Outlier-Aware Post-Training Quantization for Discrete Graph Diffusion Models

Zheng Gong [1] **Ying Sun*** [1]

Abstract

Discrete Graph Diffusion Models (DGDMs) mark a pivotal advancement in graph generation, effectively preserving sparsity and structural integrity, thereby enhancing the learning of graph data distributions for diverse generative applications. Despite their potential, DGDMs are computationally intensive due to the numerous low-parameter yet high-computation operations, thereby increasing the need of inference acceleration. A promising solution to mitigate this issue is model quantization. However, existing quantization techniques for Image Diffusion Models (IDMs) face limitations in DGDMs due to differing diffusion processes, while Large Language Model (LLM) quantization focuses on reducing memory access latency of loading large parameters, unlike DGDMs, where inference bottlenecks are computations due to smaller model sizes. To fill this gap, we introduce Bit-DGDM, a post-training quantization framework for DGDMs which incorporates two novel ideas: (i) *sparse-dense activation quantization* sparsely modeling the activation outliers through adaptively selected, data-free thresholds in full-precision and quantizing the remaining to low-bit, and (ii) *ill-conditioned low-rank decomposition* decomposing the weights into low-rank component enable faster inference and an α-sparsity matrix that models outliers. Extensive experiments demonstrate that Bit-DGDM not only reducing the memory usage from the FP32 baseline by up to $2.8\times$ and achieve up to $2.5\times$ speedup, but also achieve comparable performance to ultra-low precision of up to 4-bit.

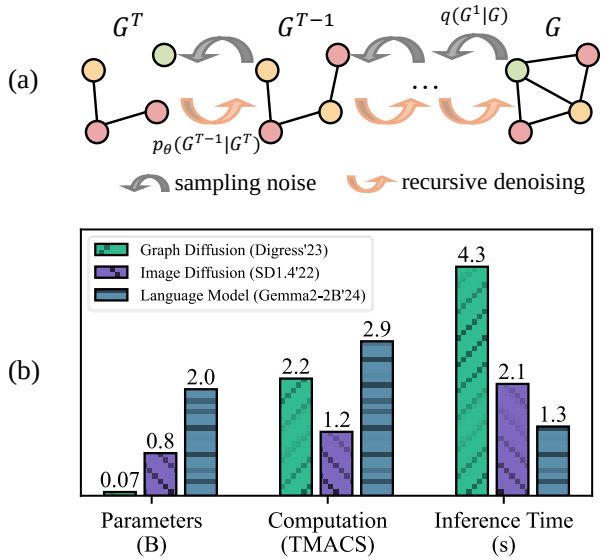

Figure 1. (a) An illustration of DGDM in the generation of a graph. (b) A comparison of the representative models widely used in DGDMs, IDMs, and LLMs, including Digress (Vignac et al., 2023), SD1.4 (Rombach et al., 2022) and Gemma2-2b (Team et al., 2024), in terms of parameter sizes, computation time, and inference time. The LLM is evaluated with 512 context tokens and 256 output tokens, while DGDM and IDM are assessed over 50 diffusion steps.

1. Introduction

Graph generation plays a pivotal role in graph learning (Li et al., 2024a; Liu et al., 2023c; Gong et al., 2023), with applications spanning drug design (Wu et al., 2018), code completion (Brockschmidt et al., 2018), and social network analysis (Newman, 2006). Recently, diffusion models (Rombach et al., 2022; Podell et al., 2024) have emerged as a cutting-edge and effective approach for generative tasks, with significant attention being directed toward graph diffusion models (Jo et al., 2022; Vignac et al., 2023). A notable development is the emergence of Discrete Graph Diffusion Models (DGDMs) (Yi et al., 2023; Liu et al., 2024), which employ discretized diffusion processes, as illustrated in Fig. 1(a). These approaches exhibit particular capability in preserving sparsity and structural integrity of the generated graphs. However, DGDMs face computational challenges due to their reliance on a recursive denoising process, thereby necessitating dedicated acceleration techniques.

[1]Artificial Intelligence Thrust, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. Correspondence to: Ying Sun <yings@hkust-gz.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

Quantization (Jacob et al., 2018; Wang et al., 2020) rises as a crucial direction for model acceleration, wherein floating-point numbers are converted into integers, thereby enhancing computational efficiency from a hardware perspective. However, existing quantization methods primarily focus on IDMs and LLMs. For IDMs, quantization approaches are centered on calibrating errors between the Gaussian denoising processes of full-precision and quantized models, such as time step-aware calibration (Li et al., 2023) and Gaussian noise correction (He et al., 2024b). Moreover, given huge model sizes, LLMs' inference latency is dominated by weight loading, prompting LLM-oriented quantization methods to emphasize memory access optimization through vector quantization (Egiazarian et al., 2024) and non-uniform quantization (Kim et al., 2024). However, existing quantization methods are inadequate to address the unique challenges of DGDMs:

(1) The inference efficiency of DGDMs is bounded by computations instead of memory access. The functioning of DGDMs relies on substantial low-parameter yet high-computation operations (*e.g.*, FiLM network (Perez et al., 2018)), which render DGDMs computationally intensive and result in longer runtime compared to basic Image Diffusion Models (IDMs) and Large Language Models (LLMs), as depicted in Fig. 1(b). However, due to the inherent differences between the discrete diffusion process in DGDMs and the Gaussian denoising process in IDMs, most quantization methods designed for IDMs face significant limitations when applied to DGDMs. Moreover, since DGDMs have substantially smaller parameter sizes compared to LLMs (0.07B versus 2B, as shown in Fig. 1(b)), the quantization methods designed to reduce the latency of LLM weight loading may be inefficient for the quantization of DGDMs.

(2) Considerable outliers in DGDMs are observed in both weights and activations. The outliers, referring to entries with magnitudes significantly larger than others, can substantially impact quantization precision by amplifying magnitudes and compressing the distinctions of remaining values. In text and image tasks, where outliers primarily occur in activations rather than weights, existing quantization methods like smoothing (Li et al., 2024b; Xiao et al., 2023) attempt to transfer these outliers from activations to weights. However, in graph generation tasks, outliers are significantly present in both activations and weights (Sec. 4.1), rendering both aspects highly susceptible to outlier effects. This dual vulnerability diminishes the effectiveness of smoothing methods in mitigating outliers (Sec. 5.2). Consequently, selecting an appropriate quantization approach to mitigate the influence of outliers in both activations and weights is crucial for maintaining the performance of quantized DGDMs.

In this paper, we propose a post-training quantization framework for DGDMs, called Bit-DGDM. Grounded in the insight that *the inference efficiency of DGDMs is primarily constrained by computation* and the observation that *outliers are prominently present in both weights and activations*, Bit-DGDM introduces a novel approach combining sparse-dense activation quantization with ill-conditioned low-rank weight decomposition to effectively mitigates the impact of outliers. Moreover, we implement equidistributed & adaptive sparse-dense kernels tailored for the decomposed matrix multiplication. These techniques achieve high efficiency at precisions as low as 4 bits, minimizing memory usage and speeding up inference without sacrificing model performance or the quality of generated graphs. *To the best of our knowledge, we are among the first to propose a quantization framework for DGDMs.*

The detailed contributions of our work are as follows:

- **Sparse-Dense Activation Quantization:** Recognizing that activations in DGDMs contain significant outliers, our sparse-dense activation quantization method utilizes a sparse matrix modeling outliers and a dense matrix modeling remaining values based on adaptively selected, data-free thresholds (Sec. 4.2).
- **Ill-Conditioned Low-Rank Weight Decomposition:** To address the outliers in weights and enhance computational efficiency, we decompose the weights into low-rank components and an α-sparsity matrix that models outliers, ensuring guaranteed recovery (Sec. 4.3).
- **Equidistributed & Adaptive Sparse-Dense Kernels:** To optimize inference speed and precision, we quantize the dense part of the product of decomposed weights and activations into low-bit representations and design equidistributed, adaptive sparse-dense kernels for efficient sparse-dense matrix multiplications (Sec. 4.4).
- **Evaluation:** We conduct extensively experiments across multiple DGDMs and diverse datasets to demonstrate that Bit-DGDM can effectively quantize DGDMs and obtain the superior performance compared with other SOTA quantization baselines. Notably, Bit-DGDM achieves $2.5\times$ inference speed while reducing the memory footprint by up to $2.8\times$ (Sec. 5.2 and Sec. 5.3).

2. Related Work

The related work of this paper falls in two categories: Graph Generation Models and Model Quantization. Detailed recent advancements of **Quantization for LLMs** and **Outlier-Aware Quantization** can be found in Appendix A.

2.1. Graph Generative Models

With the widespread prevalence of graph data (Sun et al., 2024; Ji et al., 2025; Gong & Sun, 2024b;a; Huang et al., 2022; Shi et al., 2024; Han et al., 2024), graph generation has found extensive applications, including inverse protein folding (Yi et al., 2023), molecule generation (Wu et al.,

2018), and program synthesis (Brockschmidt et al., 2018). Early on, there was considerable interest in graph generation using RNNs (You et al., 2018), VAEs (Simonovsky & Komodakis, 2018), GANs (De Cao & Kipf, 2018) or normalizing flows (Zang & Wang, 2020). Inspired by the tremendous success of diffusion models in the image domain, Niu et al. (2020) relied on Gaussian noise and generated adjacency matrices by thresholding continuous values to represent edges, while Jo et al. (2022) extended this approach to incorporate both node and edge attributes. Given the discrete nature of graph structures, Haefeli et al. (2022) were the first to design a discrete diffusion model specifically for unattributed graphs, demonstrating that discrete diffusion can enhance graph generation. Furthermore, Vignac et al. (2023) introduced the first Markovian noise model, Digress, tailored for attributed graphs, and Yi et al. (2023) extended Digress to inverse protein folding task where the node features are represented as 3D coordinates. In our study, we focus on the quantization of discrete graph diffusion models, as they inherently capture discrete structures and demonstrate greater efficacy in graph generation.

2.2. Model Quantization

Model quantization aims to save memory costs and speed up computations by reducing the precision of weights and activations. It can be generally divided into two categories. *Quantization-aware training* (QAT) (Jacob et al., 2018; Zhuang et al., 2018) integrates quantization during the training process, enabling high performance at lower precision. However, it necessitates significant time and computational resources. In contrast, *post-training quantization* (PTQ) (Wei et al., 2022a; Lin et al., 2022) does not require fine-tuning and only requires a small amount of unlabeled data for calibration, offering a faster and less resource-intensive alternative. Due to the huge computation costs of diffusion models, recent quantization approaches (He et al., 2024b;a; Li et al., 2024b) for diffusion models focused on PTQ. These approaches have pushed the boundaries of PTQ to 4-bit quantization by leveraging new rounding strategies (Nagel et al., 2020), layer-wise calibration (Wang et al., 2020), second-order statistics (Li et al., 2021), and mixed-precision schemes (He et al., 2024b). However, these methods primarily focus on continuous diffusion models, particularly in the image domain. To the best of our knowledge, there is a lack of methods addressing PTQ for DGDMs.

3. Preliminary

In this section, we first revisit discrete graph diffusion models and then introduce the quantization process. Following Vignac et al. (2023), we focus on graphs with categorical node and edge attributes, particularly well-suited for diverse structured data like chemical compounds (Wu et al., 2018) and proteins (Ingraham et al., 2019). Let $\mathcal{X}$ and $\mathcal{E}$ denote the categorical space of nodes and edges with cardinalities $|\mathcal{X}|$ and $|\mathcal{E}|$, respectively. For a graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ with $|\mathcal{V}|$ nodes, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{X}|}$ is the matrix of node categorical attributes, and each row $\mathbf{x}_i \in \mathbb{R}^{|\mathcal{X}|}$ denote the one-hot encoding of node i attributes. $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{E}|}$ represents the one-hot encoding $\mathbf{e}_{ij} \in \mathbb{R}^{|\mathcal{E}|}$ of each edge, treating the absence of edge as a particular edge type. Here, we take the 2D graph generation problem as an example, while the input for the other application of graph generation will be discussed in detail in Appendix C.

3.1. Discrete Graph Diffusion Models

Discrete graph diffusion models consist of two main component: a noise process and a recursive denoising process.

Noise process. The noise process q progressively corrupts graph $\mathcal{G}$ and creates a sequence of increasingly noisy graphs $(\mathcal{G}^1, ..., \mathcal{G}^T)$, where $\mathcal{G}^t = (\mathbf{X}^t, \mathbf{E}^t)$. It has a Markovian structure, where $q(\mathcal{G}^1, ..., \mathcal{G}^T|\mathcal{G}) = q(\mathcal{G}^1|\mathcal{G}) \prod_{t=2}^{T} q(\mathcal{G}^t|\mathcal{G}^{t-1})$. In line with Vignac et al. (2023), we add noises by sampling each node and edge type from a categorical distribution as

$$q(\mathcal{G}^t|\mathcal{G}^{t-1}) = (\mathbf{X}^{t-1}\mathbf{N}_X^t, \mathbf{E}^{t-1}\mathbf{N}_E^t), \tag{1}$$

where the transition matrix $\mathbf{N}_X^t = \alpha^t\mathbf{I} + \beta^t\mathbf{1}_a\mathbf{m}_X'$, and $\mathbf{N}_E^t = \alpha^t\mathbf{I} + \beta^t\mathbf{1}_a\mathbf{m}_E'$. $\mathbf{m}_X'$ and $\mathbf{m}_E'$ denote the true categorical distributions of node and edge attributes as the prior distributions, respectively. Since $(\mathbf{1}_a\mathbf{m}_X')^2 = \mathbf{1}_a\mathbf{m}_X'$, we have the closed-form $\overline{\mathbf{N}}_X^t = \mathbf{N}_X^1\mathbf{N}_X^2...\mathbf{N}_X^t = \overline{\alpha}^t\mathbf{I} + \overline{\beta}^t\mathbf{1}_a\mathbf{m}_X'$, where $\overline{\alpha}^t = \prod_{\tau=1}^{t}\alpha^\tau$ and $\overline{\beta}^t = 1 - \overline{\alpha}^t$, satisfying $\forall i, \lim_{T\to\infty}\overline{\mathbf{N}}_X^T\mathbb{1}_i = \mathbf{m}_X'$. Thus, it then follows $q(\mathbf{X}^t|\mathbf{X}^0) = \mathrm{Cat}(\mathbf{X}^t; \mathbf{X}\overline{\mathbf{N}}_X^t)$ and

$$q(\mathbf{X}^{t-1}|\mathbf{X}^t, \mathbf{X}^0) = \mathrm{Cat}(\mathbf{X}^{t-1}; \frac{\mathbf{X}^t\mathbf{N}_X^{t^\top} \odot \mathbf{X}^0\overline{\mathbf{N}}_X^{t-1}}{\mathbf{X}^0\overline{\mathbf{N}}_X^t\mathbf{X}^{t^\top}}), \tag{2}$$

where Cat denotes categorical distribution. The edge transition distribution is defined similarly and omitted for brevity.

Recursive denoising process. The denoising process $p_\theta(\mathcal{G}^{0:T}) = p(\mathcal{G}^T)\prod_{t=1}^{T} p_\theta(\mathcal{G}^{t-1}|\mathcal{G}^t)$ parameterized by θ aims to recover the distribution $p(\mathcal{G}^0)$ (for brevity, $\mathcal{G}^0 := \mathcal{G}$). The reversed transition $p_\theta(\mathcal{G}^{t-1}|\mathcal{G}^t)$ is a product of categorical distributions over nodes and edges, *i.e.,* $p_\theta(\mathbf{X}^{t-1}|\mathcal{G}^t)$ and $p_\theta(\mathbf{E}^{t-1}|\mathcal{G}^t)$. Aligned with the x_0-parameterization used in continuous diffusion models (Karras et al., 2022; Ho et al., 2020), $p_\theta(\mathbf{X}^{t-1}|\mathcal{G}^t)$ is modeled as:

$$p_\theta(\mathbf{X}^{t-1}|\mathcal{G}^t) \triangleq \sum_{\widetilde{\mathbf{X}}^0\in\mathcal{X}} q\left(\mathbf{X}^{t-1} \mid \mathbf{X}^t, \widetilde{\mathbf{X}}^0\right) p_\theta\left(\widetilde{\mathbf{X}}^0 \mid \mathcal{G}^t\right), \tag{3}$$

where $p_\theta(\widetilde{\mathbf{X}}^0 \mid \mathcal{G}^t)$ is a neural network predicting posterior probability of $\mathbf{X}^0$ given a noisy graph $\mathcal{G}^t$. The architectures of neural networks will be illustrated extensively in Appendix C. The reversed process for edges is similar to the nodes and thus omitted. After training, the new graph samples can be generated by sampling $\mathcal{G}_T$ from prior distributions $\mathbf{m}_X'$ and $\mathbf{m}_E'$ and subsequently sampling $p_\theta(\mathcal{G}^{t-1}|\mathcal{G}^t)$, resulting in a generation trajectory $(\mathcal{G}^T, \mathcal{G}^{T-1}, ..., \mathcal{G}^0)$.

Remark 3.1 (**DGDMs versus IDMs**). In DGDMs, both the noise process and the recursive denoising process model the node attributes and graph structures through discrete sampling from categorical distributions. In contrast, IDMs rely on Gaussian noise and denoising processes, where each step is represented as continuous values. This difference renders many IDM-oriented quantization methods, such as Gaussian noise correction (He et al., 2024b; Li et al., 2023), difficult to apply to DGDMs.

3.2. Quantization

Quantization is an effective approach to reduce model size and accelerate computations. Given a tensor $\mathbf{X}$, the quantization process is defined as:

$$\mathbf{X}_{\mathcal{Q}} = \text{clamp}(\left\lfloor \frac{\mathbf{X}}{s_{\mathbf{X}}} \right\rceil + z_{\mathbf{X}}, 0, 2^{\mathcal{B}} - 1), \tag{4}$$

where $\mathbf{X}_{\mathcal{Q}}$ is the low-bit representations of $\mathbf{X}$, $s_{\mathbf{X}} = \frac{\max(\mathbf{X}) - \min(\mathbf{X})}{2^{\mathcal{B}} - 1}$ is the scaling factor, $z_{\mathbf{X}} = -\left\lfloor \frac{\min(\mathbf{X})}{s_{\mathbf{X}}} \right\rceil$ is the zero point, and $2^{\mathcal{B}} - 1$ denotes the maximum quantized value with respect to the bit width b. The dequantized tensor is formulated as:

$$\hat{\mathbf{X}} = \mathcal{Q}^{-1}(\mathcal{Q}(\mathbf{X})) = s_{\mathbf{X}} \cdot (\mathbf{X}_{\mathcal{Q}} - z_{\mathbf{X}}). \tag{5}$$

Due to the high computational costs of diffusion model training, we focus on post-training quantizing the parameters of well-trained discrete graph diffusion models, *i.e.,* θ in Eqn. (3), to reduce memory requirements and enhance the efficiency of graph generation, e.s.p.molecule synthesis (Wu et al., 2018) and protein folding (Ingraham et al., 2019).

4. Bit-DGDM: Outlier-Aware PTQ for DGDM

In this section, we first discuss the impact of outliers in the quantization of DGDMs in Sec. 4.1. Then we introduce the our post-training quantization framework Bit-DGDM comprising sparse-dense activation quantization and ill-conditioned low-rank weight decomposition, detailed in Sec. 4.2 and Sec. 4.3, respectively. At last, we introduce our kernel implementation for efficiently inference on the quantized DGDMs in Sec. 4.4. The overall architecture and algorithm of our framework are shown in Fig. 3 and Alg. 1.

4.1. Outlier Analysis of DGDMs

The quantization process based on Eqn. (4) and (5) reveals that the presence of outliers significantly amplifies the scale factor by increasing the maximum value and decreasing the minimum value. Since adjacent ranges of the scale factor are quantized into neighboring integers, a larger scale factor implies that a wider range of floating-point numbers is mapped to the same integer, leading to greater quantization errors and ultimately degrading quantization precision. In the left panels of Fig. 2, we visualize the distributions of activations and weights in one layer of DGDMs, along with the top and bottom 0.1% and 0.5% quantiles marked as outliers. It is evident that these outliers substantially increase the magnitude. In the right of Fig. 2, kurtosis (κ) is employed as a statistical measure, defined as $\kappa\left(\{x_i\}_{i=1}^n\right) = \frac{n\sum_{i=1}^n (x_i - \bar{x})^4}{\left(\left(\sum_{i=1}^n (x_i - \bar{x})^2\right)^2\right)}$. A higher kurtosis value indicates heavier tails, while a lower kurtosis corresponds to a smoother distribution (Westfall, 2014). Removing a small number of the most significant outliers is shown to significantly reduce kurtosis, thereby improving quantization precision.

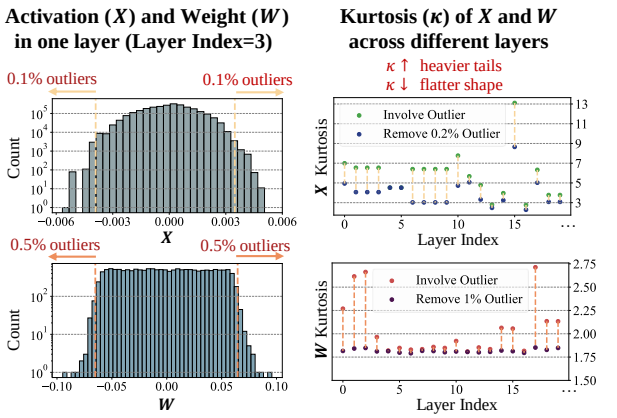

Figure 2. The left panels illustrate the numerical distributions of activations and weights for the layer with index 3. On the right, the top and bottom panels show how removing the largest and smallest 0.1% and 0.5% of outliers significantly reduces the kurtosis of the numerical distributions. A lower kurtosis corresponds to a flatter distribution, which facilitates quantization.

4.2. Sparse-Dense Activation Quantization

The analysis of DGDMs in Sec. 4.1 indicates a significant presence of outliers in the activations. Additionally, we observe that the volume of activations is substantially larger than that of the weights (*e.g.*, 10^5 versus 10^3 for a single layer, as shown in Fig. 2), highlighting the computational intensity of DGDMs. Therefore, it is imperative to propose an efficient quantization method for activations that not only ensures quantization precision but also mitigates the impact of outliers. With this in mind, we introduce a method to filter out outliers and quantize the activations. We first perform a simple yet effective decomposition of activation $\mathbf{X} \in \mathbb{R}^{b \times m}$ into a sparse matrix $\mathbf{S}_X$ containing the outliers and a remaining dense matrix $\mathbf{D}_X$:

$$\mathbf{X} = \mathbf{D}_X + \mathbf{S}_X. \tag{6}$$

$\mathbf{D}_X$ can be effectively quantized into low-bit representation $\hat{\mathbf{D}}_X$ due to its significant reduced value ranges. The outliers contained in sparse $\mathbf{S}_X$ are filtered from $\mathbf{X}$ via $\mathbf{S}_X = \mathbf{X}[x > \tau_{\text{max}} | x < \tau_{\text{min}}]$, where the thresholds $\tau_{\text{max/min}}$ are selected based on the percentile of the value distribution.

Data-free Adaptive Threshold Selection. The selection of thresholds $\tau_{\text{max/min}}$ typically requires a substantial num-

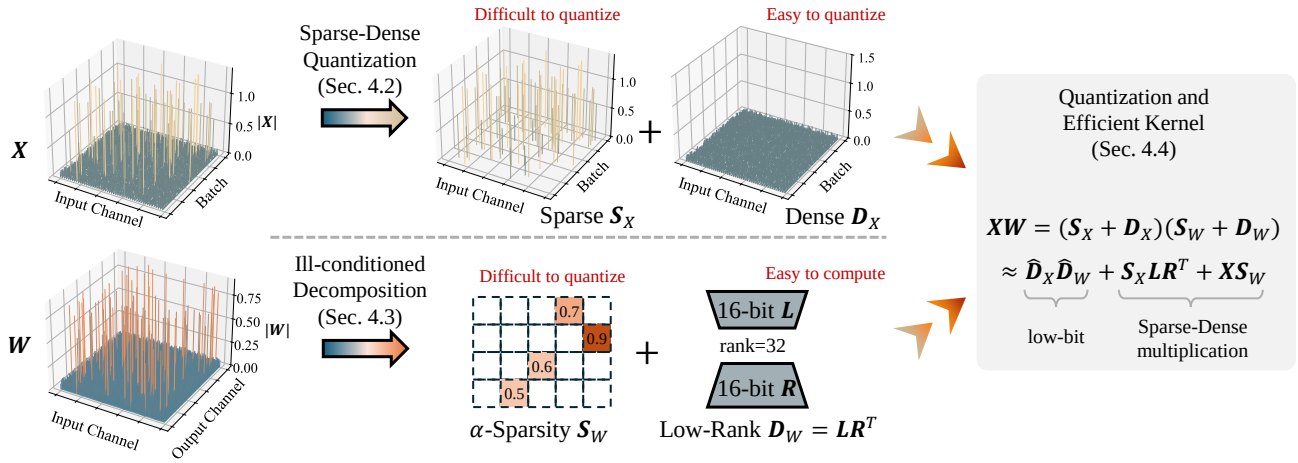

Figure 3. Our quantization framework Bit-DGDM consists of three phases. (i) Decomposing the activations into a sparse matrix modeling outliers and a dense quantized matrix modeling remaining values (Sec. 4.2). (ii) Utilizing the ill-conditioned low-rank decomposition method for decomposing weights into an α-sparsity matrix and a low-rank components (Sec. 4.3). (iii) Quantizing the dense parts to low-bit representations and implementing efficient kernel for sparse-dense matrix multiplication (Sec. 4.4).

ber of activation values to ensure accurate percentile estimation. A straightforward approach involves inferring activation values using a training dataset. However, this necessitates access to a large raw dataset. Acquiring such datasets is often challenging due to their size, privacy concerns, and copyright restrictions. Moreover, avoiding the use of raw datasets enhances the applicability of post-training quantization. With this in mind, we propose a data-free method for adaptive threshold selection. Our approach begins by sampling multiple initial graphs $\{\mathcal{G}^T\} = \{(\mathbf{X}^T, \mathbf{E}^T)\}$ from uniform distributions of node and edge types:

$$\mathbf{X}^T \sim \mathcal{U}(\{1, ..., |\mathcal{X}|\}),\ \mathbf{E}^T \sim \mathcal{U}(\{1, ..., |\mathcal{E}|\}). \tag{7}$$

These graphs are processed through the full-precision DGDM to obtain intermediate activation values in each steps. Thresholds are then selected based on the activations from all steps within the same layer, using predefined percentiles, thereby facilitating effective quantization without relying on raw datasets.

4.3. Ill-conditioned Low-Rank Weight Decomposition

Our analysis of outliers in activations and weights indicates that, unlike in LLMs where outliers predominantly occur in activations, DGDMs exhibit significant outliers in model weights as well. This characteristic may render quantization methods via smoothing inefficient (Xiao et al., 2023), as reducing the magnitude of activations comes at the cost of amplifying the magnitude of weights, thereby exacerbating the impact of outliers on weight quantization. Moreover, quantization approaches that mitigate outlier effects through SVD decomposition (Li et al., 2024b) may also be compromised, as outliers can substantially influence the magnitude of singular values and the direction of singular vectors. Inspired by robust principal component analysis (RPCA) (Candès et al., 2011) for enhancing the dimension reduction robustness for the outlier-existing matrix, we propose an ill-conditioned low-rank weight decomposition method. The overall algorithm for weight decomposition is illustrated in Alg. 2. We assume that the weight $\mathbf{W} \in \mathbb{R}^{m\times n}$ is a superposition of $\mathbf{D}^\star_W$ and $\mathbf{S}^\star_W$:

$$\mathbf{W} = \mathbf{D}^\star_W + \mathbf{S}^\star_W, \tag{8}$$

where $\mathbf{S}^\star_W$ is a sparse matrix modeling the outliers of weight $\mathbf{W}$, and $\mathbf{D}^\star_W$ is a rank-r matrix modeling the remaining component. Herein, $*$ denote the ideal or target values, while the following variables without $*$ denote the variables to be optimized. We define the sparsity of $\mathbf{S}_W$ following α-sparsity constraint (Yi et al., 2016; Netrapalli et al., 2014):

Definition 4.1 (α-sparsity)**.** The sparse matrix $\mathbf{S}$ is α-sparsity if at most α-fraction of nonzero entries per row and column for $\alpha \in [0, 1)$, which we denote as:

$$\mathcal{S}_\alpha := \{\mathbf{S} \in \mathbb{R}^{m\times n} | \forall i, j,\ \ \|\mathbf{S}_{i,:}\|_0 \le \alpha n, \|\mathbf{S}_{:,j}\|_0 \le \alpha m\}.$$

To avoid the low-rank constraint on $\mathbf{D}_W$, we write $\mathbf{D}_W$ as product of $\mathbf{D}_W = \mathbf{L}\mathbf{R}^\top$ with $\mathbf{L} \in \mathbb{R}^{m\times r}$ and $\mathbf{R} \in \mathbb{R}^{n\times r}$. We utilize the low-rank matrix $\mathbf{L}\mathbf{R}^\top$ and the sparse matrix $\mathbf{S}_W$ to reconstruct the raw weight matrix $\mathbf{W}$ through the following optimization objective:

$$\min_{\mathbf{L}\in\mathbb{R}^{m\times r}, \mathbf{R}\in\mathbb{R}^{n\times r}, \mathbf{S}_W\in\mathcal{S}_\alpha} \|\mathbf{L}\mathbf{R}^\top + \mathbf{S}_W - \mathbf{W}\|_F^2, \tag{9}$$

where $\|\cdot\|_F$ denotes Frobenius Norm. To enable gradient descent on efficiently optimizing $\mathbf{L}$ and $\mathbf{R}$, we denote the loss function in k-th iteration $\mathcal{L}_k := \mathcal{L}(\mathbf{L}_k, \mathbf{R}_k) := \|\mathbf{L}_k\mathbf{R}_k^\top + \mathbf{S}_{W,k} - \mathbf{W}\|_F^2$. To enforce the α-sparsity constraint of $\mathbf{S}_W$ in Def. 4.1, we employs a sparsification oper-

ator $\mathcal{T}_\alpha(\cdot)$ for any matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$:

$$[\mathcal{T}_\alpha(\mathbf{M})]_{i,j} = \begin{cases} [\mathbf{M}]_{i,j}, & \text{if } |[\mathbf{M}]_{i,j}| \geq |[\mathbf{M}]_{i,:}^{(\alpha n)}| \\ & \text{and } |[\mathbf{M}]_{i,j}| \geq |[\mathbf{M}]_{:,j}^{(\alpha m)}|, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $[\cdot]_{i,:}^{(a)}$ and $[\cdot]_{:,j}^{(a)}$ denote the entry with a-th largest magnitude in the i-th row and j-th column, respectively. In the $(k+1)$-th optimization iteration, the sparse matrix $\mathbf{S}_W$ is updated as:

$$\mathbf{S}_{W,k+1} = \mathcal{T}_\alpha(\mathbf{W} - \mathbf{L}_k\mathbf{R}_k^\top). \tag{11}$$

After updating $\mathbf{S}_{W,k+1}$, the gradients w.r.p $\mathbf{L}_k$ and $\mathbf{R}_k$ can be exactly computed as:

$$\begin{aligned} \nabla_{\mathbf{L}_k}\mathcal{L}_k &= \left(\mathbf{L}_k\mathbf{R}_k^\top + \mathbf{S}_{W,k+1} - \mathbf{W}\right)\mathbf{R}_k, \\ \nabla_{\mathbf{R}_k}\mathcal{L}_k &= \left(\mathbf{L}_k\mathbf{R}_k^\top + \mathbf{S}_{W,k+1} - \mathbf{W}\right)\mathbf{L}_k. \end{aligned} \tag{12}$$

However, the vanilla gradient descent optimization suffers from the ill-conditioned matrix, and thus we incorporate the scaled terms $\left(\mathbf{R}_k^\top\mathbf{R}_k\right)^{-1}$ and $\left(\mathbf{L}_k^\top\mathbf{L}_k\right)^{-1}$ to overcome the impacts of outliers following (Tong et al., 2021). We update the low-rank components:

$$\begin{aligned} \mathbf{L}_{k+1} &= \mathbf{L}_k - \eta_{k+1}\nabla_{\mathbf{L}_k}\mathcal{L}_k \cdot \left(\mathbf{R}_k^\top\mathbf{R}_k\right)^{-1}, \\ \mathbf{R}_{k+1} &= \mathbf{R}_k - \eta_{k+1}\nabla_{\mathbf{R}_k}\mathcal{L}_k \cdot \left(\mathbf{L}_k^\top\mathbf{L}_k\right)^{-1}, \end{aligned} \tag{13}$$

where η_{k+1} denote the step size at the $(k+1)$-th iteration.

Initialization. We introduce an initialization of low-rank components $\mathbf{L}_0$ and $\mathbf{R}_0$ as well as the sparse matrix $\mathbf{S}_{W,0}$ for reducing the effects of outliers in the weight decomposition. We first assign the sparse matrix $\mathbf{S}_{W,0} = \mathcal{T}_\alpha(\mathbf{W})$ to remove the obvious outliers with largest magnitude. Then we initialize $\mathbf{L}_0 = \mathbf{U}_0\mathbf{\Sigma}_0^{\frac{1}{2}}$, $\mathbf{R}_0 = \mathbf{V}_0^\top\mathbf{\Sigma}_0^{\frac{1}{2}}$, where $\mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0$ is the best rank-r approximation of $\mathbf{W} - \mathbf{S}_{W,0}$.

Theoretical Guarantee. We present the recovery guarantee of our proposed ill-conditioned low-rank weight decomposition method here. We start with the μ-incoherence assumption of $\mathbf{D}_W^\star$ following RPCA works (Candès et al., 2011; Chen, 2015):

Assumption 4.2 (μ-incoherence)**.** $\mathbf{D}_W^\star \in \mathbb{R}^{m\times n}$ is a rank-r matrix with μ-incoherence, *i.e.*,

$$\|\mathbf{U}_\star\|_{2,\infty} \leq \sqrt{\frac{\mu r}{m}} \quad \text{and} \quad \|\mathbf{V}_\star\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}}$$

for $1 \leq \mu \leq \max(m,n)$, where $\mathbf{U}_\star\mathbf{\Sigma}_\star\mathbf{V}_\star^\top$ is the compact SVD of $\mathbf{D}_W^\star$, and $\|\mathbf{M}\|_{2,\infty} = \max_i(\sum_j \mathbf{M}_{i,j}^2)^{\frac{1}{2}}$.

Herein, $\mathbf{U}_\star \in \mathbb{R}^{m\times r}$ and $\mathbf{V}_\star \in \mathbb{R}^{n\times r}$ are composed of r left and right singular vectors, respectively. $\mathbf{\Sigma}_\star \in \mathbb{R}^{r\times r}$ is a diagonal matrix consisting of r singular values of $\mathbf{D}_W^\star$ organized in a non-increasing order, *i.e.*, $\sigma_1(\mathbf{D}_W^\star) \geq ... \geq \sigma_r(\mathbf{D}_W^\star) > 0$. And we define $\kappa := \sigma_1(\mathbf{D}_W^\star)/\sigma_r(\mathbf{D}_W^\star)$.

Assumption 4.3 (α-Sparsity of Outliers)**.** Let $\mathbf{S}_W \in \mathbb{R}^{m\times n}$ be an α-sparsity matrix representing the outliers. Specifically, we impose the condition $\alpha \lesssim \mathcal{O}(1/(\mu r^{3/2}\kappa))$ to ensure guaranteed recovery.

Remark 4.4. Our assumption aligns with the most common RPCA models, which address ill-conditioned low-rank compositions under a sparsity assumption. Notably, this assumption is based on the observation that the number of outliers in the weights is inherently limited, whereas their magnitude has a more significant impact on quantization.

Based on the rank-sparsity uncertainty principle, a matrix cannot simultaneously exhibit incoherence and sparsity (Chandrasekaran et al., 2011). The two assumptions outlined above ensure the uniqueness of our weight decomposition solution. We now present our main theorem:

Theorem 4.5 (Guaranteed Recovery)**.** *Let $\mathbf{D}_W^\star$ be a rank-r matrix with μ-incoherence, and let $\mathbf{S}_W^\star$ be an α-sparsity matrix with $\alpha \lesssim \mathcal{O}(1/(\mu r^{3/2}\kappa))$. The iterative optimization at step k satisfies*

$$\begin{aligned} |\mathbf{L}_k\mathbf{R}_k^\top - \mathbf{D}_W^\star|_F &\leq 0.03(1-0.6\eta)^k\sigma_r(\mathbf{D}_W^\star), \\ \mathit{dist}(\mathbf{L}_k, \mathbf{R}_k; \mathbf{L}^\star, \mathbf{R}^\star) &\leq 0.02(1-0.6\eta)^k\sigma_r(\mathbf{D}_W^\star), \end{aligned}$$

where the step size $\eta \in [0.1, 2/3]$.

Proof. The proof is detailed in Appendix B. □

4.4. Quantization and Kernel Implementation

To efficiently handle matrix multiplication between activations and weights in the presence of outliers, two conditions must be satisfied: (1) outliers should be computed with high precision, and (2) the quantized matrices should be computed using the same bit-widths. With these considerations, we proceed as follows: (i) We first decompose the activations and weights using Eqn. (6) and (8). (ii) We then perform quantization on the dense components $\mathbf{D}_X$ and $\mathbf{D}_W$ to the same bit-width, ensuring the removal of outliers. For the multiplication involving activation outliers $\mathbf{S}_X$, we decompose $\mathbf{D}_W$ into a low-rank approximation $\mathbf{LR}^\top$ to maintain precision and efficiency. (iii) Finally, we compute the product of the sparse weight outliers $\mathbf{S}_W$ with the raw activation $\mathbf{X}$. This process is formulated as:

$$\begin{aligned} \mathbf{XW} &= (\mathbf{D}_X + \mathbf{S}_X)(\mathbf{D}_W + \mathbf{S}_W) \\ &\overset{(i)}{=} \mathbf{D}_X\mathbf{D}_W + \mathbf{S}_X\mathbf{D}_W + \mathbf{D}_X\mathbf{S}_W + \mathbf{S}_X\mathbf{S}_W \\ &\overset{(ii)}{\approx} \hat{\mathbf{D}}_X\hat{\mathbf{D}}_W + \mathbf{S}_X\mathbf{LR}^\top + \mathbf{D}_X\mathbf{S}_W + \mathbf{S}_X\mathbf{S}_W \\ &\overset{(iii)}{=} \underbrace{\hat{\mathbf{D}}_X\hat{\mathbf{D}}_W}_{\text{low-bit}} + \underbrace{\mathbf{S}_X\mathbf{LR}^\top + \mathbf{XS}_W}_{\text{Sparse-Dense Multiplication}}. \end{aligned} \tag{14}$$

Equidistributed and Adaptive Sparse-Dense Kernel. To optimize the computational efficiency of step (iii) in Eqn. (14), the sparse-dense matrix multiplication plays a

Table 1. Molecule generation performance of full-precision Digress (Vignac et al., 2023) (*i.e.,* baseline) and quantized Digress with different quantization methods on QM9 and MOSES datasets. "WxAy" denotes that the weights (W) and activations (A) are represented in x-bit and y-bit precision, respectively. "Mem." refers to the peak memory usage during the model inference.

Molecule Generation		QM9				MOSES				
Method	**Precision**	**Valid** (%)	**Unique** (%)	**Speedup** (↑)	**Mem.** (GB ↓)	**Novel** (%)	**Valid** (%)	**Unique** (%)	**Speedup** (↑)	**Mem.** (GB ↓)
baseline	FP32	99.0	96.2	1×	6.4	93.8	82.5	99.8	1×	8.7
	BF16	98.8	96.0	1.3×	3.4	93.6	82.5	99.7	1.3×	4.6
RTN	W4A4	57.6	88.6	2.7×	1.6	84.2	63.8	87.9	2.8×	2.3
GPTQ	W4A4	72.4	86.1	2.7×	1.6	85.1	72.0	92.6	2.8×	2.3
SVDQuant	W4A4	95.6	92.7	2.2×	2.0	91.6	80.6	96.5	2.2×	3.0
DuQuant	W4A4	96.5	94.6	2.1×	2.1	92.8	81.1	98.0	2.1×	3.2
SqueezeLLM	W4A16	95.9	94.8	1.1×	2.3	92.5	81.3	98.0	1.1×	3.6
Bit-DGDM	W4A4	98.2	95.5	2.5×	2.3	93.1	81.9	98.9	2.5×	3.8

critical role. The term $\mathbf{X}\mathbf{S}_W$, where $\mathbf{S}_W$ adheres to α-Sparsity, implies that each row and column contains at most an α-fraction of nonzero entries. In other words, the nonzero elements are evenly distributed across each row and column. To address this, we design a *equidistributed sparse-dense kernel* that assigns each thread a column, ensuring uniform workload distribution across threads. For the term $\mathbf{S}_X\mathbf{L}\mathbf{R}^\top$, given that the occurrence of outliers in the activations is random, we have developed an *adaptive sparse-dense kernel* based on (Flegar & Quintana-Ortí, 2017). This approach assigns an equal number of nonzero entries to each thread. Although this necessitates additional synchronization, as each column of $\mathbf{L}$ is processed by multiple threads, it ensures a more balanced workload in terms of time. Since the computations for $\mathbf{X}\mathbf{S}_W$ and $\mathbf{S}_X\mathbf{L}\mathbf{R}^\top$ are independent, the equidistributed and adaptive sparse-dense kernels are launched in a single call. The detailed descriptions of these two kernels are demonstrated in Appendix E.

5. Experiments

In this section, we conduct empirical experiments to demonstrate the efficacy of our Bit-DGDM[1]. We include quantization cost analysis and other results in Appendix F.

5.1. Experimental Setup

Models and Datasets. We conduct a comprehensive evaluation of Bit-DGDM's effectiveness in the quantization of various DGDMs and their applications. For 2D structured graphs, where node relationships are represented using adjacency matrices, we employ the widely used discrete DGDM, Digress (Vignac et al., 2023), which utilize FiLM (Perez et al., 2018) and Graph Transformer (Dwivedi & Bresson, 2020) as backbones. These models are applied to molecular synthesis datasets, including QM9 (Wu et al., 2018) and MOSES (Polykovskiy et al., 2020), as well as non-molecular benchmarks (Martinkus et al., 2022) such as SBM and planar graphs. Moreover, we investigate protein inverse folding task, an essential application of graph generation, where node relationships are provided by 3D coordinates and assess the model's ability to recover the correct amino acid sequence given protein's 3D structure. We utilize the SOTA GRADE-IF (Yi et al., 2023) as the DGDM, based on Equivariant Graph Neural Network (Satorras et al., 2021).

Baselines. As there is a lack of PTQ baselines for DGDMs, we compare Bit-DGDM against various PTQ methods for LLMs and IDMs including RTN (round-to-nearest), GPTQ (Frantar et al., 2023), SqueezeLLM (Kim et al., 2024), SVDQuant (Li et al., 2024b), DuQuant (Lin et al., 2024). Their descriptions are provided in Appendix F.1.

Metrics. The metrics for evaluating the effectiveness of molecular graph generation include Validity, Uniqueness, and Novelty, while the key metrics for assessing 3D protein inverse folding are Perplexity and Recovery (detailed descriptions can be found in Appendix F.2). The efficiency of model inference after quantization is evaluated using Speedup and Peak Memory Usage (Mem.).

Quantization details. For Bit-DGDM, we first determine the thresholds, τ_{max} and τ_{min}, for identifying activation outliers by selecting the top 0.1% highest and lowest values, respectively. To obtain the activation values, we randomly generate 32 samples using the full-precision DGDMs. For low-rank weight decomposition, we set the rank r to 32, the α-sparsity parameter α to 1%, and the step size η to 0.1.

Latency Profiling. We use the Torch CUDA profiler to measure the latency and peak memory usage for generating graphs with a batch size of 16 on a single NVIDIA RTX-3090 GPU, complemented by two 2.20GHz Intel Xeon Gold 5220R CPU, and 512GB of CPU memory. The node size n and cardinalities of nodes and edges $|\mathcal{X}|, |\mathcal{E}|$ is decided based on the dataset settings.

[1]Our code is publicly available here.

Table 2. Inverse protein folding performance of full-precision GRADE-IF (Yi et al., 2023) (*i.e.,* baseline) and quantized GRADE-IF with different quantization methods on CATH dataset.

Protein Folding		CATH			
Method	**Precision**	**Perplexity** (↓)	**Recovery** (%)	**Speedup** (↑)	**Mem.** (GB ↓)
baseline	FP32	4.4	52.2	1×	12.8
	BF16	4.4	52.1	1.3×	6.8
RTN	W4A4	7.6	32.8	3.0×	3.7
GPTQ	W4A4	7.1	35.4	3.0×	3.7
SVDQuant	W4A4	6.3	39.5	2.1×	4.6
DuQuant	W4A4	5.6	42.7	2.1×	4.7
SqueezeLLM	W4A16	4.7	48.6	1.0×	5.4
Bit-DGDM	W4A4	4.5	51.6	2.5×	4.9

5.2. Graph Generation Results

The graph generation performance of molecules and non-molecular benchmarks are shown in Table 1 and Table A2, respectively. Overall, Bit-DGDM achieves graph generation quality closer to BF16 while delivering higher computational speed and significantly reduced memory usage compared to quantization baselines. For instance, on the QM9 dataset, Bit-DGDM surpasses the state-of-the-art quantization method DuQuant in graph generation quality, with a 1.7% improvement on the Validity metric and a 0.9% improvement on the Uniqueness metric. Additionally, it achieves higher computational efficiency, with a speedup of 2.5× compared to DuQuant's 2.1×. These results demonstrate the superiority of Bit-DGDM in quantizing DGDM models. Notably, while SqueezeLLM stores weights as low-bit values using a lookup table, it still loads them as BF16 and performs weight-activation multiplications in BF16. SqueezeLLM does not achieve significant acceleration on these datasets, indicating that reducing model weight loading contributes only marginally to the acceleration of DGDMs. This finding supports the argument that DGDMs are low-parameter and high-computation models. Moreover, Bit-DGDM significantly outperforms SVDQuant, which mitigates the impact of outliers through smoothing and SVD, as well as DuQuant, which addresses outlier effects via rotation, by a large margin. This demonstrates that outlier-aware quantization methods designed for LLMs and IDMs are inefficient when applied to DGDMs.

5.3. Inverse Protein Folding Results

Inverse protein folding is a critical application of graph generation. In Table 2, we validate the effectiveness of Bit-DGDM in quantizing and accelerating DGDM. Bit-DGDM achieves the best performance among all quantization baselines in terms of both Perplexity and Recovery metrics, remaining very close to the performance of BF16 precision.

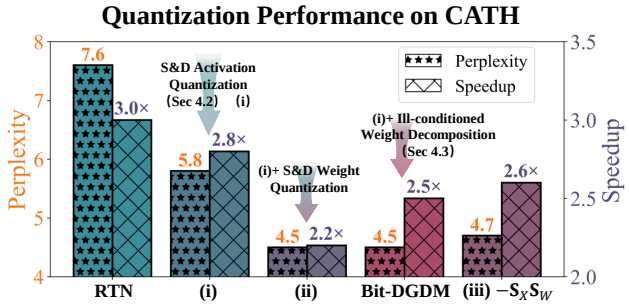

Figure 4. Perplexity and speedup on CATH with different ablations of Bit-DGDM components. "S&D" denotes sparse-dense.

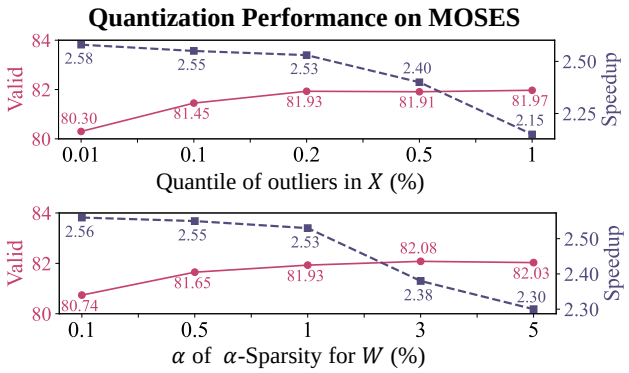

Figure 5. Results of valid ratio and speedup on MOSES with different hyperparameters.

Furthermore, compared to BF16, Bit-DGDM significantly reduces memory usage and achieves nearly 2× acceleration in computational speed. This result highlights the generalizability of Bit-DGDM across various DGDM generation models. It demonstrates that Bit-DGDM not only maintains high-generation quality but also effectively balances computational efficiency and resource utilization, making it a practical choice for real-world applications.

5.4. Ablation Study

To validate the effectiveness of our proposed module, Figure 4 presents a performance comparison between Bit-DGDM and several variants on CATH dataset. (i) "RTN + S&D activation quantization" represents the RTN baseline integrated with our sparse-dense activation quantization (Sec.4.2). (ii) "(i) + S&D weight quantization" extends variant (i) by incorporating our ill-conditioned low-rank weight decomposition module. (iii) Bit-DGDM$-\mathbf{S}_X\mathbf{S}_W$ removes the term $\mathbf{S}_X\mathbf{S}_W$ in Eqn.(14). Our findings reveal that: (1) Sparse-dense activation quantization significantly enhances model accuracy (i vs. RTN). (2) Ill-conditioned weight decomposition improves efficiency while maintaining comparable performance (ii vs. Bit-DGDM). (3) Overlapping outliers in weights and activations substantially degrade performance (iii vs. Bit-DGDM).

5.5. Hyperparameter Analysis

In our proposed Bit-DGDMs, the quantile of activations with extreme values is defined as outliers, and the α-Sparsity configuration for weights serves as two critical hyperparameters that balance the model's inference efficiency and predictive performance. Figure 5 depicts our evaluation of diverse quantile of activation outliers and α-Sparsity configurations on MOSES dataset. Experimental results indicate that selecting an appropriate number of outliers ensures comparable performance without significantly reducing inference speed. Excessive outliers slow down inference, while too few compromise performance.

6. Conclusion

This study investigated a crucial research question: post-training quantization for DGDMs. The aim is to reduce the memory usage and enable faster inference of DGDMs without compromising model performance or the quality of generated graphs. To tackle this issue, we proposed a novel quantization framework for DGDMs called Bit-DGDM. Within this framework, we developed a sparse-dense activation quantization and an ill-conditioned low-rank weight decomposition to mitigate the outlier impacts in the quantization and improve computation efficiency. Furthermore, we implemented equidistributed and adaptive sparse-dense kernels to accelerate sparse-dense matrix multiplication. Our comprehensive experimental results demonstrated the effectiveness of our framework in quantizing DGDMs.

Acknowledgement

This work is partly supported by the National Natural Science Foundation of China (No. 62306255, 92370204), the National Key Research and Development Program of China (No. 2023YFF0725000), the Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515011839), the Fundamental Research Project of Guangzhou (No. 2024A04J4233), and the Education Bureau of Guangzhou Municipality.

Impact Statement

This paper presents a machine learning method that enhances computational efficiency. The technique aims to expand the accessibility of machine learning across various sectors, without anticipated negative social impacts. Our goal is to foster innovation and inclusivity, making advanced technologies available to a broader audience of developers.

References

Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., and King, I. Binarybert: Pushing the limit of bert quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4334–4348, 2021.

Bondarenko, Y., Nagel, M., and Blankevoort, T. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7947–7969, 2021.

Brockschmidt, M., Allamanis, M., Gaunt, A. L., and Polozov, O. Generative code modeling with graphs. In *International Conference on Learning Representations*, 2018.

Buluç, A., Fineman, J. T., Frigo, M., Gilbert, J. R., and Leiserson, C. E. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pp. 233–244, 2009.

Cai, H., Liu, J., and Yin, W. Learned robust pca: A scalable deep unfolding approach for high-dimensional outlier detection. *Advances in Neural Information Processing Systems*, 34:16977–16989, 2021.

Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3): 1–37, 2011.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

Chen, Y. Incoherence-optimal matrix completion. *IEEE Transactions on Information Theory*, 61(5):2909–2923, 2015.

De Cao, N. and Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Dettmers, T., Svirschevski, R. A., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. In *The Twelfth International Conference on Learning Representations*, 2024.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Egiazarian, V., Panferov, A., Kuznedelev, D., Frantar, E., Babenko, A., and Alistarh, D. Extreme compression of large language models via additive quantization. In *Forty-first International Conference on Machine Learning*, 2024.

Flegar, G. and Quintana-Ortí, E. S. Balanced csr sparse matrix-vector product on graphics processors. In *Euro-Par 2017: Parallel Processing: 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28–September 1, 2017, Proceedings 23*, pp. 697–709. Springer, 2017.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate post-training compression for generative pretrained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M. W., and Keutzer, K. Ai and memory wall. *IEEE Micro*, (01): 1–5, 2024.

Gong, Z. and Sun, Y. An energy-centric framework for category-free out-of-distribution node detection in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 908–919, 2024a.

Gong, Z. and Sun, Y. Graph reasoning enhanced language models for text-to-sql. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2447–2451, 2024b.

Gong, Z., Wang, G., Sun, Y., Liu, Q., Ning, Y., Xiong, H., and Peng, J. Beyond homophily: Robust graph anomaly detection via neural sparsification. In *IJCAI*, pp. 2104–2113, 2023.

Haefeli, K. K., Martinkus, K., Perraudin, N., and Wattenhofer, R. Diffusion models for graphs benefit from discrete state spaces. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

Han, X., Cao, M., Xu, D., Feng, X., Liang, Y., Lang, X., and Guan, R. Seoe: an option graph based semantically embedding method for prenatal depression detection. *Frontiers of Computer Science*, 18(6):186911, 2024.

He, Y., Liu, J., Wu, W., Zhou, H., and Zhuang, B. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024a.

He, Y., Liu, L., Liu, J., Wu, W., Zhou, H., and Zhuang, B. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024b.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Huang, M., Liu, Y., Ao, X., Li, K., Chi, J., Feng, J., Yang, H., and He, Q. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM Web Conference 2022*, pp. 1311–1321, 2022.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Jeon, Y., Lee, C., Cho, E., and Ro, Y. Mr. biq: Post-training non-uniform quantization based on minimizing the reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12329–12338, 2022.

Ji, Y., Sun, Y., Zhang, Y., Wang, Z., Zhuang, Y., Gong, Z., Shen, D., Qin, C., Zhu, H., and Xiong, H. A comprehensive survey on self-interpretable neural networks. *arXiv preprint arXiv:2501.15638*, 2025.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pp. 10362–10383. PMLR, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

Kim, S., Hooper, C. R. C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. Squeezellm: Dense-and-sparse quantization. In *Forty-first International Conference on Machine Learning*, 2024.

Lee, W., Lee, J., Seo, J., and Sim, J. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 155–172, 2024.

Li, K., Chen, Y., Liu, Y., Wang, J., He, Q., Cheng, M., and Ao, X. Boosting the adversarial robustness of graph neural networks: An ood perspective. In *The Twelfth International Conference on Learning Representations*, 2024a.

Li, M., Lin, Y., Zhang, Z., Cai, T., Li, X., Guo, J., Xie, E., Meng, C., Zhu, J.-Y., and Han, S. Svdqunat: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024b.

Li, X., Liu, Y., Lian, L., Yang, H., Dong, Z., Kang, D., Zhang, S., and Keutzer, K. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17535–17545, 2023.

Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., and Gu, S. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Lin, H., Xu, H., Wu, Y., Cui, J., Zhang, Y., Mou, L., Song, L., Sun, Z., and Wei, Y. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Lin, Y., Zhang, T., Sun, P., Li, Z., and Zhou, S. Fq-vit: Post-training quantization for fully quantized vision transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 1173–1179, 2022.

Liu, J., Huang, Z., Ma, Z., Liu, Q., Chen, E., Su, T., and Liu, H. Guiding mathematical reasoning via mastering commonsense formula knowledge. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1477–1488, 2023a.

Liu, J., Huang, Z., Zhai, C., and Liu, Q. Learning by applying: A general framework for mathematical reasoning via enhancing explicit knowledge learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4497–4506, 2023b.

Liu, Y., Ao, X., Feng, F., Ma, Y., Li, K., Chua, T.-S., and He, Q. Flood: A flexible invariant learning framework for out-of-distribution generalization on graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1548–1558, 2023c.

Liu, Y., Du, C., Pang, T., Li, C., Lin, M., and Chen, W. Graph diffusion policy optimization. *arXiv preprint arXiv:2402.16302*, 2024.

Lv, C., Chen, H., Guo, J., Ding, Y., and Liu, X. Ptq4sam: Post-training quantization for segment anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15941–15951, 2024.

Martinkus, K., Loukas, A., Perraudin, N., and Wattenhofer, R. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., and Blankevoort, T. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Netrapalli, P., UN, N., Sanghavi, S., Anandkumar, A., and Jain, P. Non-convex robust pca. *Advances in neural information processing systems*, 27, 2014.

Newman, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 4474–4484. PMLR, 2020.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Shang, Y., Yuan, Z., Xie, B., Wu, B., and Yan, Y. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1972–1981, 2023.

Shen, D., Song, G., Xue, Z., Wang, F.-Y., and Liu, Y. Rethinking the spatial inconsistency in classifier-free diffusion guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9370–9379, 2024.

Shi, C., Chen, J., Liu, J., and Yang, C. Graph foundation model. *Frontiers of Computer Science*, 18(6):186355, 2024.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pp. 412–422. Springer, 2018.

So, J., Lee, J., Ahn, D., Kim, H., and Park, E. Temporal dynamic quantization for diffusion models. *Advances in neural information processing systems*, 36:48686–48698, 2023.

Sun, Y., Zhu, H., Wang, L., Zhang, L., and Xiong, H. Large-scale online job search behaviors reveal labor market shifts amid covid-19. *Nature Cities*, 1(2):150–163, 2024.

Team, G., Riviere, M., Pathak, S., Sessa, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Tong, T., Ma, C., and Chi, Y. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. *J. Mach. Learn. Res.*, 22:150:1–150:63, 2021.

Tseng, A., Sun, Q., Hou, D., and De Sa, C. Qtip: Quantization with trellises and incoherence processing. *arXiv preprint arXiv:2406.11235*, 2024.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023.

Wang, C., Wang, Z., Xu, X., Tang, Y., Zhou, J., and Lu, J. Towards accurate post-training quantization for diffusion models. *arXiv preprint arXiv:2305.18723*, 2023.

Wang, P., Chen, Q., He, X., and Cheng, J. Towards accurate post-training network quantization via bit-split and stitching. In *International Conference on Machine Learning*, pp. 9847–9856. PMLR, 2020.

Wei, X., Gong, R., Li, Y., Liu, X., and Yu, F. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *International Conference on Learning Representations*, 2022a.

Wei, X., Zhang, Y., Zhang, X., Gong, R., Zhang, S., Zhang, Q., Yu, F., and Liu, X. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414, 2022b.

Westfall, P. H. Kurtosis as peakedness, 1905–2014. rip. *The American Statistician*, 68(3):191–195, 2014.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

Yi, K., Zhou, B., Shen, Y., Liò, P., and Wang, Y. Graph denoising diffusion for inverse protein folding. *Advances in Neural Information Processing Systems*, 36, 2023.

Yi, X., Park, D., Chen, Y., and Caramanis, C. Fast algorithms for robust pca via gradient descent. *Advances in neural information processing systems*, 29, 2016.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Zadeh, A. H., Edo, I., Awad, O. M., and Moshovos, A. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 811–824. IEEE Computer Society, 2020.

Zang, C. and Wang, F. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 617–626, 2020.

Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7920–7928, 2018.

A. Related Work

A.1. Quantization for LLMs

Quantization methods can be categorized into two types based on whether retraining is required. The first type, Quantization-Aware Training (QAT), necessitates retraining the model after quantizing its parameters to maintain performance comparable to the original model (Bai et al., 2021). The second type, Post-Training Quantization (PTQ), directly quantizes model parameters without retraining (Kim et al., 2024; Frantar et al., 2023). Although QAT generally achieves superior performance, due to the substantial computational cost of training LLMs, most studies adopt PTQ.

Many LLM-specific quantization studies have identified that the primary bottleneck in LLM inference (Liu et al., 2023a;b) lies in the weight loading process rather than computation speed (Lee et al., 2024; Gholami et al., 2024). Given the vast number of model parameters, the time required to transfer weights from GPU/CPU memory to the computation cache often exceeds the actual computation time, meaning that merely accelerating computation does not significantly enhance inference speed. To address this issue, some works have proposed non-uniform quantization methods to accelerate weight loading. Unlike uniform quantization (Dettmers et al., 2024; Frantar et al., 2023), which partitions weight values into bins at equal intervals and maps each bin to an integer, non-uniform quantization (Zadeh et al., 2020; Kim et al., 2024; Jeon et al., 2022) employs clustering techniques such as K-means to group weight values into a limited set of representative values. These values are stored in a lookup table, and each weight value is replaced by the index of its closest match in the table. During model loading, only the integer indices and a small set of floating-point values from the lookup table need to be retrieved, enabling rapid weight reconstruction.

Furthermore, to mitigate the weight loading issue, some works leverage Multi-Codebook Quantization retrieval methods, which quantize LLM weights at the vector level rather than as individual numerical values (Egiazarian et al., 2024; Tseng et al., 2024). In these approach, the lookup table stores floating-point vectors, and model parameters are represented as a operation of multiple vectors from the table, *e.g.*, additive quantization (Egiazarian et al., 2024). However, these methods often require a small amount of labeled data to finetune the vectors stored in the codebook. Given that DGDMs have significantly fewer parameters than LLMs, these memory-oriented quantization techniques are inefficient when applied to DGDMs, as discussed in Sec. 5.2.

A.2. Quantization for IDMs

Some quantization works (Li et al., 2023; Shang et al., 2023; Wang et al., 2023) for image diffusion models (IDMs) (Shen et al., 2024) optimize quantized weights by minimizing the MSE between quantized and full-precision continuous outputs. (So et al., 2023) introduces time-step specific encoding for image diffusion models. (Lv et al., 2024) addresses bimodal distributions in post-key-linear layers and post-softmax discrepancies, specific to Segment Anything Models. However, these approaches cannot be directly applied to Discrete Graph Diffusion Models (DGDMs) due to fundamental incompatibilities. As shown in Remark 3.1, in DGDMs, the intermediate node attributes and graph structures are obtained through discrete sampling from categorical distributions. While IDMs relay on Gaussian noise and continuous denoising processes. Our method advances existing method in three innovations. (i) Recognizing the significant outliers in model weights, we first propose an ill-conditioned low-rank weight decomposition. This contrasts with basic SVD approach, allowing our method to achieve better numerical stability. (ii) For the residual component derived from the raw weight and low-rank decomposition, our method enforces α-sparsity, enabling efficient sparse matrix multiplication during inference. (iii) For activation outliers, our approach eliminates the need for calibration data to select thresholds, making it more practical.

A.3. Outlier-Aware Quantization

In low-bit model quantization, a critical challenge is the presence of outliers, which can unnecessarily expand the quantization range. To address this issue, outlier-aware quantization methods (Bondarenko et al., 2021) have been proposed. Wei et al. (2022b) alleviate outlier influences by transferring outlier factors to later layers without affecting model functionality. Xiao et al. (2023) introduce a method that shifts the quantization difficulty from activations to weights by applying channel-wise scaling to reduce the magnitude of activations while increasing the magnitude of weights. Li et al. (2024b) further extend this approach by employing SVD to decompose weights and quantize the residuals of its principal components. SqueezeLLM (Kim et al., 2024) improves the robustness of non-uniform quantization by enhancing K-means clustering to reduce sensitivity to outliers, proposing a sensitivity-based clustering method. DuQuant (Lin et al., 2024) adopts a block-wise rotation strategy to redistribute outliers across adjacent channels. Unlike these approaches, our proposed Bit-DGDM

introduces two novel methods: (1) Sparse-Dense quantization, which provides a simple and effective solution for handling activations, and (2) Ill-conditioned Low-Rank Decomposition, which decomposes the weight matrix into two components, ensuring computational efficiency while maintaining precision. These innovations yield substantial improvements over existing outlier-aware quantization frameworks.

B. Theoretical Proof

In Sec. 4.3, we have introduced Theorem 4.5 to promise the recovery guarantee of our proposed weight decomposition approach with respect to the ideal dense low-rank matrix $\mathbf{D}_W^*$. We now illustrate how it can be derived. Note that our guaranteed recovery theorem follows the route built in (Tong et al., 2021; Cai et al., 2021). We first introduce basic definitions. Let $\mathbf{L}_\star := \mathbf{U}_\star \mathbf{\Sigma}_\star^{1/2}$ and $\mathbf{R}_\star := \mathbf{V}_\star \mathbf{\Sigma}_\star^{1/2}$ where $\mathbf{U}_\star \mathbf{\Sigma}_\star \mathbf{V}_\star^\top$ is the compact SVD of $\mathbf{D}_W^\star$. The error metric for decomposed rank- r matrices is defined as:

$$\operatorname{dist}\left(\mathbf{L}, \mathbf{R}; \mathbf{L}_\star, \mathbf{R}_\star\right) := \inf_{\mathbf{Q} \in \mathbb{R}^{r \times r}, \operatorname{rank}(\mathbf{Q}) = r} \left(\left\| \left(\mathbf{L}\mathbf{Q} - \mathbf{L}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_F^2 + \left\| \left(\mathbf{R}\mathbf{Q}^{-\mathrm{T}} - \mathbf{R}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_F^2 \right)^{1/2}$$

where the optimal alignment matrix $\mathbf{Q}$ exists and invertible if $\mathbf{L}$ and $\mathbf{R}$ are sufficiently close to $\mathbf{L}_\star$ and $\mathbf{R}_\star$. Particularly, the following lemma exists

Lemma B.1 (Lemma 9 in Tong et al. (2021)). *For any $\mathbf{L} \in \mathbb{R}^{n \times r}$ and $\mathbf{R} \in \mathbb{R}^{m \times r}$, if*

$$\operatorname{dist}\left(\mathbf{L}, \mathbf{R}; \mathbf{L}_\star, \mathbf{R}_\star\right) < c\sigma_r\left(\mathbf{D}_W^\star\right)$$

for some $0 < c < 1$, then the optimal alignment matrix $\mathbf{Q}$ between $[\mathbf{L}, \mathbf{R}]$ and $[\mathbf{L}_\star, \mathbf{R}_\star]$ exists and is invertible.

We then present the theorems of local linear convergence and guaranteed initialization.

Theorem B.2 (Local Linear Convergence, Theorem 3 in Cai et al. (2021)). *Let $\mathbf{D}_W^\star = \mathbf{L}_\star \mathbf{R}_\star^\top$ be a rank-r matrix with incoherence parameter μ, and let $\mathbf{S}_\star$ be an α-sparse matrix where $\alpha \leq \frac{1}{10^4 \mu r^{\frac{3}{2}}}$. Define $\mathbf{Q}_k$ as the optimal alignment matrix between $[\mathbf{L}_k, \mathbf{R}_k]$ and $[\mathbf{L}_\star, \mathbf{R}_\star]$. Assuming the initial guesses satisfy the prescribed conditions, we have the following:*

$$\begin{aligned} &\operatorname{dist}\left(\mathbf{L}_0, \mathbf{R}_0; \mathbf{L}_\star, \mathbf{R}_\star\right) \leq \varepsilon_0 \sigma_r\left(\mathbf{D}_W^\star\right), \\ &\left\| \left(\mathbf{L}_0\mathbf{Q}_0 - \mathbf{L}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \vee \left\| \left(\mathbf{R}_0\mathbf{Q}_0^{-\top} - \mathbf{R}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}} \sigma_r\left(\mathbf{D}_W^\star\right) \end{aligned}$$

with $\varepsilon_0 := 0.02$, by using a fixed step size $\eta_k = \eta$ where $\eta \in \left[0.1, \frac{2}{3}\right]$, the iterates of weight decomposition satisfy the following conditions:

$$\begin{aligned} &\operatorname{dist}\left(\mathbf{L}_k, \mathbf{R}_k; \mathbf{L}_\star, \mathbf{R}_\star\right) \leq \varepsilon_0 \tau^k \sigma_r\left(\mathbf{D}_W^\star\right), \\ &\left\| \left(\mathbf{L}_k\mathbf{Q}_k - \mathbf{L}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \vee \left\| \left(\mathbf{R}_k\mathbf{Q}_k^{-\top} - \mathbf{R}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}} \tau^k \sigma_r\left(\mathbf{D}_W^\star\right), \end{aligned}$$

where the convergence rate $\tau := 1 - 0.6\eta$.

Theorem B.3 (Guaranteed Initialization, Theorem 4 in Cai et al. (2021)). *Suppose $\mathbf{D}_W^\star = \mathbf{L}_\star \mathbf{R}_\star^\top$ is a rank-r matrix with incoherence parameter μ, and $\mathbf{S}_\star$ is an α-sparse matrix such that $\alpha \leq \frac{c_0}{\mu r^{\frac{3}{2}} \kappa}$, where c_0 is a small positive constant with $c_0 \leq \frac{1}{35}$. Let $\mathbf{Q}_0$ denote the optimal alignment matrix between $[\mathbf{L}_0, \mathbf{R}_0]$ and $[\mathbf{L}_\star, \mathbf{R}_\star]$. It has the following results:*

$$\begin{aligned} &\operatorname{dist}\left(\mathbf{L}_0, \mathbf{R}_0; \mathbf{L}_\star, \mathbf{R}_\star\right) \leq 10 c_0 \sigma_r\left(\mathbf{D}_W^\star\right), \\ &\left\| \left(\mathbf{L}_0\mathbf{Q}_0 - \mathbf{L}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \vee \left\| \left(\mathbf{R}_0\mathbf{Q}_0^{-\top} - \mathbf{R}_\star\right) \mathbf{\Sigma}_\star^{1/2} \right\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}} \sigma_r\left(\mathbf{D}_W^\star\right). \end{aligned}$$

Besides, we cite a conclusion from Tong et al. (2021) as Lemma B.4:

Lemma B.4 (Eqn. (48) in Tong et al. (2021))**.**

$$\left\|\mathbf{L}_k\mathbf{R}_k^\top - \mathbf{D}_W^\star\right\|_{\mathrm{F}} \leq 1.5\,\mathrm{dist}\left(\mathbf{L}_k, \mathbf{R}_k; \mathbf{L}_\star, \mathbf{R}_\star\right)$$

.

Given the above lemmas and theorems, we are ready to prove Theorem 4.5:

Proof of Theorem 4.5. The results of Theorem B.3 meet the conditions stipulated in Theorem B.2. By setting $c_0 = 10^{-4}$, we obtain the following:

$$\mathrm{dist}(\mathbf{L}_k, \mathbf{R}_k; \mathbf{L}^*, \mathbf{R}^*) \leq 0.02(1-0.6\eta)^k \sigma_r(\mathbf{D}_W^*),$$

$$\left\|\left(\mathbf{L}_k\mathbf{Q}_k - \mathbf{L}_\star\right)\mathbf{\Sigma}_\star^{1/2}\right\|_{2,\infty} \vee \left\|\left(\mathbf{R}_k\mathbf{Q}_k^{-\top} - \mathbf{R}_\star\right)\mathbf{\Sigma}_\star^{1/2}\right\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}}(1-0.6\eta)^k \sigma_r\left(\mathbf{D}_W^\star\right)$$

for all $k \geq 0$. Based on Lemma B.4, we can proves that

$$\left\|\mathbf{L}_k\mathbf{R}_k^\top - \mathbf{D}_W^\star\right\|_{\mathrm{F}} \leq 1.5\,\mathrm{dist}\left(\mathbf{L}_k, \mathbf{R}_k; \mathbf{L}_\star, \mathbf{R}_\star\right) \leq 0.03(1-0.6\eta)^k \sigma_r(\mathbf{D}_W^*)$$

under the condition of $\left\|\left(\mathbf{L}_k\mathbf{Q}_k - \mathbf{L}_\star\right)\mathbf{\Sigma}_\star^{1/2}\right\|_{2,\infty} \vee \left\|\left(\mathbf{R}_k\mathbf{Q}_k^{-\top} - \mathbf{R}_\star\right)\mathbf{\Sigma}_\star^{1/2}\right\|_{2,\infty} \leq \sqrt{\frac{\mu r}{n}}\sigma_r\left(\mathbf{D}_W^\star\right)$. Thus our claim is proved.

□

C. Backbones of Graph Diffusion Models

C.1. 2D Graph Diffusion Model

Regarding the input for the graph generation task, we have already introduced it in Sec. 3 of the main text. We will now proceed to describe the model architecture.

Architecture. We employ a typical graph diffusion model, Digress (Vignac et al., 2023), as the target for our model quantization acceleration. Digress utilizes the adjacency matrix of nodes to model the structure of graphs and adopts a Graph Transformer architecture as its backbone to represent nodes and structural features. The structure is illustrated in Figure A1. Notably, FiLM and PNA modules are critical components, and their computational processes are described as:

$$\begin{aligned} \mathrm{FiLM}(\mathbf{M}_1, \mathbf{M}_2) &= \mathbf{M}_1\mathbf{W}_1 + (\mathbf{M}_1\mathbf{W}_2) \odot \mathbf{M}_2 + \mathbf{M}_2, \\ \mathrm{PNA}(\mathbf{X}) &= [\max(\mathbf{X}) : \min(\mathbf{X}) : \mathrm{mean}(\mathbf{X}) : \mathrm{std}(\mathbf{X})]\mathbf{W}, \end{aligned} \tag{15}$$

where $\odot$ denotes the element-wise product and $[:]$ denote the concatenation.

C.2. Protein Inverse Folding Diffusion Model

Regarding the protein inverse folding task, we provide detailed descriptions of the input and the model architecture.

Input. The protein graph $\mathcal{G} = \mathbf{X}, \mathbf{E}$ comprises node feature $\mathbf{X}$ and edge features $\mathbf{E}$. The node feature depicts the amino acid (AA) position, AA type and the spatial and biochemical properties to reflect its physicochemical and topological attributes. The local structure of a given node is defined by its spatial neighbors, determined by the k-nearest neighbor algorithm following (Yi et al., 2023). The edge attributes illustrate the relationships between connected nodes, such as inter-atomic distances, local N-C positions and a sequential position encoding scheme.

Architecture. For protein-related tasks, modeling the 3D structure between nodes is crucial due to the importance of distances, angles, and other spatial properties to protein characteristics. We select GRADE-IF (**?**) as the DGDM for protein tasks. GRADE-IF employs an enhanced Equivariant Graph Convolution (EGC) layer to ensure SO(3) rotation equivariance and E(3) translation invariance. At the l-th layer, the EGC layer uses n node representations $\mathbf{h}_i^{(l)}$ to describe each node's

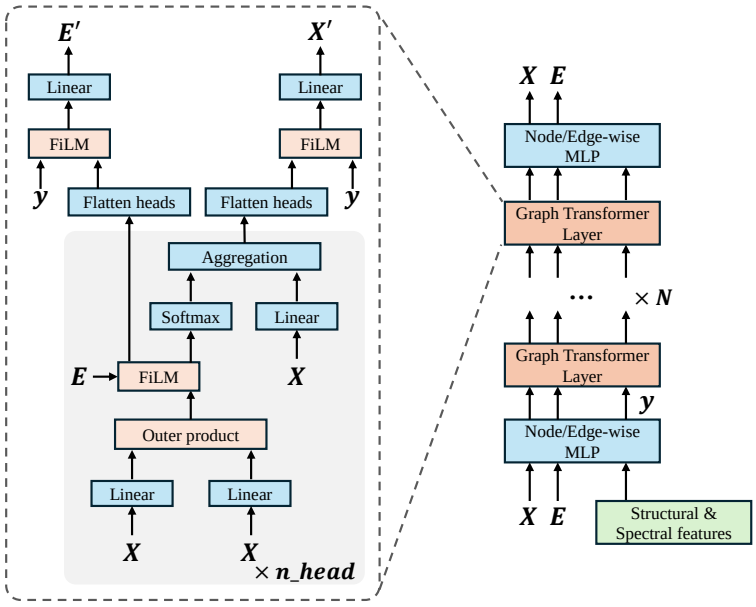

Figure A1. An illustration of graph transformer architectures used in Digress.

amino acid (AA) type and geometric properties, edge embeddings $\mathbf{m}_{ij}^{(l)}$ to connect nodes i and j, and $\mathbf{x}_i^{\text{pos}}$ to describe the coordinates of the nodes. The EGC layer updates the node representations as follows:

$$\begin{aligned}
\mathbf{m}_{ij}^{(l+1)} &= \phi_e\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \left\|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right\|^2, \mathbf{m}_{ij}^{(l)}\right) \\
\mathbf{x}_i^{(l+1)} &= \mathbf{x}_i^{(l)} + \frac{1}{n}\sum_{j\neq i}\left(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right)\phi_x\left(\mathbf{m}_{ij}^{(l+1)}\right) \\
\mathbf{h}_i^{(l+1)} &= \phi_h\left(\mathbf{h}_i^{(l)}, \sum_{j\neq i}\mathbf{m}_{ij}^{(l+1)}\right),
\end{aligned} \tag{16}$$

where ϕ_e, ϕ_h are the operations of edge and node propagations, respectively, and ϕ_x is an operation that projects the edge embedding to a scalar.

D. Algorithm and Complexity Analysis

Figure 3 provides a succinct overview of the comprehensive quantization process in Bit-DGDM. To enhance clarity and facilitate comprehension, we delineate the systematic protocol for the overall inference process of quantized model in Algorithm 1. Besides, we extensively illustrate the procedure of ill-conditioned low-rank decomposition of weights (Sec. 4.3) in Algorithm 2. This section presents a complexity analysis of the ill-conditioned low-rank decomposition. First, the initializations of the α-Sparsity matrix $\mathbf{S}_{W,0}$ and low-rank component $\mathbf{L}_0\mathbf{R}_0^\top$ requires $2nm + nmr + nr^2 + mr^2$ flops. The computational complexity w.r.p the iterative optimizations of these components is $\mathcal{O}(nmr + mr^2 + nr^2 + r^3)$ flops. Since the rank r is typically much less than the channel sizes n, m of weight $\mathbf{W}$, the overall complexity of the ill-conditioned low-rank decomposition requires $\mathcal{O}(nmr)$.

E. Equidistributed and Adaptive Sparse-Dense Kernel Implementation

In this section, we will introduce the core concept of our designed Equidistributed and Adaptive Sparse-Dense kernel. For each sparse matrix, we utilize the CSR (Compressed Sparse Row) format (Buluç et al., 2009) to represent it. The CSR format is a storage scheme that efficiently represents sparse matrices by storing non-zero elements along with their row and column indices, thereby reducing memory usage and improving computational efficiency.

Algorithm 1 The inference process of quantized multiplication $\mathbf{XW}$ through Bit-DGDM

Input: $\mathbf{X} \in \mathbb{R}^{b\times m}$: activations, $\tau_{\text{max}}, \tau_{\text{min}}$: thresholds for filter activation outliers, $\hat{\mathbf{D}}_W$: quantized dense part of weight $\mathbf{W}$, $\mathbf{S}_W$: sparse component of weight $\mathbf{W}$, $\mathbf{LR}^\top$: low rank component of weight $\mathbf{W}$.
▷ Filter the activation outliers: $\mathbf{S}_X = \mathbf{X}[x > \tau_{\text{max}} | x < \tau_{\text{min}}]$, $\mathbf{D}_X = \mathbf{X}[\tau_{\text{min}} < x < \tau_{\text{max}}]$
▷ Quantize the $\mathbf{D}_X$ to low-bit $\hat{\mathbf{D}}_X$ through Eqn. (4)
▷ Calculating $\hat{\mathbf{D}}_X\hat{\mathbf{D}}_W$ through low-bit multiplication
▷ Calculating $\mathbf{S}_X\mathbf{LR}^\top$ through adaptive sparse-dense kernel
▷ Calculating $\mathbf{XS}_W$ through equidistributed sparse-dense kernel
Output: $\hat{\mathbf{D}}_X\hat{\mathbf{D}}_W + \mathbf{S}_X\mathbf{LR}^\top + \mathbf{XS}_W$

Algorithm 2 Ill-conditioned low-rank decomposition of weight $\mathbf{W}$

Input: $\mathbf{W} \in \mathbb{R}^{m\times n}$: weight matrix, r: the rank of underlying low-rank matrix, $\mathcal{T}_\alpha(\cdot)$: sparsification operator, $\{\eta_k\}$: a set of step size.
// Initialization:
$\mathbf{S}_{W,0} = \mathcal{T}_\alpha(\mathbf{W})$ $\rightarrow nm$ *flops*
$[\mathbf{U}_0, \mathbf{\Sigma}_0, \mathbf{V}_0] = \text{SVD}_r(\mathbf{W} - \mathbf{S}_{W,0})$ $\rightarrow nmr + nm$ *flops*
$\mathbf{L}_0 = \mathbf{U}_0\mathbf{\Sigma}_0^{\frac{1}{2}}, \mathbf{R}_0 = \mathbf{V}_0^\top\mathbf{\Sigma}_0^{\frac{1}{2}}$ $\rightarrow nr^2 + mr^2$ *flops*
// Iterative optimization:
repeat
  $\mathbf{S}_{W,k+1} = \mathcal{T}_\alpha(\mathbf{W} - \mathbf{L}_k\mathbf{R}_k^\top)$ $\rightarrow \mathcal{O}(nmr)$ *flops*
  $\mathbf{L}_{k+1} = \mathbf{L}_k - \eta_{k+1}\left(\mathbf{L}_k\mathbf{R}_k^\top + \mathbf{S}_{W,k+1} - \mathbf{W}\right)\mathbf{R}_k\left(\mathbf{R}_k^\top\mathbf{R}_k\right)^{-1}$ $\rightarrow \mathcal{O}(nmr + mr^2 + r^3)$ *flops*
  $\mathbf{R}_{k+1} = \mathbf{R}_k - \eta_{k+1}\left(\mathbf{L}_k\mathbf{R}_k^\top + \mathbf{S}_{W,k+1} - \mathbf{W}\right)\mathbf{L}_k\left(\mathbf{L}_k^\top\mathbf{L}_k\right)^{-1}$ $\rightarrow \mathcal{O}(nmr + nr^2 + r^3)$ *flops*
until $\|\mathbf{L}_{k+1}\mathbf{R}_{k+1}^\top + \mathbf{S}_{W,k+1} - \mathbf{W}\|_F^2$ is convergent
Output: α-Sparsity matrix $\mathbf{S}_W := \mathbf{S}_{W,k}$, r-rank component $\mathbf{D}_W = \mathbf{LR}^\top(\mathbf{L} := \mathbf{L}_k, \mathbf{R} := \mathbf{R}_k)$

Equidistributed Sparse-Dense Kernel. For the Equidistributed Kernel, since the sparse matrix $\mathbf{S}_W$, which models the outliers of weights, satisfies the α-Sparsity constraint, we only need to assign one thread per row to handle the computation, ensuring load balancing.

Adaptive Sparse-Dense Kernel. The adaptive sparse-dense kernel is designed to optimize workload distribution on GPUs, particularly for matrices characterized by irregular row lengths. The core concept involves partitioning the non-zero elements into segments allocated to warps, rather than assigning entire rows to warps. The overall algorithm is shown in Algorithm 3.

The pipeline initiates with a preprocessing phase, during which the matrix is analyzed to determine the optimal segmentation of non-zero elements. A specialized array, termed **srow**, is introduced to store the starting row index for each segment. This preprocessing step entails constructing a histogram of the **rowptr** array and executing an exclusive scan operation to compute the **srow** array. This preprocessing is performed once and stored, incurring negligible computational overhead.

During the execution phase, each warp is responsible for processing its designated segment of the **val** and **colidx** arrays. Each segment comprises a contiguous block of non-zero elements, irrespective of row boundaries. The warp processes these elements in fixed-size chunks (e.g., 32 elements at a time, corresponding to the warp size). For each element, the corresponding thread multiplies the value by the respective element in the **x** vector and accumulates the result.

However, since elements from different rows may reside within the same segment, the algorithm must dynamically detect row boundaries. This is achieved by examining the **rowptr** array. When a row boundary is encountered within a chunk, the warp employs warp shuffle instructions to execute a segmented reduction. This mechanism facilitates the efficient combination of partial sums from the same row without the need for shared memory.

Upon processing a chunk, if any thread transitions to a new row, a segmented scan operation is performed to aggregate the results for each row. The first thread in the warp corresponding to each row then executes an atomic addition to update the global **y** vector. This approach minimizes the number of atomic operations by ensuring that each row is updated only once per warp after the accumulation of partial sums. Overall, the pipeline achieves load balancing by evenly distributing non-zero elements across warps and handles irregular row lengths through optimized intra-warp communication.

Algorithm 3 Adaptive Sparse-Dense Kernel on GPUs

Input: `val`, `colidx`, `rowptr`: CSR-format sparse matrix, x: input vector, `srow`: segment starting rows, n_z: number of non-zeros
// Initialize
Divide `val` and `colidx` into T warp-level segments
Segment k range: $[\lfloor kn_z/T \rfloor, \lfloor (k+1)n_z/T \rfloor)$
Precompute `srow` array via histogram of `rowptr`
// Execution on GPU
for each warp W in parallel **do**
  start$_W \leftarrow \lfloor W \cdot n_z/T \rfloor$ // Start index of segment
  end$_W \leftarrow \lfloor (W+1) \cdot n_z/T \rfloor$ // End index
  current_row $\leftarrow$ `srow`$[W]$
  local_sum $\leftarrow 0$
  // Process chunk of 32 elements (warp size)
  repeat
    Initialize $i = 0$
    for each thread t in warp **do**
      Load val$[i+t]$ and colidx$[i+t]$
      partial $\leftarrow$ val$[i+t] \times x[$colidx$[i+t]]$
      if column index crosses row boundary **then**
        Detect row changes using `rowptr`
        Perform segmented scan within warp
        Update `current_row` via warp shuffle
      end if
      Accumulate partial to local_sum
      if any thread changed row **then**
        Warp-wide segmented reduction
        Reset local sums except first thread per row
        Atomic add completed rows to global y
      end if
      $i \leftarrow i + 32$ // Move to next chunk
    end for
    // Flush remaining local sums
    Perform final warp reduction and atomic updates to y
  until current index $i \geq$ end$_W$
end for
Output: Output vector y

F. Additional Experiment Details

F.1. Baseline Description

The detailed descriptions of the baselines used for comparison in this work are as follows:

- RTN (round-to-nearest) quantizes each value of weights and activations to the nearest integer. After performing the multiplication, the values are dequantized back to floating-point numbers.
- GPTQ (Frantar et al., 2023) leverages the Hessian matrix of activations for layer-wise error calibration and performs channel-wise quantization. The quantization loss is then propagated to other channels. Additionally, it introduces techniques such as arbitrary order, lazy-batch updates, and Cholesky reformulation, ensuring efficient quantization even with a large number of parameters.
- SqueezeLLM (Kim et al., 2024) isolates a small number of weight outliers and employs a lookup table to focus all weight values on a few cluster centers in a sensitivity-aware manner. Quantized weights are stored as integers and

loaded via the lookup table, significantly reducing the LLM weight loading time. Computation is still performed using FP16 weights and activations.

- SVDQuant (Li et al., 2024b) reduces the quantization difficulty of activations by shrinking the activation magnitude and amplifying the weight magnitude. The amplified weights are decomposed into low-rank components using SVD, while the residual is quantized to low-bit precision.

- DuQuant (Lin et al., 2024), based on specific outlier dimensions as prior knowledge, employs rotation and permutation transformations to redistribute outliers to adjacent channels through block-wise rotation, thereby mitigating the impact of outliers.

F.2. Metrics

Graph Generation Evaluation. For the molecules generation tasks, we employ three widely-used metrics to evaluate the quanlity of generated graphs: *Validity*, *Uniqueness* and *Novelty*. Specifically, Validity quantifies the fraction of generated molecules that satisfy fundamental chemical valency rules, ensuring their structural feasibility. Uniqueness evaluates the diversity of the generated molecules by calculating the proportion of distinct SMILES strings, which indicates the presence of non-isomorphic molecular structures. Novelty assesses the extent to which the generated molecules differ from the training data by measuring the fraction of molecules that do not appear in the training set, thereby reflecting the model's ability to produce chemically novel compounds. Together, these metrics provide a comprehensive evaluation of the chemical validity, structural diversity, and innovation of the generated molecular outputs. Following (Vignac et al., 2023), we do not include novelty as a reported metric for QM9 in the main table. This decision is based on the fact that QM9 represents an exhaustive enumeration of small molecules adhering to a specific set of constraints. Consequently, generating molecules outside this predefined set does not inherently indicate that the model has effectively learned the underlying data distribution. Instead, it may reflect deviations from the constrained chemical space that QM9 encompasses.

Protein Inverse Folding Evaluation. In our evaluations, we focus on the protein inverse folding as an important application for the graph generation. The quality of reconstructed protein sequence is evaluated using two key metrics: *perplexity* and *Recovery*. Perplexity quantifies the degree to which the model's predicted amino acid (AA) probabilities align with the actual AA at each position in the sequence. A lower perplexity value indicates a stronger agreement between the model's predictions and the observed data, reflecting a better fit. On the other hand, the recovery rate measures the model's ability to accurately reconstruct the correct AA sequence based on the protein's 3D structure. This metric is calculated as the percentage of AAs in the predicted sequence that match those in the original sequence. A higher recovery rate signifies a superior capability of the model to infer the original sequence from the structural information. Together, these metrics provide a comprehensive assessment of the model's performance in protein inverse folding tasks.

F.3. Quantization Cost Analysis

In Table A1, we report the memory and running time required for quantizing DGDMs. Note that we use the Digress model from the QM9 dataset as a representative for validating quantized Digress, as the quantization time for Digress on other datasets is similar. The quantization overhead is mainly divided into two parts: first, determining activation outlier thresholds by randomly generating a certain number of graphs (set to 32 in our study); second, the iterative estimation required for ill-conditioned weight decomposition, including the sparse matrix and low-rank component. As shown in Table A1, the time and memory overhead of quantizing DGDMs is entirely acceptable, especially considering the significant improvement in inference speed, and does not require high-performance hardware.

Table A1. Peak memory requirements and running time of Bit-DGDM for quantizing DGDMs. The running time is broken down into (i) activation thresholds computation ($\mathbf{X}$ comp.) and (ii) ill-conditioned low rank weight decomposition ($\mathbf{W}$ deco.).

QM9 (Digress)			**CATH (GRADE-IF)**		
$\mathbf{X}$ Comp. (min)	$\mathbf{W}$ deco. (min)	Peak Memory (GB)	$\mathbf{X}$ Comp. (min)	$\mathbf{W}$ deco. (min)	Peak Memory (GB)
2.9	13.6	2.1	5.5	10.1	1.9

Table A2. Non-molecule generation performance of full-precision Digress (Vignac et al., 2023) (*i.e.,* baseline) and quantized Digress with different quantization methods on Stochastic Block Model (SBM) and Planar datasets.

Non-Molecule Generation		SBM					Planar graphs				
Method	**Precision**	**Deg.** (↓)	**Clus.** (↓)	**Orb.** (↓)	**Speedup** (↑)	**Mem.** (GB ↓)	**Deg.** (↓)	**Clus.** (↓)	**Orb.** (↓)	**Speedup** (↑)	**Mem.** (GB ↓)
baseline	FP32	1.6	1.5	1.7	1×	12.6	1.4	1.2	1.7	1×	10.4
	BF16	1.6	1.6	1.7	1.3×	6.8	1.4	1.2	1.7	1.3×	5.5
RTN	W4A4	15.6	3.4	2.8	2.9×	3.7	21.6	8.6	2231	2.8×	3.1
GPTQ	W4A4	4.6	2.7	3.2	2.9×	3.7	3.8	2.9	2.6	2.8×	3.1
SVDQuant	W4A4	1.9	2.1	1.9	2.1×	4.0	1.8	1.7	2.1	2.1×	3.5
DuQuant	W4A4	2.3	2.6	2.5	2.0×	4.1	3.8	3.6	2.4	2.0×	3.5
SqueezeLLM	W4A16	2.0	1.8	2.0	1×	5.3	1.7	1.5	2.2	1×	5.1
Bit-DGDM	W4A4	1.8	1.7	1.9	2.7×	4.5	1.5	1.3	1.9	2.6×	3.9

F.4. Graph Generation Results on Non-Molecule Graphs

In Table A2, we validate the graph generation performance of our quantization method, Bit-DGDM, compared to other quantization approaches on the non-molecule benchmark (Martinkus et al., 2022). The datasets used for evaluation include graphs from the Stochastic Block Model (SBM), where the training graphs are sampled from the stochastic block model (with up to 200 nodes per graph), and planar graphs, where each graph consists of 64 nodes. Compared to molecule graphs, these two datasets contain graphs of larger scales. The evaluation metrics include node degrees (Deg.), clustering coefficient (Clus.), and orbit count (Orb.) (Liao et al., 2019). As shown in Table A2, our proposed Bit-DGDM not only achieves faster inference speed than most quantization baselines but also generates higher-quality graphs. This demonstrates the generalization capability of our method for generating larger-scale graphs.

F.5. Kernel Efficiency Analysis

The analysis of Table A3 reveals distinct suitability of the Equidistributed and Adaptive kernels for handling the sparse components of weights ($\mathbf{S}_W$) and activations ($\mathbf{S}_A$). The Equidistributed kernel is particularly effective for processing $\mathbf{S}_W$, as the α-sparsity constraint align well with its balanced thread allocation strategy. In contrast, the Adaptive kernel is better suited for $\mathbf{S}_A$, as its dynamic resource allocation can efficiently handle the irregular sparsity typically found in activations. The optimal configuration combines the Equidistributed kernel for $\mathbf{S}_W$ and the Adaptive kernel for $\mathbf{S}_A$, leveraging their respective strengths to maximize computational efficiency.

F.6. Rank Selection Effect

The selection of rank is based a systematic trade-off analysis between computational efficacy and precision. Through experiments on CATH dataset (as shown in Table A4), we observed that low ranks (rank=8) led to significant degradation in graph generation quality, and higher ranks (rank=32) provided marginal quality improvements while incurring substantial latency overhead.

Table A3. A comparative study of MOSES dataset on the Speedup (compared with FP32) of different kernels applied to the sparse components of activations and weights, *i.e.*, $\mathbf{S}_A$ and $\mathbf{S}_W$.

Speedup		$\mathbf{S}_W$	
		Equidistributed	Adaptive
$\mathbf{S}_A$	Equidistributed	2.3×	2.0×
	Adaptive	2.5×	2.2×

Table A4. The rank selection of effect evaluation on CATH datasets. The Speedup is compared with the FP32 baseline.

	Perplexity	**Recovery (%)**	**Speedup**	**Mem. (GB)**
Rank=8	5.1	46.5	2.6	4.7
Rank=16	4.5	51.6	2.5	4.9
Rank=32	4.5	51.8	2.2	5.4