbers. Despite these variations, there are shared features among the datasets, including sparsity in the feature and adjacency matrices, alongside density in the weight matrix.

B. Graph Convolutional Network Layers

Two distinct and commonly employed GNN layers were selected to construct the network architectures. The first of these is the GCN layer, a structure derived from the work of Kipf et al. [20]. GCN holds prominence in contemporary implementations and is extensively utilized. The second layer employed in the GNN is the Graph Isomorphism Network (GIN) layer, introduced in the study by Xu et al. [21], widely applied across various tasks in GNNs. While GCNs focus solely on adjacent nodes when updating node properties, GINs incorporate rules beyond adjacent nodes in their computations. This distinction renders GCNs effective for tasks involving local information, whereas GIN layers are apt for global tasks such as graph classification. In this study, both layer types are configured with a layer size of 2 and a hidden size of 64.

C. Quantization Performance

To evaluate the system performance of the quantization techniques, we fully quantized the model parameters using the Dorefa-Net, Dorefa-Graph and SGQuant' methods. These methods quantized the weighted adjacency matrix, input feature matrix, weight matrix and activations. Our results diverge from the original paper for SGQuant because the authors only quantized features and activations. Quantization bits were chosen as powers of 2 and we followed the quantization process until the precision reached 1 bit. The experiments were performed on the Google Co-Laboratory environment. T4 GPU, Intel Xeon CPU, and 12 GB RAM provided by Co-Laboratory were used as test hardware. In this research, training and testing applications were done using the Py-Torch framework. Note that conventional hardware was used in the experiments, but this was done to demonstrate our methodology. Future work will integrate the approach with FPGA hardware. The identical hardware setup was applied for both training and inference phases. Every model initiated the training phase with 32-bit floating point accuracy, and no quantization was executed during this phase. Quantization was applied to reach the desired bit value. Then, the quantized model was used for inference to test datasets with the trained model. The experimental findings are presented in Table I. According to the results, the Dorefa-Net and Dorefa-Graph methods consistently give higher or equivalent accuracy values compared to the SGQuant method up to the 4-bit integer quantization level. However, for some specific cases, including Citeseer 4-bit int (GCN/GIN) and Pubmed 4-bit int (GIN), Dorefa-Net has significantly less accuracy than the SGQuant method when quantized to 4 bits. At lower levels of quantization (2-bit and 1-bit), Dorefa-Net is not successful, whereas SGQuant sometimes achieves model quantization with an acceptable loss of accuracy. Dorefa-Net uses only the sign of the value for 1-bit quantization, unlike other cases. This causes the quantization results to be more accurate for 1-bit quantization than for 2-bit quantization. In addition to this, Dorefa-Graph has the best accuracy performance for all bit values except in rare cases. The modified scheme has prevented the deterioration of the aggressive quantization results of the original Dorefa-Net scheme and has placed it ahead of the methods compared. A standard number of tests were done for both approaches to get the precision values and the confidence interval for the final value was set to 80%. The assessments revealed that the Dorefa-Net algorithm is able to quantize up to 4-bit integer accuracy in most instances, while the Dorefa-Graph algorithm can achieve 1-bit integer accuracy with relatively superior loss of accuracy.

V. CONCLUSION

In this work, we investigated the impact and performance of the Dorefa-Net quantization algorithm on GCNs. To our knowledge, this is the first comprehensive study of the Dorefa-Net algorithm's impact on GCN inference accuracy. Our research indicates that both the Dorefa-Net and Dorefa-Graph algorithms successfully perform weight, adjacency, feature,

Dataset	Network	Accuracy %					
		Base Model	F16W16A16	F8W8A8	F4W4A4	F2W2A2	F1W1A1
Cora	Dorefa-Graph GCN	81.5±0.35	80.1±1.04	80.6±0.06	80.1±0.63	67.8±0.21	67.4±1.44
	Dorefa-Net GCN	81.5±0.35	79.9±0.67	80.3±0.13	79.1±0.72	26.1±3.3	31.2±5.35
	SGQuant' GCN	81.5±0.35	76.7±1.07	76.1±1.2	78.1±0.97	65.4±3.26	37.3±2.53
Cora	Dorefa-Graph GIN	74.3±0.27	74.6±0.63	75.1±0.47	74.4±0.27	64.7±0.72	52.4±4.32
	Dorefa-Net GIN	74.3±0.27	71.7±0.88	71.3±1.4	71.3±1.11	19.4±4.81	23.7±2.32
	SGQuant' GIN	74.3±0.27	71.1±0.57	70.7±0.94	71.6±0.77	64.2±1.23	33.0±2.05
Citeseer	Dorefa-Graph GCN	70.4±0.59	68.0±1.26	66.7±0.94	63.0±0.9	57.2±1.27	52.2±1.64
	Dorefa-Net GCN	70.4±0.59	65.8±1.22	68.2±0.84	41.2±9.05	18.01±0.03	23.2±1.59
	SGQuant' GCN	70.4±0.59	63.1±1.07	58.9±2.91	57.9±3.15	48.6±3.29	17.0±1.87
Citeseer	Dorefa-Graph GIN	66.4±0.24	68.1±0.68	66.5±0.22	65.7±1.04	56.6±0.57	40.4±5.71
	Dorefa-Net GIN	66.4±0.24	63.9±0.67	64.7±1.15	41.6±1.46	17.3±3.29	24.2±8.02
	SGQuant' GIN	66.4±0.24	60±2.05	63.8±0.69	60.9±2.03	52.5±2.72	29.9±1.82
Pubmed	Dorefa-Graph GCN	79.7±0.23	79.2±0.5	79.8±0.2	78.4±0.67	60.5±0.57	69.2±0.54
	Dorefa-Net GCN	79.7±0.23	78.2±0.36	78.4±0.51	75.3±1.3	32.6±6.76	47.9±6.25
	SGQuant' GCN	79.7±0.23	76.7±0.95	75.7±0.94	75.1±0.98	72.2±0.92	51.4±2.21
Pubmed	Dorefa-Graph GIN	74.4±0.60	75.1±0.26	74.9±0.18	74.6±0.34	68.0±0.73	66±2.42
	Dorefa-Net GIN	74.4±0.60	75±0.55	73.9±1.12	70.3±0.60	32.9±6.3	56.5±4.17
	SGQuant' GIN	74.4±0.60	74.8±0.55	74.8±0.37	74.5±0.28	69.1±0.88	56.1±2.14

*SGQuant' results are generated using publicly available quantization code [9].

*The base model comprises 32-bit floating point values. In other cases, the bit values shown are used as integers.

TABLE I: Accuracy Results for Different Bit Levels

	Cora	Pubmed	Citeseer
Graph Nodes	2708	19717	3327
Graph Edges	10556	88648	9104
Graph Features	1433	500	3703
Number of Classes	7	3	6
Adjacency Sparsity	99.85%	99.97%	99.91%
Feature Sparsity	98.73%	89.97%	99.14%

TABLE II: Characteristics of graph datasets

and activation quantization on GCNs with acceptable loss of accuracy. Dorefa-Net quantizes all network parameters up to 4-bit integer precision, resulting in a loss of only 2% accuracy in the best-case scenario. For low bit values (e.g. 1 or 2 bits), our proposed method, Dorefa-Graph, achieved relatively superior accuracy improvement over other models. Except for a couple of cases, Dorefa-Graph was significantly better (up to 10%) than SGquant for all quantization levels. These results provide evidence of the effectiveness of Dorefa-Graph in GCN architectures. Future research is to improve the Dorefa-Graph algorithm with additional methods to achieve more aggressive quantization while minimizing loss of accuracy. Future work includes the development of accelerators specifically designed for embedded systems with low-precision quantized matrices.

REFERENCES

- [1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

- [2] X. Ju, D. Murnane, P. Calafiura, N. Choma, S. Conlon, S. Farrell, Y. Xu, M. Spiropulu, J.-R. Vlimant, A. Aurisano *et al.*, "Performance of a geometric deep learning pipeline for h-lhc particle tracking," *The European Physical Journal C*, vol. 81, pp. 1–14, 2021.
- [3] J. Pope, J. Liang, V. Kumar, F. Raimondo, X. Sun, R. McConville, T. Pasquier, R. Piechocki, G. Oikonomou, B. Luo *et al.*, "Resource-interaction graph: Efficient graph representation for anomaly detection," *arXiv preprint arXiv:2212.08525*, 2022.
- [4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.
- [5] J. Nunez-Yanez, "Accelerating graph neural networks in pytorch with hls and deep dataflows," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, F. Palumbo, G. Keramidas, N. Voros, and P. C. Diniz, Eds. Cham: Springer Nature Switzerland, 2023, pp. 131–145.
- [6] J. Nunez-Yanez and M. Hosseinabady, "Sparse and dense matrix multiplication hardware for heterogeneous multi-precision neural networks," *Array*, vol. 12, p. 100101, 2021.
- [7] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.
- [8] B. Feng, Y. Wang, X. Li, S. Yang, X. Peng, and Y. Ding, "Sgquant: Squeezing the last bit on graph neural networks with specialized quantization," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 1044–1052.
- [9] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.
- [10] M. Bahri, G. Bahl, and S. Zafeiriou, "Binary graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9492–9501.
- [11] M. Ding, K. Kong, J. Li, C. Zhu, J. Dickerson, F. Huang, and T. Goldstein, "Vq-gnn: A universal framework to scale up graph neural networks using vector quantization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6733–6746, 2021.
- [12] L. Huang, Z. Zhang, Z. Du, S. Li, H. Zheng, Y. Xie, and N. Tan, "Epquant: A graph neural network compression approach based on product quantization," *Neurocomputing*, vol. 503, pp. 49–61, 2022.

- [13] S. A. Taylor, J. Fernandez-Marques, and N. D. Lane, "Degree-quant: Quantization-aware training for graph neural networks," *arXiv preprint arXiv:2008.05000*, 2020.
- [14] Y. Zhao, D. Wang, D. Bates, R. Mullins, M. Jamnik, and P. Lio, "Learned low precision graph neural networks," *arXiv preprint arXiv:2009.09232*, 2020.
- [15] Z. Liu, K. Zhou, F. Yang, L. Li, R. Chen, and X. Hu, "Exact: Scalable graph neural networks training via extreme activation compression," in *International Conference on Learning Representations*, 2021.
- [16] Q. Wu, L. Zhao, H. Liang, X. Wang, L. Tao, T. Tian, T. Wang, Z. He, W. Wu, and X. Jin, "Gcint: Dynamic quantization algorithm for training graph convolution neural networks using only integers," 2022.
- [17] S. Wang, B. Eravci, R. Guliyev, and H. Ferhatosmanoglu, "Low-bit quantization for deep graph neural networks with smoothness-aware message propagation," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 2626–2636.
- [18] Y. Wang, B. Feng, and Y. Ding, "Qgtc: accelerating quantized graph neural networks via gpu tensor core," in *Proceedings of the 27th ACM SIGPLAN symposium on principles and practice of parallel programming*, 2022, pp. 107–119.
- [19] J. Li, A. Louri, A. Karanth, and R. Bunescu, "Gcnax: A flexible and energy-efficient accelerator for graph convolutional neural networks," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 775–788.
- [20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
- [21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.