Low-bit Quantization for Deep Graph Neural Networks with Smoothness-aware Message Propagation

Shuang Wang
University of Warwick
Coventry, United Kingdom
Shuang.Wang.1@warwick.ac.uk

Bahaeddin Eravci
TOBB University of Economics and Technology
Ankara, Turkey
beravci@gmail.com

Rustam Guliyev
University of Warwick
Coventry, United Kingdom
Rustam.Guliyev@warwick.ac.uk

Hakan Ferhatosmanoglu*
University of Warwick
Coventry, United Kingdom
Hakan.F@warwick.ac.uk

ABSTRACT

Graph Neural Network (GNN) training and inference involve significant challenges of scalability with respect to both model sizes and number of layers, resulting in degradation of efficiency and accuracy for large and deep GNNs. We present an end-to-end solution that aims to address these challenges for efficient GNNs in resource constrained environments while avoiding the oversmoothing problem in deep GNNs. We introduce a quantization based approach for all stages of GNNs, from message passing in training to node classification, compressing the model and enabling efficient processing. The proposed GNN quantizer learns quantization ranges and reduces the model size with comparable accuracy even under low-bit quantization. To scale with the number of layers, we devise a message propagation mechanism in training that controls layer-wise changes of similarities between neighboring nodes. This objective is incorporated into a Lagrangian function with constraints and a differential multiplier method is utilized to iteratively find optimal embeddings. This mitigates oversmoothing and suppresses the quantization error to a bound. Significant improvements are demonstrated over state-of-the-art quantization methods and deep GNN approaches in both full-precision and quantized models. The proposed quantizer demonstrates superior performance in INT2 configurations across all stages of GNN, achieving a notable level of accuracy. In contrast, existing quantization approaches fail to generate satisfactory accuracy levels. Finally, the inference with INT2 and INT4 representations exhibits a speedup of 5.11 × and 4.70 × compared to full precision counterparts, respectively.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Theory of computation** → **Approximation algorithms analysis**; • **Hardware** → *Power estimation and optimization.*

*Also with Amazon Web Services. This publication presents work performed at the University of Warwick and is not associated with Amazon.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614955

KEYWORDS

graph neural networks; quantization; oversmoothing in GNNs; large-scale graph management; scalable machine learning

ACM Reference Format:
Shuang Wang, Bahaeddin Eravci, Rustam Guliyev, and Hakan Ferhatosmanoglu. 2023. 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 (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3614955

1 INTRODUCTION

Data analytics and machine learning on large graphs encompass a wide array of applications, including recommender systems, social networks, web analysis, and computational biochemistry. Some tasks within this scope include node classification, community detection, link prediction, reachability analysis, and influence optimization. Recently, Graph Neural Networks (GNNs) have shown to be effective for learning over graphs [59]. GNNs utilize an iterative process, aggregating features from neighboring nodes through learnable parameters, thus generating rich and informative embeddings.

The versatility of GNNs often comes at a price – elevated memory and computation demands. This poses challenges when scaling up to larger graphs and deeper models. Large-scale graphs naturally increase the storage costs and the neighborhood size during the aggregation phase. While deeper models, with more iterative layers, add computational strain, they do capture intricate relationships by broadening the nodes' receptive fields. To counter these, recently, quantization approaches were developed to compress both the model and graph, aiming to reduce storage, computation, and power requirements for inference workloads [11, 16, 51, 67].

Quantization is the process of mapping continuous numerical values into smaller sized representations (e.g., using 8-bits). There are a variety of quantization methods developed for data-intensive tasks, ranging from multi-dimensional indexing for range and similarity queries [6, 14, 17, 52, 56, 57, 61] to processing convolutional and recurrent neural networks [33, 55, 60]. For GNNs, quantization is useful in many practical settings, such as resource-efficient representation learning, reducing energy and communication in sensors, IoT and mobile devices, on-device and embedded learning, managing data/models in distributed and edge computing, and

recommender systems which commonly employ graph neural networks.

Unlike conventional applications of quantization, GNNs present unique challenges due to their intrinsic characteristics, which are not, effectively, addressed by the current methods. (i) The process of neighborhood aggregation in GNNs can lead to significant variance in high in-degree node embeddings, thereby exacerbating the quantization error, especially in low-bit cases [51]. (ii) As GNNs deepen, they tend to experience the "oversmoothing" issue where each embedding loses its discriminative information due to the repeated, unregulated message passing [19]. It is important to understand if this problem remains or is aggravated with the introduction of model quantization. Thus, while reducing GNN size and enabling compressed processing are pivotal for performance efficiency, addressing oversmoothing is crucial to ensure accuracy, especially in deeper models.

While recent studies [27, 51, 67] have delved into GNN quantization, the problem is far from being solved and there is no effective solution for low-bit quantization that scales for deeper GNNs. Our paper underscores this challenge, revealing that state-of-the-art GNN quantization methods undergo significant degradation at low bit counts (INT4 and INT2). This is more pronounced in deeper GNNs, due to accumulated layer-by-layer quantization errors. We aim to address these intricacies and develop an end-to-end solution.

Our solution involves a quantizer that learns the quantization ranges (QLR) along with a skewness-aware bitwise truncation (BT*) mechanism. Additionally, we introduce a smoothness-aware message propagation scheme (SMP) to counter the oversmoothing issue in quantized models. This quantizer determines an optimized, data-aware learnable range grounded in the input data distribution, thereby minimizing model redundancy. It is shown to retain its effectiveness with low-bit representations, which makes it apt for large deep GNNs. The skewness-aware truncation embedded within the quantizer improves the accuracy particularly in low-bit (INT2) scenarios. Our message propagation scheme aims to mitigate oversmoothing in deep GNNs by constraining the layer-wise shifts in similarities among neighboring nodes. Furthermore, we prove that by using SMP, the quantization error can be suppressed to a bound. Finally, we demonstrate the efficiency and accuracy of our solution through node classification accuracy on quantized GNN models.

Experimental results demonstrate improvements over the state-of-the-art approaches across various performance measures and workloads. Specifically, our quantizer (QLR) demonstrates remarkable advancements in low-bit quantization, outperforming existing quantization methods while resulting in reduced model sizes. For deeper GNNs, our SMP method delivers more accurate classification compared to other deep GNN approaches both in full-precision and quantized versions. The low-bit quantized SMP, using QLR, achieves greater improvement over alternative deep quantized GNN approaches with the help of the quantization error bound with SMP. BT* improves node classification accuracy on large datasets with INT2 representation, making it comparable to INT8 accuracy. We also show that the INT2 quantization model can yield an inference speedup of 5.11 × compared to the full-precision model.

2 RELATED WORK

Quantization has been commonly employed for neural network (NN) models [20]. NN training is bottlenecked by high memory requirements to handle large data involving intermediate results and feature maps [2]. NNs can be trained with low precision using dynamic fixed point arithmetic [9].

Quantization for neural networks can be performed during or after training. The post-training approaches quantize weights or activation of neural networks on a pre-trained full-precision model [4, 24]. Their low-bit quantization performance incurs significant accuracy degradation. The quantization-aware training aims to avoid this performance degradation [5, 12]. A useful technique is to expose errors from the quantization operation to the forward pass at model training and use straight-through estimator(STE) to compute the gradients [5]. Banner et al. [3] provide a theoretical analysis showing considerable room for quantization under Gaussian weight assumption leading to 8-bit DNNs with comparable accuracy. The success of quantization has led to binary NNs (BNN) drastically reducing computation and memory requirements using hardware-supported bitwise operations with strong precision performances [28]. The efficacy of high-order bit representations, involving bitwise truncation applied to 32-bit word embeddings, has been demonstrated in previous studies [8].

Pioneering work on learning node representations [46] has been followed by variants of architectures, utilizing convolutions [26, 32, 38] and autoencoder structures [7, 31]. GNN based representations have been used for various analytics tasks, including node similarity search [13], link prediction [43], and entity disambiguation [53]. Solutions for scaling GNNs mostly focus on distributed processing [10, 62, 65]. Scalability challenges on large graphs have also been studied in the context of memory optimizations [48] and scalable processing [37, 44, 45].

GNN quantization have started to receive attention in recent years. Tailor et al. [51] propose quantization-aware training for GNNs, where high in-degree nodes are selected for full-precision training while all other nodes are converted to INT8/INT4. This can achieve reasonable accuracy especially on INT8 models. Huang et al. [27] employ product quantization to compress input data but do not address the more challenging task of quantization of parameters. A recent GNN quantization approach [67] addresses low-bit representation of the weights and input features by learning the parameters that are equal with the weight dimension and the number of input nodes, respectively, while leaving the core message propagation unquantized. However, this approach necessitates the learning of parameters that scale proportionally with the number of input nodes, resulting in considerable storage and space overheads. Neural Architecture Search (NAS) is used to span possible quantization levels suggesting an INT4 weight and INT8 activation as an effective strategy for GNNs [64]. Recent studies adapt binary NN methods for GNNs [1, 54] offering a trade-off between time/space efficiency and classification accuracy. These methods typically either need an additional teacher model for knowledge distillation or learn binary weights for each layer's input message, which require higher storage and computational load than a typical quantization based approach.

Towards addressing oversmoothing in deep GNNs, Liu et al. [35] propose Elastic Graph Neural Network with long-range information propagation using ℓ_1 and ℓ_2-based graph smoothing. APPNP [19] addresses the oversmoothing with a propagation scheme based on an approximation of personalized PageRank. Zhu et al. [66] proposed low-pass and high-pass filtering kernels which have empirically reduced the effect of oversmoothing. DropEdge [42] aims to address the oversmoothing by dropping a number of edges, which can be interpreted as both a data augmentation method generating random deformed graphs and message passing reducer by sparsifying edge connections. PairNorm [63] quantifies the oversmoothing and proposes a two-step center-and-scale normalization layer to prevent nodes converging to similar representations. Compared to enforcing local smoothness, our method, constrains the layer-wise message propagation to counteract oversmoothing, which achieves performance improvements over the prior approaches as also demonstrated in our experiments.

3 PRELIMINARIES AND ANALYSIS

We first provide the technical background, covering quantization for GNNs, analysis of quantization errors, and the oversmoothing problem in GNNs.

3.1 GNN Basics

A graph $\mathcal{G}$ is represented as $\mathcal{G} = (V, E, \mathbf{X})$, where $V = \{v_1, \cdots v_n\}$ is the set of n nodes $|V| = n$, E is the set of all edges, $\mathbf{H^l} = [\mathbf{h}_1^l, \cdots, \mathbf{h}_n^l]^\top$ is the node feature (embedding) matrix for layer $l \in L$ where L represents the number of layers in $\mathcal{G}$, and $\mathbf{h}_i \in \mathbb{R}^{d_l}$ is the feature vector for $v_i \in V$ node with initial $\mathbf{H^0} = \mathbf{X}$. The adjacency matrix of $\mathcal{G}$ is a binary matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}(i, j) = 1$ if the edge between nodes v_i and v_j exists ($e_{ij} \in E$), and 0 otherwise.

GNNs comprise a sequence of layers with three main functions for each layer: message, aggregate and update. This framework is generally called Message Passing NNs (MPNN) [21]. Each message, which is a flow of data from nodes' neighbors, is aggregated and joined with existing embedding to form a new one for the respective node as given in Equation 1, where $\mathcal{N}_u$ are the neighboring nodes of node u. The feed-forward iteration starts with the initial embeddings, $\mathbf{h}_u^0$=$\mathbf{x}_u$.

$$\mathbf{h}_u^{(l+1)} = Update_l(\mathbf{h}_u^l, Aggregate_l(\mathbf{h}_v^l, \forall v \in \mathcal{N}_u) \tag{1}$$

Various GNN architectures are proposed in the literature, essentially, varying the message, aggregate and update functions [59]. We consider, the popular, GCN (Graph Convolutional Network) architecture where the update function given in Equation 2 with activation function σ and learnable weight matrix $\mathbf{W}_l$ [32].

$$\mathbf{h}_{(l+1)}^u = \sigma\Big(\sum_{v \in \mathcal{N}^u \cup u} \frac{1}{\sqrt{d_u d_v}} \mathbf{W}_l \mathbf{h}_l^v) \Big) \tag{2}$$

Recently, a different perspective on common GNN models was proposed by Ma et al. [36], where the authors unified different GNN models, such as GCN, GAT, PPNP, and APPNP, by posing them as solutions to the graph denoising problem.

$$\underset{\mathbf{H}}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{H} - \mathbf{X}\|_F^2 + \frac{\mu}{2}\mathbf{tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H}) \tag{3}$$

where $\mathbf{H}$=$[\mathbf{h}_1, \cdots, \mathbf{h}_n]^\top \in \mathbb{R}^{n \times d}$, $\mathbf{L} \in \mathbb{R}^{n \times n}$ represent final representations and the graph Laplacian matrix respectively. The first term drives the embeddings $\mathbf{H}$ closer to the graph input features $\mathbf{X}$ while the second term imposes global smoothness by enforcing similarity amongst the connected nodes, as

$$\mathbf{tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H}) = \sum_{(v_i, v_j) \in E} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2 \tag{4}$$

Following this, **EMP** (Elastic Message Passing) [35] method was proposed enabling ℓ_1 based smoothing constraints on GNNs.

3.2 Challenges with Deeper GNNs

While in traditional ML, deeper models can extract more powerful representations, for GNNs this inherently leads to several major challenges. First, as the depth increases, GNNs demand exponentially more computations and larger storage to be managed and processed, which makes their deployment on resource constrained platforms more challenging. We seek to design an inference-friendly quantizer, i.e., performing inference directly on quantized elements with high accuracy. Second, deeper GNNs suffer from the oversmoothing problem, where node representations converge to indistinguishable embeddings, degrading accuracy of downstream tasks. It was shown that GCN exponentially loses its expressive power for node classification tasks in many practical cases [39].

There are some proposals towards mitigating the oversmoothing problem for full precision models [19, 34, 35, 42, 63] as discussed in Section 2, including **DropEdge**, **PairNorm**, **APPNP** and **EMP**. Our experiments confirm that **DropEdge** and **PairNorm** are particularly ineffective for low-bit quantization. These methods do not consider the smoothness of message propagation amongst layers, resulting in accuracy drops and unrestricted quantization error especially in low-bit cases. In contrast, we seek layer-wise smoothness by enforcing constraints at message propagation, restricting the quantization error, and denoising the message passing procedure which lead to enhanced accuracy in low-bit quantization.

3.3 Quantization Basics

Quantization is the process of mapping continuous data, e.g., parameters, weights and activations of neural networks, to smaller sized representations. In the scope of our analysis, we denote U (e.g. $\mathbf{H^l}$) as a high-precision tensor-valued random variable with probability density function $f_U(u)$. A tensor is commonly quantized between its maximum and minimum *observed* values [29]. Considering observed values as $U_o \in [\alpha, \beta]$ and the corresponding b-bit quantized values as $U_q \in [\alpha_q, \beta_q]$, the quantization function is given by

$$Q(U, s, z) = \text{clip}\Big(\lfloor \frac{U}{s} + z \rceil, \alpha_q, \beta_q \Big) = U_q \tag{5}$$

where s=$\frac{\beta - \alpha}{\beta_q - \alpha_q}$ is the scale, $\lfloor \cdot \rceil$ denotes the round function, and z=$\lfloor \frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha} \rceil$ is the zero point. The corresponding de-quantization function is as follows

$$D(U_q, s, z) = s(U_q - z) \tag{6}$$

The range $[\alpha, \beta]$ is usually partitioned into 2^b equal interval regions with a quantization step Δ=$\frac{\beta - \alpha}{2^b}$. Given that de-quantized

value in Equation 6 is represented as $\hat{U}$, the mean squared error (MSE) between U and $\hat{U}$ is given by

$$E[(U-\hat{U})^2] = \int_{-\infty}^{\alpha} f_U(u)(u-\hat{\alpha})^2 du + \int_{\beta}^{\infty} f_U(u)(u-\hat{\beta})^2 du + \sum_{i=0}^{2^b-1} \int_{\alpha+i\Delta}^{\alpha+(i+1)\Delta} f_U(u)(u-\hat{u})^2 du \quad (7)$$

The MSE consists of three terms. The first two items are *overload distortion* caused by clipping the values of u beyond $[\alpha, \beta]$. The third term means *granular distortion* led by the quantization step Δ. For any $u \in [\alpha, \beta]$, its granular distortion is in $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$. Therefore, it can be reduced by setting an appropriate Δ based on the distribution of U. This becomes particularly critical in GNN quantization, as GNNs show large variance at aggregated values [51].

3.4 Challenges with GNN Quantization

Compared to quantizing CNNs, GNNs involve more types of elements to be quantized with complex interdependencies. These elements include inputs of each layer, weights, messages between nodes, inputs and outputs of aggregation stage, and outputs of update stage. Since the variance of the updated features after propagation in GNNs is high due to the varying number of neighbors [51], it is particularly challenging to design a low-bit uniform quantizer. Tailor et al. [51] use percentiles to manually decide the quantization range $[\alpha, \beta]$, and a *momentum* parameter to perform weighted average of the statistics of tensors during training. We empirically observe that the accuracy is highly sensitive to the setting of percentiles and *momentum*, which increases the difficulty of obtaining accurate results especially using low number of bits. Zhu et al. [67] learn the quantization step size for each node of input features and each dimension of weights, respectively. However, as it does not quantize the message propagation part, the resulting model size and computations are significantly larger. Moreover, learning parameters per each node yields a higher model parameterization and limits its inductive capabilities including mini-batch training. It is akin to applying N times learned step size [15], where N is the number of nodes in the graph. To address these challenges, we introduce a quantizer with learnable ranges (QLR) which determines the quantization range and is also friendly for mini-batch training on large datasets.

4 LOW-BIT QUANTIZATION FOR GRAPH NEURAL NETWORKS

This section describes our solution for quantization with learnable ranges (QLR) and a skewness-aware bitwise truncation (BT*) that captures the underlying data distribution to preserve accuracy with low-bit representations.

4.1 Quantization with Learnable Range

GNN involves various components such as layer activations, weights, messages, inputs/outputs of the aggregation and update stages. We aim to quantize all of the aforementioned components to reduce the model size and maintain high accuracy during inference.

According to the quantization error analysis in Section 3.3, given quantization level $[\alpha_q, \beta_q]$, s is directly influenced by different settings of $[\alpha, \beta]$. Using this observation, we design a scaling parameter

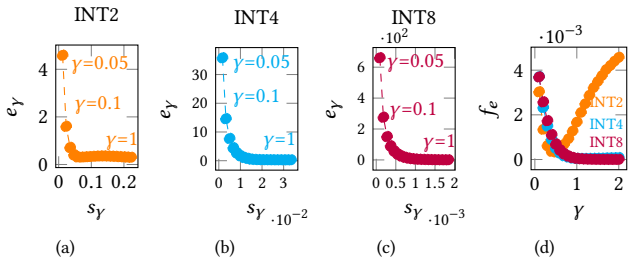


Figure 1: Quantization error with different γ

γ to modify $[\alpha, \beta]$ into a more data-aware range $[\gamma\alpha, \gamma\beta]$, which updates the quantization range to reduce the quantization error in low-bit cases.

With the learnable quantization range $[\gamma\alpha, \gamma\beta]$, the quantization function can be updated as

$$Q(U, s_\gamma, z_\gamma) = \text{clip}(\lfloor \frac{U}{s_\gamma} + z_\gamma \rceil, \alpha_q, \beta_q) = U_{q,\gamma} \quad (8)$$

where $s_\gamma = \frac{\gamma(\beta-\alpha)}{\beta_q-\alpha_q} = \gamma s$ is the updated scale, while zero point stays the same $z_\gamma = \lfloor \frac{\gamma\beta\alpha_q - \gamma\alpha\beta_q}{\gamma\beta-\gamma\alpha} \rceil = z$. The de-quantization function is modified accordingly

$$D(U_{q,\gamma}, s_\gamma, z_\gamma) = s_\gamma(U_{q,\gamma} - z_\gamma) = \hat{U}_\gamma \quad (9)$$

To optimize γ at the backward propagation, Straight-Through Estimator [5] can be used to calculate the gradient of γ as

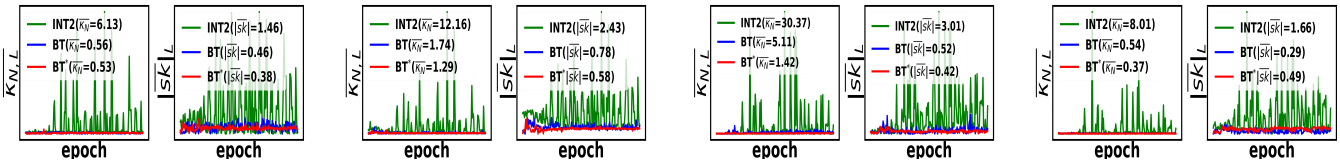
$$\frac{\partial \hat{U}_\gamma}{\partial \gamma} = \begin{cases} \alpha_q, & u \le \alpha_q \\ \beta_q, & u \ge \beta_q \\ s\lfloor \frac{u}{\gamma s} \rceil - \frac{u}{\gamma}, & \alpha_q < u < \beta_q \end{cases} \quad (10)$$

QLR learns a scale (γ) relative to the quantization range of observed values, allocates the limited quantization budget to the remaining observed data points while accounting for the final task. Notably, this is different from learned step size quantization (LSQ) [15], which optimizes the step size over the full observed values. We have empirically observed that LSQ tends to be highly sensitive to the learning rate. This means that achieving satisfactory accuracy often requires an exhaustive search for the proper hyperparameters. The challenges with LSQ are further amplified because, in GNNs, the value ranges can differ significantly across layers, leading to uneven convergence rates between them.

Quantization error analysis for QLR. Given a value $u \in [\alpha, \beta]$, its quantization error can be written as

$$f_e(u) = \|u - \hat{u}\|^2 = \|s_\gamma(\frac{u}{s_\gamma} - \lfloor \frac{u}{s_\gamma} \rceil)\|^2 \quad (11)$$

It comes from two sources, s_γ and $\frac{u}{s_\gamma} - \lfloor \frac{u}{s_\gamma} \rceil$, which are the quantization level and distortion caused by rounding operation, respectively. Figure 1 shows the distortion error e_γ and total quantization error $f_e(u) = \|s_\gamma e_\gamma\|^2$, for aggregate output of Cora dataset, across varying scales in $\gamma \in [0.05, 1.0]$. Noteworthy that, e_γ and s_γ affect the error in different directions across different γ varying from INT2–INT8. Specifically, with an optimized γ, the distortion in INT2 can be reduced to a scale similar to that of INT4 and INT8 as shown in Figure 1(d). Hence, a learnable γ can optimize the quantization range, reducing the total error even in extreme low-bit representations.



(a) Cora-Message (b) Cora-Aggregate (c) CS-Message (d) CS-Aggregate

Figure 2: Kurtosis and Skewness of different datasets at each epoch

4.2 Skewness-aware Bitwise Truncation

A basic approach for INT2 can be simply keeping the most (two) significant bits of the higher precision output (e.g., INT8, INT4) as

$$Q_{b_2 \leftarrow b_1}(U_q, s_0) = \lfloor \frac{U_q}{s_0} \rceil s_0 \tag{12}$$

where b_1 and b_2 ($b_1 \geq b_2$) are the number of bits used for quantization, $b_2 \leftarrow b_1$ means the b_1-bit quantized representation being truncated into b_2 bits; U_q denotes the b_1-bit representation obtained with Equation 8, and s_0 is the scale for truncating the low-significant bit representation depending on b_1 and b_2. s_0 can be obtained as

$$s_0 = \frac{\alpha_{q_1} - \beta_{q_1}}{\alpha_{q_2} - \beta_{q_2}} \tag{13}$$

where $[\alpha_{q_1}, \beta_{q_1}]$ and $[\alpha_{q_2}, \beta_{q_2}]$ are the quantization levels for b_1-bit and b_2-bit quantization, respectively.

While such formulation implicitly assumes the uniformity of U, for GNNs this can significantly vary depending on the graph topology. Measures such as kurtosis (κ) and skewness (sk) [22] can be employed to better capture information about normality and symmetry of the distribution respectively. While prior methods assumed that the neural networks activations follow close-to-normal distributions, we have empirically observed that, for GNNs, these have relatively large kurtosis and are rather asymmetrical on low-bit quantization (Figure 2). All these bring further challenges in employing bitwise truncation (BT) for GNNs.

We formulate a data-aware truncation mechanism that accounts for skewness of input data. Skewness-aware BT (BT*), defined in Equation 14, can capture the abnormal distribution of quantized elements even under low-bits.

$$\hat{Q}_{b_2 \leftarrow b_1}(U_q, s_0) = \lfloor \frac{U_q + \lfloor sk \rceil}{s_0} \rceil s_0 \tag{14}$$

where sk is the skewness of input tensor U.

Figure 2 illustrates the kurtosis and skewness parameters of the message passing and aggregate output blocks for 10-layer SMP, detailed later in Section 5, using INT2, INT2-8 (BT) and INT2-8* (BT*) quantization across two datasets. We note that kurtosis ($|\kappa|$) and skewness of the normal distribution are 3 and 0 respectively. Therefore, $\kappa_N = |\kappa - 3|$ can be used to measure the normality of the tensor, where smaller κ_N means a distribution closer to normal. The average kurtosis of BT* remains continuously smaller throughout each epoch, as compared to INT2 and BT, which indicates a robust training process. Similar trends are observed for the skewness sk: values for BT* fall in $(-1.0, 1.0)$ range. As a result, using skewness (sk) in the bitwise truncation process, as provided in Equation 14, maintains the symmetry of the quantized elements while ensuring their normality.

5 LAYER-WISE SMOOTHNESS-AWARE MESSAGE PROPAGATION

In GNN learning, each node's feature consists of a true signal, which relates to its class, and a noise component. The essence of message passing is to increase the signal-to-noise ratio by adaptively aggregating node features. However, unexpected or out-of-distribution features from a neighboring node, possibly of different class, can adversely affect the goal of enhancing signal-to-noise ratio. In the asymptotic case, aggregating features from different classes can cause a blending of true features, resulting in oversmoothing. The layer-wise smoothness that preserves locality between layers of GNN, can be helpful in achieving deeper GNNs [35, 36].

In Section 4.1, we introduced a quantizer that reduces the quantization error by learning an optimal quantization range. We also need to ensure its efficiency with respect to increasing model depth. From our empirical analyses, it is evident that the observed quantization range (β -α) for low-bit setting expands as the number of layers grows (Figure 3). Based on Equation 7, an expanded quantization range directly influences its error. This suggests that quantization may further compromise the accuracy of deeper models which already suffer from over-smoothing. This potential degradation of accuracy is also reflected in Figure 3, where we measure it on GCN with INT2 quantization.

Motivated by the above observations, we devise a layer-wise smoothness approach that brings forth two primary benefits. Firstly, it facilitates smooth message propagation, thereby mitigating the problem of oversmoothing. Secondly, it helps to address the challenges of obtaining satisfactory, low-bit representations caused by substantial and abrupt updates during message propagation.

Outline: In this section, we present our Smoothness-aware Message Propagation (SMP) solution which aims to reduce the oversmoothing effect and suppress the quantization error to a bound. We first quantify the layer-wise smoothness and analyze the local smoothness of existing GNNs at message propagation. We then present the SMP mechanism that smooths the message propagation with a graph denoising approach. After transforming the optimization problem into a Lagrangian function, we develop an optimal solution involving a differential multiplier method (BDMM). We also prove the existence of the quantization error bound for quantized SMP. The results presented in Figure 3 (GCN+SMP) confirm that SMP can also help improving the general GCN in INT2 quantization and deeper layer settings.

5.1 Layer-wise Smoothness

We quantify the smoothness objective in Definition 5.1 by measuring the layer-wise local smoothness during message propagation between each GNN layer.

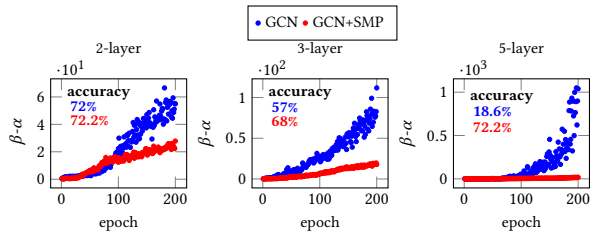


Figure 3: Effect of layer-wise smoothness on the aggregate output quantization range in INT2 quantization

Definition 5.1. (Layer-wise Smoothness) Given a graph $\mathcal{G}$=(V, E, $\mathbf{X}$), the l-th layer-wise smoothness is the change of connected nodes $\forall(v_i, v_j) \in E$ with a degree normalization from layer l-1 to layer l.

The layer-wise smoothness can be formulated as

$$\mathbf{S}_l = \sum_{(v_i,v_j)\in E} \|(\frac{\mathbf{h}_i^l}{\sqrt{d_i}} - \frac{\mathbf{h}_j^l}{\sqrt{d_j}}) - (\frac{\mathbf{h}_i^{l-1}}{\sqrt{d_i}} - \frac{\mathbf{h}_j^{l-1}}{\sqrt{d_j}})\|_2^2 \tag{15}$$

Specifically, $\mathbf{S}_l$ can also be represented as

$$\begin{aligned}\mathbf{S}_l &= \sum_{(v_i,v_j)\in E} \|\frac{(\mathbf{h}_i^l - \mathbf{h}_i^{l-1})}{\sqrt{d_i}} - \frac{(\mathbf{h}_j^l - \mathbf{h}_j^{l-1})}{\sqrt{d_j}}\|_2^2 \\ &= \mathbf{tr}((\mathbf{H^l} - \mathbf{H^{l-1}})^\top \tilde{\mathbf{L}}(\mathbf{H^l} - \mathbf{H^{l-1}}))\end{aligned} \tag{16}$$

where $\tilde{\mathbf{L}}$ represents the normalized Laplacian matrix, $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{A}$, and $\mathbf{D_{ii}} = d_i = \sum_j \hat{\mathbf{A}}_{ij}$. The $\mathbf{tr}(\mathbf{H}^\top\tilde{\mathbf{L}}\mathbf{H})$ is the Laplacian regularization to make $\mathbf{H}$ smooth over graph G, similarly, the l-th layer-wise smoothness $\mathbf{tr}((\mathbf{H^l} - \mathbf{H^{l-1}})^\top\tilde{\mathbf{L}}(\mathbf{H^l} - \mathbf{H^{l-1}}))$ can be explained as smoothing the changes from layer l-1 to l over G.

5.2 Smoothness-aware Message Propagation

SMP is designed to guide the training process to achieve local smoothness at message propagation of each layer, by utilizing the smoothness measure presented in Definition 5.1. Intuitively, SMP aims to avoid drastic correlation/similarity changes for connected nodes to achieve local smoothness at each message update.

We formulate the SMP objective based on graph denoising formulation (at each layer $l \in L$) with degree normalization,

$$\begin{aligned}\mathbf{H}^* = \underset{\mathbf{H}}{\text{argmin}} \quad & \frac{1}{2}\|\mathbf{H}-\mathbf{X}\|_F^2 + \frac{\mu}{2}\mathbf{tr}(\mathbf{H}^\top\tilde{\mathbf{L}}\mathbf{H}) \\ \text{subject to: } & \mathbf{S}_l \le \delta = \delta_0|E|, \forall l \in L\end{aligned} \tag{17}$$

where $\mathbf{S}_l$=$\mathbf{tr}((\mathbf{H} - \mathbf{H^{l-1}})^\top\tilde{\mathbf{L}}(\mathbf{H} - \mathbf{H^{l-1}}))$, $\tilde{\mathbf{L}}$ represents the normalized Laplacian matrix, $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{A}$, and $\mathbf{D_{ii}} = d_i = \sum_j \hat{\mathbf{A}}_{ij}$.

The optimization objective aims to find an optimal $\mathbf{H}^*$ which we assume to be the correct feature embedding for the particular graph. We impose three different priors to extract this optimal embedding. The first term minimizes the distance to measure the original feature matrix ($\mathbf{X}$), the second term imposes neighborhood similarity in Equation 17. These two objectives have been used in different methods in the literature. In SMP, we impose a new constraint (with $\mathbf{S}_l$) which limits the change of embeddings between layers of the GNN and makes smooth transitions at each message passing iteration. δ_0 is the threshold for controlling an allowed variation between the correlations/similarities for the connected node between layers, $|E|$ is the number of edges in the graph.

This formulation shows that the constraint aims to mitigate the abrupt changes in the relations between connected nodes due to possibly interfering signals coming from neighboring nodes. Alternatively, $\mathbf{S}_l$ can be configured to capture the changes of the smoothness with different distance measures, e.g., ℓ_1 norm.

The Lagrange function for the objective function at layer l has the following form

$$\mathcal{L}^l(\mathbf{H},\lambda,s) = \underbrace{\frac{1}{2}\|\mathbf{H}-\mathbf{X}\|_F^2 + \frac{\mu}{2}\mathbf{tr}(\mathbf{H}^\top\tilde{\mathbf{L}}\mathbf{H})}_{f(\mathbf{H})} + \lambda g(\mathbf{H},\mathbf{H}^{l-1},s) \tag{18}$$

where s is slack variable, λ is Lagrangian multiplier, and $g(\mathbf{H}, \mathbf{H}^{l-1}, s)$ = $\delta - \mathbf{S}_l - s^2$.

Equation 18 is differentiable with respect to $\mathbf{H}$, λ and s. However, Lagrangian multiplier method does not directly work with gradient descent optimization and the derivation of optimal solution from KKT (Karush–Kuhn–Tucker) conditions will be cumbersome when the constraints are complex as in SMP case. Therefore, the optimal solution to Equation 18 can be derived with the basic differential multiplier method (BDMM) [41], which has been proved to optimize Lagrange multipliers in conjunction with the objective argument in a sequential manner. The BDMM updates are given below

$$\begin{cases}\overline{\mathbf{H}}^{l+1} = \mathbf{H}^l - \eta_{\mathbf{H}}\nabla f(\mathbf{H}^l) \\ \mathbf{H}^{l+1} = \overline{\mathbf{H}}^{l+1} - \eta_{\mathbf{H}}\lambda^l\nabla_{\mathbf{H}} g(\overline{\mathbf{H}}^{l+1}, \mathbf{H}^l, s^l) \\ s^{k+1} = s^l + 2\eta_s\lambda^l s^l \\ \lambda^{k+1} = \lambda^l + \eta_\lambda g(\mathbf{H}^{l+1}, \mathbf{H}^l, s^{l+1})\end{cases} \tag{19}$$

When we calculate the respective gradients in Equation 18 and incorporate into Equation 19, we easily reach the formulation as

$$\begin{cases}\overline{\mathbf{H}}^{l+1} = (1 - (1+\mu)\eta_{\mathbf{H}})\mathbf{H}^l + \mu\eta_{\mathbf{H}}\tilde{\mathbf{A}}\mathbf{H}^l + \eta_{\mathbf{H}}\mathbf{X} \\ \mathbf{H}^{l+1} = \overline{\mathbf{H}}^{l+1} + 2\eta_{\mathbf{H}}\lambda(\mathbf{I} - \tilde{\mathbf{A}})(\overline{\mathbf{H}}^{l+1} - \mathbf{H}^l) \\ s^{l+1} = s^l + 2\eta_s\lambda^l s^l \\ \lambda^{l+1} = \lambda^l + \eta_\lambda g(\mathbf{H}^{l+1}, \mathbf{H}^l, s^{l+1})\end{cases} \tag{20}$$

where $\eta_{\mathbf{H}}$, η_λ, and η_s are the respective step sizes.

Variation of S_l for existing GNNs. To provide an intuitive understanding of layer-wise smoothness ($\mathbf{S}$), we measure $\mathbf{S}$ for SMP and existing GNNs to quantitatively show how the measure varies with different GNN solutions. We compute $\mathbf{S}$ on 10-layer GCN, SMP, and several existing deep GNN solutions, including DropEdge, APPNP, EMP, and PairNorm. Figure 4 shows the average layer-wise smoothness of 10-layer GNNs ($\overline{\mathbf{S}}$) at each epoch, where $\overline{\mathbf{S}}$=$\frac{\sum_{l\in L}\mathbf{S}_l}{L-1}$ (l∈[2,L]). The GCN, without any safeguard mechanism for oversmoothing, creates extremely large layer-wise variations (higher $\overline{\mathbf{S}}_l$) when compared to deep GNN solutions. This example illustrates that deep GNN methods mitigating against oversmoothing are effective to control and increase layer-wise smoothness (decreasing $\overline{\mathbf{S}}_l$) when compared with the general GCN. Additionally, the $\overline{\mathbf{S}}$ of SMP drops continuously when compared with other comparable deep GNNs. These experiments show that the iterative solution in Equation 20 is effective in controlling the change between layers by enforcing both node-wise smoothness and layer-wise smoothness.

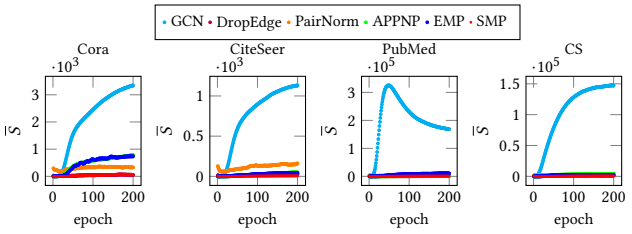


Figure 4: Avg layer-wise smoothness against different epochs ($\bar{S}$) for GNNs

SMP Contribution to Quantization. The quantization error for the l-th layer representation can be expressed as $f_e^l = \|\mathbf{H}^l - \mathbf{H}^{l,q}\|_2^2$, where $\mathbf{H}^{l,q}$ is the quantized representation of $\mathbf{H}^l$. Accordingly, for the quantized SMP, the smoothness constraint can be written as $S_{l,q} = \mathbf{tr}((\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q})^\top \tilde{\mathbf{L}}(\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q}))$. We can prove f_e^l is smaller than a bound as provided in LEMMA 1, which underlines the superiority of SMP in terms of quantization.

LEMMA 1. *For the l-th layer representation $\mathbf{H}^l$, the quantization error is $f_e^l \leq l\delta\mathbf{tr}(\mathbf{\Lambda}^{-1}) + \|\mathbf{H}^l - \mathbf{X}^q\|_2^2$*

PROOF. The Laplacian matrix $\tilde{L}$ is eigendecomposable, i.e., $\tilde{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is orthogonal matrix ($\mathbf{U}\mathbf{U}^\top = \mathrm{I}$). $\mathrm{S}_{l,q}$ can be represented as $\mathrm{S}_{l,q} = \|\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top(\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q})\|_2^2 \leq \delta$. The derivation process is summarized as follows

$$\begin{cases} \|\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top(\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q})\|_2^2 = \|(\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q})\mathbf{\Lambda}^{\frac{1}{2}}\|_2^2 \leq \delta & ① \\ \|\mathbf{H}^{l,q} - \mathbf{H}^l\|_2^2 - \|\mathbf{H}^{l-1,q} - \mathbf{H}^l\|_2^2 \leq \|\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q}\|_2^2 \leq \\ \|(\mathbf{H}^{l,q} - \mathbf{H}^{l-1,q})\mathbf{\Lambda}^{\frac{1}{2}}\|_2^2\|\mathbf{\Lambda}^{-\frac{1}{2}}\|_2^2 \leq \delta\mathbf{tr}(\mathbf{\Lambda}^{-1}) & ② \\ f_e^l \leq \delta\mathbf{tr}(\mathbf{\Lambda}^{-1}) + \|\mathbf{H}^{l-1,q} - \mathbf{H}^l\|_2^2 & ③ \\ \|\mathbf{H}^{i-1,q} - \mathbf{H}^l\|_2^2 \leq \|\mathbf{H}^{i-1,q} - \mathbf{H}^{i-2,q}\|_2^2 + \|\mathbf{H}^{i-2,q} - \mathbf{H}^l\|_2^2 \\ \quad = \delta\mathbf{tr}(\mathbf{\Lambda}^{-1}) + \|\mathbf{H}^{i-2,q} - \mathbf{H}^l\|_2^2, \text{ where } 1 \leq i \leq l. & ④ \\ f_e^l \leq l\delta\mathbf{tr}(\mathbf{\Lambda}^{-1}) + \|\mathbf{H}^l - \mathbf{X}^q\|_2^2 & ⑤ \end{cases}$$

□

6 EXPERIMENTS

This section presents our experiments on benchmark datasets that illustrate the effectiveness of QLR and QLR with BT (BT*) quantizers under low-bit settings. We also compare our SMP with comparable deep GNN baselines, highlighting the capability of SMP in addressing the oversmoothing issue.

6.1 Experimental Setup

Datasets and Baselines. Our experiments are performed on five datasets, Cora, PubMed, CiteSeer [47], CS [49] and Reddit [23] in a semi-supervised node classification setting. The statistics of the datasets are summarized in Table 1.

Table 1: Statistics of benchmark datasets

Dataset	Cora	CiteSeer	PubMed	CS	Reddit
Nodes	2708	3327	19717	18333	232965
Edges	5278	4552	44324	81894	114848857
Features	1433	3703	500	6805	602
Labels	7	6	3	15	41

We start by comparing QLR against two state-of-the-art GNN quantizers, Degree-Quant [51] and Aggregate-Quant [67], on GCN [32]. These experiments are complemented with their respective model sizes. To showcase the effectiveness of SMP for quantization and oversmoothing we compare with 4 comparable deep GNN methods, which are, APPNP [19], DropEdge [42], PairNorm [63] and EMP [35]. They are evaluated on 10-layer GNNs with 64 hidden units. Subsequently, we apply BT (BT*) within SMP and EMP pipeline to verify its effectiveness on extreme low-bit representations and its scalability with respect to the numbers of layers. Finally, we present the runtime efficiency by measuring the throughput under the representations of various quantization levels.

Parameter settings. For APPNP, DropEdge, PairNorm and EMP, we used the optimal parameters provided within their public repositories. For SMP, we set the parameters from the following search space: (1) learning rate (lr) $\in$ {0.005, 0.008, 0.01, 0.015}; (2) weight decay (wd) $\in \{5e^{-4}, 1e^{-4}, 5e^{-5}\}$; (3) drop rate $\in$ {0.8}; (4) $\mu \in$ {3, 6, 9}; (5) the initial value of $\gamma \in$ {1.0} ;(6) η_λ, $\eta_s \in \{1e^{-5}, 1e^{-6}\}$; (7) $\delta_0 \in$ {0.01, 0.1, 0.5, 1, 2}. Due to a significant difference in magnitude between the gradients of the scaling parameters (γ) and other GNN parameters, we have established a distinct search space for the former: learning rate ($\mathrm{lr}_\gamma \in$ {0.001, 0.002}) and weight decay ($\mathrm{wd}_\gamma \in \{1e^{-4}, 5e^{-5}\}$).

We present the average accuracy and standard deviation over 10 random data splits for Cora and CiteSeer, and 5 for PubMed, CS and Reddit. For Reddit dataset, owing to its size, we have employed mini-batch training with a batch_size of 20000. All of the experiments are based on Pytorch [40] and PyTorch Geometric [18]. The experiments are ran on Ubuntu 20.04 with 64GB RAM.

6.2 Comparison with different quantizers

We compare QLR with the state-of-the-art GNN quantization solutions, Degree-Quant and Aggregate-Quant. Results are summarized in Table 2.

We notice that Aggregate-Quant in default maintains a fixed quantization level of INT4 for weights, while having smaller bits for input features. Moreover, it does not quantize the message-passing blocks of GCN, whereas QLR and Degree-Quant quantize all the elements equally. Hence, for fairness, we also add quantizers for its message-passing blocks and removed the INT4 constraint on its model weights. Also important to note that Aggregate-Quant maintains a step size parameter for each node, which can be viewed as an extension of learned step size quantization (LSQ) [15], which makes it highly inflexible for inductive tasks. Due to that, it does not support mini-batch training, as the topology of the input graph changes with each mini-batch training iteration.

We observe that performance of QLR significantly outperforms those of its competitors irrespective of the quantization level. The approach of optimizing the quantization range in the backward pass makes QLR more robust and effective, especially in low-bit cases. Aggregate-Quant demonstrates superior performance for CiteSeer when applied to INT8 quantization, which can be courtesy of its significantly larger parameter size. However, the accuracy of low-bit cases degrades significantly when message passing blocks are also quantized fairly. As for Degree-Quant, while it can achieve comparable performance on INT8, it cannot generate expected performance with INT4 and INT2 quantization on Reddit. Due to its mask sampling strategy and low quantization level, one node

Table 2: Classification Accuracy (%) of Different Quantization on GCN

Method		QLR			Aggregate-quant			Degree-quant		
Dataset	FP	INT8	INT4	INT2	INT8	INT4	INT2	INT8	INT4	INT2
Cora	80.79±1.54	**81.03±1.03**	80.40±1.06	74.56±2.35	78.57±3.09	59.03±2.50	–	80.64±1.47	77.81±1.22	–
Citeseer	67.74±**2.29**	67.59±2.07	67.10±2.73	60.12±3.02	**68.09**±1.83	47.79 ±4.41	–	67.62±2.08	66.10±1.97	–
PubMed	78.00±2.07	**78.74±2.56**	78.10±2.58	71.40±4.24	77.60 ±2.07	68.80 ±0.75	62.26 ±0.38	77.48±3.36	72.38±4.03	–
CS	90.42±0.44	**90.51±0.45**	88.42±1.62	71.54±6.97	89.45±0.65	40.52 ±6.74	–	89.96±0.58	82.51±3.09	–
Reddit	94.04 ±0.18	**94.13±0.07**	90.46 ±0.08	85.19 ±1.36	×	×	×	93.75 ±0.20	–	–

– denotes accuracy ≤ 40.00%, × indicates the quantization method does not support mini-batch training

Table 3: Model size (MB) with different number of hidden units (d) on different quantization methods

d		QLR			Aggregate-quant			Degree-quant		
	FP	INT8	INT4	INT2	INT8	INT4	INT2	INT8	INT4	INT2
64	1.75	0.441	0.223	0.114	3.70	3.48	3.37	0.438	0.220	0.111
128	3.49	0.878	0.441	0.223	4.19	3.75	3.53	0.875	0.438	0.220
512	14.0	3.500	1.770	0.878	7.79	6.05	5.17	3.490	1.750	0.875

sampled to different mini-batches will generate different representations at different batch training, which curbs the overall accurate representation. However, QLR can directly optimize the learnable quantization range based on the observations of subgraphs, hence, it can reduce the comprehensive quantization errors. These further confirm that optimizing the quantization range in QLR enables better preservation of accuracy in low-bit representations.

It is noteworthy that QLR preserves its accuracy results even for INT2 quantization across all datasets, while the alternatives fail to get comparable accuracy. Additionally, QLR even outperforms the full precision (FP) model in INT8 quantization in many cases, showcasing its effectiveness as a noise filter for GNNs.

In Table 3, we report the model sizes of different quantization approaches with varying quantization levels and hidden units (d). Due to space limit, we only present the model sizes on CS dataset. As there is a native 8-bit support, under constant d, the sizes of INT8 with QLR and Degree-Quant are consistently reduced to approximately one-fourth of the FP counterpart. For smaller bits, however, we pack INT2 and INT4 similar to the process described in [30]. The size of QLR is slightly larger to that of Degree-quant due to storage of s_γ, z_γ and γ parameters. Given the superior accuracy performance of QLR in low-bit settings, its slight increase in model size becomes negligible in comparison. Overall, with QLR and Degree-quant, the INT2 and INT4 model sizes are significantly smaller than their FP counterparts, with reductions of 16× and 8×, respectively. However, the size of Aggregate-Quant is 2–6 times that of its counterparts, largely due to the dimension and per-node nature of the parameters.

6.3 Comparisons with existing deep GNNs

6.3.1 Node classification with existing deep GNNs. We compare SMP with the existing deep GNNs in terms of both full-precision (FP) and quantized models using QLR. Table 4 presents the classification accuracy results using 10-layer GNN. Notably, SMP consistently outperforms the alternative methods on Cora, CiteSeer and CS with FP models, and slightly lower than that of APPNP on PubMed. SMP improves over EMP by enforcing smoothness at layer-wise message propagation during training and inference.

For quantized models, although INT8 achieves high accuracy close to FP for all methods, INT4 performance of DropEdge and PairNorm drops significantly, rendering it incomparable in some cases, and throws OOM errors on larger datasets. This is primarily due to the complexity of the "backbone" models [42]. For example, PairNorm, employing a GCN backbone, trains a weight matrix for each layer. Likewise, DropEdge utilizes more intricate backbones such as GCN, ResGCN [25], InceptGCN [50], and introduces connection perturbations at every layer. Consequently, these factors contribute to higher computational and storage requirements, which further escalate with the depth of the model. On the contrary, EMP, APPNP, and SMP employ much simpler architectures that involve only two weight matrices prior to GNN propagation. As a result, these models have more moderate and scalable requirements, making them more amenable for deep GNN quantization.

SMP also consistently improves the low-bit performance enabling more stable training as compared to other methods. This can be explained by the smooth, narrow-ranged representations across layers, enabling QLR to identify more precise low-bit representations. The empirical results are also in line with the quantization error upper bound for quantized SMP as proved in LEMMA 1. We also note that QLR can also improve EMP and APPNP to achieve reasonable accuracy with graceful degradation even in INT2 quantization. This highlights the importance of optimizing the quantization range for informative representation.

6.3.2 Performance with varying number of GNN layers. Figure 5 presents the performance of SMP and EMP with respect to different number of GNN layers, using FP, INT8, INT4, INT2, INT2-8 (BT), and INT2-8*(BT*) representations. For simplicity, we narrow the search space as lr ∈ {0.005, 0.008, 0.01, 0.015}, wd=$5e^{-4}$, lr_γ=$5e^{-4}$, wd_γ=$1e^{-5}$. We note that SMP-FP and SMP-INT8 outperform EMP-FP and EMP-INT8 in most cases with varying margins. The improvements can be further enhanced by tuning the parameters in a wider search space as listed in Section 6.1. We note that SMP-INT4 continuously outperforms EMP-INT4 in nearly all cases with different values of L (except L=2–4 on PubMed and L=2 on CS). SMP-INT2 achieves relatively high accuracy when compared with EMP-INT2, which underlines the benefits of smoothness constraint of SMP for extreme low-bit quantization.

6.3.3 Effect of Bitwise Truncation on GNN quantization. For INT2 quantization in Figure 5, the performance of BT (INT2-8) and BT* (INT2-8*) outperforms INT2 quantization with the basic QLR in most cases, while the accuracy of EMP-INT2-8* on CiteSeer is lower (around 0.1%–2.5%) than that of INT2 at L=6–12. Similar results are observed with PubMed at SMP-INT2-8* when L=6–8 with margins of 0.1%–1.3%. When we compare INT2-8 and INT2-8* under the same circumstances, the accuracy of INT2-8* is significantly larger than that of INT2-8 in most cases (except for CiteSeer, the performance of INT2-8* is slightly smaller than that of INT2-8 on SMP at L=6–8 and EMP at L=8). The performance of INT2-8* demonstrates the advantage over INT2-8 on the large datasets, i.e., PubMed and CS, and the accuracy of INT2-8* is close to that of INT4. Especially,

Table 4: Classification accuracy of Deep GNN methods (%) on benchmark datasets

	SMP				EMP				APPNP				PairNorm			DropEdge		
Dataset	FP	INT8	INT4	INT2	FP	INT8	INT4	INT2	FP	INT8	INT4	INT2	FP	INT8	INT4	FP	INT8	INT4
Cora	82.91 ±0.64	82.92 ±0.51	82.69 ±0.68	72.69 ±2.48	82.59 ±0.67	81.58 ±0.79	77.93 ±2.62	66.20 ±6.75	80.78 ±1.42	81.82 ±1.44	79.68 ±1.81	71.28 ±2.34	71.20 ±2.14	70.21 ±1.50	–	76.08 ±2.86	79.10 ±1.00	78.31 ±1.01
Citeseer	71.60 ±1.26	71.40 ±1.58	69.76 ±1.63	65.01 ±1.26	70.84 ±1.20	71.02 ±1.49	67.53 ±2.68	61.53 ±3.65	70.30 ±1.53	68.90 ±2.19	68.77 ±2.01	63.87 ±2.24	51.39 ±4.16	–	–	60.07 ±5.76	61.15 ±2.88	59.60 ±3.73
PubMed	79.36 ±2.15	79.90 ±2.41	78.44 ±2.51	75.92 ±2.29	78.64 ±2.93	78.48 ±3.32	76.18 ±3.44	74.22 ±3.03	79.59 ±1.39	79.31 1.98	78.49 ±2.37	73.41 ±1.19	75.72 ±2.21	OOM	OOM	75.56 ±1.81	OOM	OOM
CS	92.44 ±0.64	92.41 ±0.52	92.24 ±0.38	85.42 ± 1.80	92.17 ±0.46	91.16 ±0.42	91.81 ±0.54	82.62 ±1.09	91.58 ±0.66	92.33 ±0.57	91.94 ±0.57	81.25 ±8.08	75.12 ±4.24	OOM	OOM	87.30 ±1.64	OOM	OOM

– denotes accuracy ≤ 40.00%, OOM means 'out-of-memory'

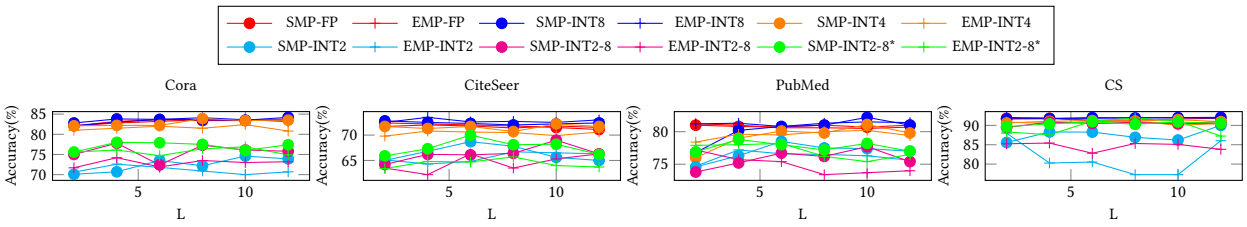


Figure 5: Results of SMP and EMP with varying layers

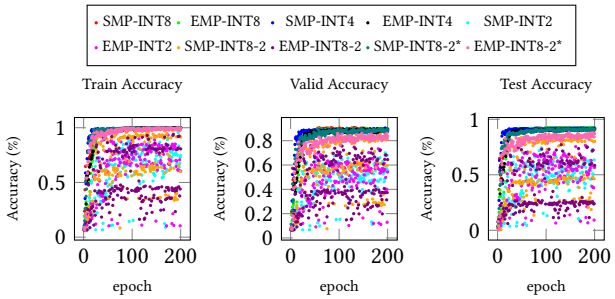


Figure 6: Quantization performance of SMP and EMP on CS

the accuracy of INT2-8* is, continuously, around 5%–7% higher than that of INT2-8 on CS dataset with respect to varying number of layers. This shows the effectiveness of BT* in reaching relatively high accuracy with low-bit representations.

Figure 6 presents the full training process of SMP and EMP with variations of our quantization approach on CS dataset. While SMP and EMP show unstable performance with INT2 representation generated by the basic quantizer, applying skewness-aware BT (BT*) contributes improvements that are close to that of INT8 quantization. Furthermore, SMP is more robust than EMP, due to the existence of quantization error bound in SMP.

6.4 Inference Speedup

In Table 5, we elucidate the inference times associated with varying quantization levels for the 2-layer GCN and SMP architectures, respectively. These model inferences are conducted on the Reddit dataset. The ↑ signifies the inference speedup comparing with FP model. To realize quantized GNN speed improvements across different quantization levels, we leverage the recent Tensor Core-based approach, QGCT [58], applied to both GCN and SMP. We observe a notable speedup of 5.11 × and 6.44 × with SMP and GCN, respectively, in the context of low-bit representation (INT2), in comparison to the FP counterparts. Notably, the speedup for SMP exhibits a slight reduction compared to GCN, attributed to the additional computational overhead of SMP. Remarkably, with the same number of layer (L), SMP showcases superior accuracy

Table 5: Inference time (ms) of SMP and GCN in different quantization levels on Reddit dataset

	FP	INT8	↑	INT4	↑	INT2	↑
SMP	178.3	46.46	3.84 ×	37.96	4.70 ×	34.89	5.11 ×
GCN	156.97	34.96	4.49 ×	27.29	5.75 ×	24.37	6.44 ×

performance relative to GCN. This is exemplified in Table 2 and Figure 5 in the CS dataset with L=2. Specifically, for SMP, the INT8 and INT4 accuracy outperforms GCN by approximately 2%, while SMP in INT2 mode demonstrates a performance advantage over GCN by up to 13.5%.

7 CONCLUSION

We have introduced an end-to-end solution towards achieving scalable deep GNNs, involving an efficient quantization with learnable ranges, with skewness-aware bitwise truncation, and a smoothness-aware message propagation (SMP) mechanism for efficient training and managing large deep GNNs. The solution reduces the model size and maintains its accuracy for classification even in low-bit representations. The message passing block in training is enforced to have layer-wise smoothness and constrains the changes between neighbor nodes. We formulate it as an additional constraint to a graph denoising optimization function and solved by Lagrange functions with an iterative BDMM algorithm. It aims to mitigate the oversmoothing problem in GNNs and to avoid the performance degradation encountered in low-bit quantization-aware training. We provide an upper bound on the error for the quantized SMP algorithm. Experiments show how the proposed solution achieves significant improvements over the-state-of-the-art approaches, providing a significant reduction in model sizes, an order of magnitude smaller than the full precision (FP) model with comparable accuracy results, and mitigating the oversmoothing problem on benchmark datasets.

ACKNOWLEDGMENTS

This work is supported in part by the UK Engineering and Physical Sciences Research Council under Grant No. EP/T51794X/1.

REFERENCES

[1] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. 2021. Binary graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9492–9501.

[2] Neil Band. 2020. MemFlow: Memory-Aware Distributed Deep Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2883–2885.

[3] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. 2018. Scalable Methods for 8-Bit Training of Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 5151–5159.

[4] Ron Banner, Yury Nahshan, and Daniel Soudry. 2019. *Post Training 4-Bit Quantization of Convolutional Networks for Rapid-Deployment*. Curran Associates Inc., Red Hook, NY, USA.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[6] S. Berchtold, C. Bohm, H.V. Jagadish, H.-P. Kriegel, and J. Sander. 2000. Independent quantization: an index compression technique for high-dimensional data spaces. In *Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073)*. 577–588. https://doi.org/10.1109/ICDE.2000.839456

[7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Feb. 2016). https://ojs.aaai.org/index.php/AAAI/article/view/10179

[8] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226* (2013).

[9] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Training deep neural networks with low precision multiplications. arXiv:1412.7024 [cs.LG]

[10] Gunduz Vehbi Demirci, Aparajita Haldar, and Hakan Ferhatosmanoglu. 2022. Scalable Graph Convolutional Network Training on Distributed-Memory Systems. *Proc. VLDB Endow.* 16, 4 (dec 2022), 711–724. https://doi.org/10.14778/3574245.3574256

[11] Mucong Ding, Kezhi Kong, Jingling Li, Chen Zhu, John Dickerson, Furong Huang, and Tom Goldstein. 2021. VQ-GNN: A universal framework to scale up graph neural networks using vector quantization. *Advances in Neural Information Processing Systems* 34 (2021), 6733–6746.

[12] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 293–302.

[13] Karima Echihabi. 2020. High-dimensional vector similarity search: from time series to deep network embeddings. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2829–2832.

[14] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. New Trends in High-D Vector Similarity Search: Al-Driven, Progressive, and Distributed. *Proc. VLDB Endow.* 14, 12 (oct 2021), 3198–3201. https://doi.org/10.14778/3476311.3476407

[15] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2019. LEARNED STEP SIZE QUANTIZATION. In *International Conference on Learning Representations*.

[16] Boyuan Feng, Yuke Wang, Xu Li, Shu Yang, Xueqiao Peng, and Yufei Ding. 2020. 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, 1044–1052.

[17] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. 2000. Vector approximation based indexing for non-uniform high dimensional data sets. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*. 202–209.

[18] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[19] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1gL-2A9Ym

[20] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*. Chapman and Hall/CRC, 291–326.

[21] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. 1263–1272.

[22] Richard A Groeneveld and Glen Meeden. 1984. Measuring skewness and kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician)* 33, 4 (1984), 391–399.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[24] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[26] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. arXiv:1506.05163 [cs.LG]

[27] Linyong Huang, Zhe Zhang, Zhaoyang Du, Shuangchen Li, Hongzhong Zheng, Yuan Xie, and Nianxiong Tan. 2022. EPQuant: A Graph Neural Network compression approach based on product quantization. *Neurocomputing* 503 (2022), 49–61.

[28] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *J. Mach. Learn. Res.* 18, 1 (jan 2017), 6869–6898.

[29] Benoit Jacob et al. 2017. gemmlowp: a small self-contained low-precision GEMM library.(2017).

[30] Vadim Kantorov. 2020. *Pack bool and other integer tensors into smaller bitwidth in PyTorch*. https://gist.github.com/vadimkantorov/30ea6d278bc492abf6ad328c6965613a

[31] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. arXiv:1611.07308 [stat.ML]

[32] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJU4ayYgl

[33] Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. Towards accurate binary convolutional neural network. *Advances in neural information processing systems* 30 (2017).

[34] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 338–348.

[35] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic graph neural networks. In *International Conference on Machine Learning*. PMLR, 6837–6849.

[36] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*. 1202–1211.

[37] Jayanta Mondal and Amol Deshpande. 2012. Managing large dynamic graphs efficiently. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 145–156.

[38] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 2014–2023.

[39] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. *8th International Conference on Learning Representations* (2019).

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[41] John C Platt and Alan H Barr. 1988. Constrained differential optimization for neural networks. (1988).

[42] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903* (2019).

[43] Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. 2022. Explaining link prediction systems based on knowledge graph embeddings. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*. 2062–2075.

[44] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. 2017. The ubiquity of large graphs and surprising challenges of graph processing. *Proceedings of the VLDB Endowment* 11, 4 (2017), 420–431.

[45] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. 2020. The ubiquity of large graphs and surprising challenges of graph processing: extended survey. *VLDB J.* 29, 2-3 (2020), 595–618. https://doi.org/10.1007/s00778-019-00548-x

[46] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[47] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[48] Bin Shao, Haixun Wang, and Yatao Li. 2013. Trinity: A distributed graph engine on a memory cloud. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 505–516.

[49] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[51] Shyam A. Tailor, Javier Fernandez-Marques, and Nicholas D. Lane. 2021. Degree-Quant: Quantization-Aware Training for Graph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=NSBrFgJAHg

[52] Ertem Tuncel, Hakan Ferhatosmanoglu, and Kenneth Rose. 2002. VQ-index: An index structure for similarity searching in multimedia databases. In *Proceedings of the tenth ACM international conference on Multimedia*. 543–552.

[53] Alina Vretinaris, Chuan Lei, Vasilis Efthymiou, Xiao Qin, and Fatma Özcan. 2021. Medical entity disambiguation using graph neural networks. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*. 2310–2318.

[54] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, Xiangjian He, Yiguang Lin, and Xuemin Lin. 2021. Binarized graph neural network. *World Wide Web* 24, 3 (2021), 825–848. https://doi.org/10.1007/s11280-021-00878-3

[55] Peiqi Wang, Xinfeng Xie, Lei Deng, Guoqi Li, Dongsheng Wang, and Yuan Xie. 2018. HitNet: Hybrid ternary recurrent neural network. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 602–612.

[56] Runhui Wang and Dong Deng. 2020. DeltaPQ: lossless product quantization code compression for high dimensional similarity search. *Proceedings of the VLDB Endowment* 13, 13 (2020), 3603–3616.

[57] Shuang Wang and Hakan Ferhatosmanoglu. 2020. PPQ-trajectory: spatio-temporal quantization for querying in large trajectory repositories. *Proceedings of the VLDB Endowment* 14, 2 (2020), 215–227.

[58] Yuke Wang, Boyuan Feng, and Yufei Ding. 2022. 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*. 107–119.

[59] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[60] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. 2021. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*. PMLR, 11875–11886.

[61] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*. 2487–2496.

[62] Dalong Zhang, Xin Huang, Ziqi Liu, Jun Zhou, Zhiyang Hu, Xianzheng Song, Zhibang Ge, Lin Wang, Zhiqiang Zhang, and Yuan Qi. 2020. AGL: A Scalable System for Industrial-purpose Graph Machine Learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3125–3137.

[63] Lingxiao Zhao and Leman Akoglu. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkecl1rtwB

[64] Yiren Zhao, Duo Wang, Daniel Bates, Robert Mullins, Mateja Jamnik, and Pietro Lio. 2020. Learned Low Precision Graph Neural Networks. *arXiv preprint arXiv:2009.09232* (2020).

[65] Chenguang Zheng, Hongzhi Chen, Yuxuan Cheng, Zhezheng Song, Yifan Wu, Changji Li, James Cheng, Hao Yang, and Shuai Zhang. 2022. ByteGNN: efficient graph neural network training at large scale. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1228–1242.

[66] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*. 1215–1226.

[67] Zeyu Zhu, Fanrong Li, Zitao Mo, Qinghao Hu, Gang Li, Zejian Liu, Xiaoyao Liang, and Jian Cheng. 2022. A^2Q:Aggregation-Aware Quantization for Graph Neural Networks. In *The Eleventh International Conference on Learning Representations*.