IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 April 2025, accepted 25 May 2025, date of publication 6 June 2025, date of current version 13 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3576737

RESEARCH ARTICLE

COMQ: A Backpropagation-Free Algorithm for Post-Training Quantization

AOZHONG ZHANG[1], ZI YANG[1], NAIGANG WANG[2], YINGYONG QI[3], JACK XIN[3], XIN LI[4], (Fellow, IEEE), AND PENGHANG YIN[1]

[1]Department of Mathematics and Statistics, State University of New York at Albany, Albany, NY 12222, USA
[2]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA
[3]Department of Mathematics, University of California at Irvine, Irvine, CA 92617, USA
[4]Department of Computer Science, State University of New York at Albany, Albany, NY 12222, USA

Corresponding author: Aozhong Zhang (azhang3@albany.edu)

This work was supported in part by NSF under Grant DMS-2208126, Grant IIS-2110546, Grant DMS-2219904, Grant DMS-2151225, Grant DMS2219904, and Grant CCSS-2348046; in part by the Start-Up Grant from SUNY Albany; in part by SUNY-IBM AI Collaborative Research Grant; and in part by the Qualcomm Faculty Gift Award. The work of Penghang Yin and Aozhong Zhang was supported by in part by NSF under Grant DMS-2208126 and Grant IIS-2110546 and in part by the SUNY-IBM AI Research Alliance Grant. The work of Zi Yang was supported in part by NSF under Grant DMS-2110836 and in part by the Start-Up Grant from SUNY Albany. The work of Jack Xin was supported in part by NSF under Grant DMS-2151225 and Grant DMS-2219904 and in part by the Qualcomm Faculty Gift Award. The work of Xin Li was supported by NSF under Grant CCSS-2348046.

ABSTRACT Post-training quantization (PTQ) has emerged as a practical approach to compress large neural networks, making them highly efficient for deployment. However, effectively reducing these models to their low-bit counterparts without compromising the original accuracy remains a key challenge. In this paper, we propose an innovative PTQ algorithm termed COMQ, which sequentially conducts **co**ordinate-wise **m**inimization of the layer-wise reconstruction errors. We consider the widely used integer quantization, where every quantized weight can be decomposed into a shared floating-point scalar and an integer bit-code. Within a fixed layer, COMQ treats all the scaling factor(s) and bit-codes as the variables of the reconstruction error. Every iteration improves this error along a single coordinate while keeping all other variables constant. COMQ is easy to use and requires no hyper-parameter tuning. It instead involves only dot products and rounding operations. We update these variables in a carefully designed greedy order, significantly enhancing the accuracy. COMQ achieves remarkable results in quantizing 4-bit Vision Transformers, with a negligible loss of less than 1% in Top-1 accuracy. In 4-bit INT quantization of convolutional neural networks, COMQ maintains near-lossless accuracy with a minimal drop of merely 0.3% in Top-1 accuracy. The code is available at https://github.com/AozhongZhang/COMQ

INDEX TERMS Post-training quantization, coordinate descent, layer-wise reconstruction, optimization.

I. INTRODUCTION

Over the past decade, deep learning has enjoyed remarkable success in a wide range of fields [1], [2], [3], [4], [5], [6], [7]. Deep neural networks (DNNs) are scaled to unprecedented sizes for better performance, resulting in models with billions or even trillions of parameters. With the increasing demand to deploy these models on resources-constrained devices, substantial efforts have been dedicated to advancing model compression techniques [8], [9], [10], [11], [12], [13] to train lightweight DNNs without sacrificing achievable accuracy. Among these techniques, quantization has recently garnered significant attention. Quantization techniques involve representing the weights and activations of DNNs with low precision using just a few bits (e.g., 4-bit). Quantized DNNs offer a substantially reduced memory footprint and rely on fixed-point or integer arithmetic during inference, leading to accelerated and efficient deployment.

Quantization methods can be categorized into two primary categories: Quantization Aware Training (QAT) [11], [13], [14], [15], [16], [17], [18], [19], [20], [21] and Post-Training

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Huang.

Quantization (PTQ) [22], [23], [24], [25], [26], [27], [28]. In general terms, QAT is designed to globally minimize the conventional training loss of the model for quantization parameters. It involves tackling a formidable nonconvex minimization problem with a discrete nature. QAT requires an all-encompassing training pipeline and computational cost that is at least on par with regular full-precision models. In contrast, PTQ directly applies low-precision calibration to a pre-trained full-precision model. Computationally, PTQ aims to identify an optimal quantized model locally by minimizing a simplified surrogate loss, so it enjoys significantly reduced algorithmic complexity and appears to be a faster and more resource-efficient process. On the downside, PTQ suffers a heavier performance degradation than QAT, especially when it comes to low-bit quantization of Vision Transformers (ViTs), which has received much recent attention [29], [30], [31], [32].

A. MOTIVATION

Unlike QAT, PTQ enjoys the benefits of resource efficiency. The downside of PTQ includes potential accuracy drop, sensitivity to model and data distribution, and limited flexibility in precision levels. Therefore, it is desirable to pursue some middle ground between QAT and PTQ by preserving the cost benefits of PTQ without sacrificing the accuracy. Optimization theory offers a rich weaponry for striking an improved tradeoff between cost and performance in many vision systems. For model compression, a greedy path-following mechanism was developed for PTQ of neural networks with provable guarantees in [33]; a novel bit-split optimization approach was developed in [34] to achieve minimal accuracy degradation based on the analysis of the quantization loss in the final output layer. Inspired by these recent advances, we advocate an optimization-based approach to PTQ based on coordinate descent [35], [36], [37] that minimizes the objective functions along the coordinate directions iteratively.

B. CONTRIBUTION

In this work, we introduce COMQ, a post-training quantization method that performs uniform weight quantization on a layer-by-layer basis (see Figure 1). Similarly to existing works [23], [24], [26], [33], our main objective is to minimize the layerwise squared error $\|\boldsymbol{X}\boldsymbol{W}_q - \boldsymbol{X}\boldsymbol{W}\|^2$ with respect to quantized weights $\boldsymbol{W}_q$. To efficiently address optimization, COMQ enforces the decomposition $\boldsymbol{W}_q = \delta \cdot \boldsymbol{Q}$ with δ being the full-precision scalar(s) and $\boldsymbol{Q}$ storing the integer bit-codes, and it then minimizes this error over the new variables through a coordinate-wise minimization procedure. Adhering to a greedy selection rule, we select one variable at a time, whether scaling factor or bit-code, to update while maintaining the others at their most recent states. Unlike recent works [22], [23], [24] that require back-propagation or estimation of the Hessian inverse to minimize the reconstruction error, COMQ solves a sequence of minimization of *univariate quadratic functions* which enjoy closed-form minimizers. This leads to backpropagation-free iterations that primarily involve dot products and rounding operations without using any hyperparameters.

We detail the implementation of COMQ for per-layer and per-channel weight quantization, respectively. Our empirical results demonstrate that the proposed selection order of variables can enhance the performance at extremely low bit-widths, outperforming the standard index-based update order (the so-called cyclic order). Our experiments show that the proposed COMQ method achieves state-of-the-art performance on convolutional neural networks and Vision Transformers. Specifically, our 4-bit CNN almost reaches the accuracy of full-precision models with less than 0.05% accuracy loss, and 4-bit ViT reaches less than 1% accuracy loss. Moreover, our approach outperforms the existing state-of-the-art PTQ methods on CNNs with per-layer quantization and ViTs with per-channel quantization. Although our main focus is on weight quantization, we remark that the proposed framework exhibits versatility, allowing seamless extension to full quantization tasks by incorporating existing activation quantization techniques, especially that for quantizing ViTs such as [38].

C. NOTATIONS

Throughout this paper, we denote vectors with bold small letters and matrices with bold capital ones. For any two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^n x_i y_i$ is their inner product. We denote by $\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ the Euclidean norm and denote by $\|\boldsymbol{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$ the l_∞ norm. Similarly for two matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{m\times n}$, the inner product is given by $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle := \sum_{i=1}^m \sum_{j=1}^n X_{i,j} Y_{i,j}$, and $\|\boldsymbol{X}\| := \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle}$ is the Frobenius norm. Moreover, we denote the outer product of two vectors by $\boldsymbol{x} \otimes \boldsymbol{y} := \boldsymbol{x}\boldsymbol{y}^\top \in \mathbb{R}^{n\times n}$. $\boldsymbol{x} \odot \boldsymbol{y} := (x_1 y_1, \ldots, x_n y_n) \in \mathbb{R}^n$ denotes the Hadamard (or element-wise) product, and $\boldsymbol{x} \oslash \boldsymbol{y} := (x_1/y_1, \ldots, x_n/y_n) \in \mathbb{R}^n$ denotes the Hadamard division. Finally, for any positive integer n, $[n] := \{1, \ldots, n\}$ denotes the set of integers up to n.

II. RELATED WORKS

A. POST-TRAINING QUANTIZATION

The earlier PTQ work, DFQ [39], is a data-free approach that operates independently of any training data. It involves minimizing the expected error between outputs of the corresponding linear layers in both the pre-trained and quantized models over the inputs. DFQ employs pre-processing steps to appropriately re-scale the weights and biases across the layers before the quantization. DFQ relies on prior information about the mean of layer inputs, a value that can be estimated from batch normalization parameters to rectify the noises introduced during the quantization process. DFQ is proven to work well on INT8 quantization but suffers noticeable accuracy degradation at lower bit-widths. Cai et al. [40] introduce ZeroQ, which distills the input data distribution that matches the statistics in the model's

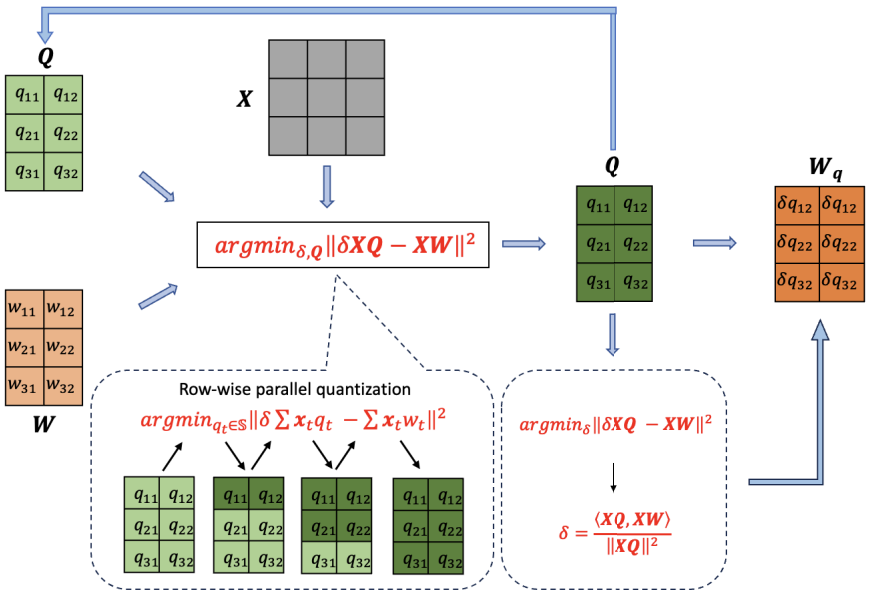


FIGURE 1. The workflow of COMQ for per-layer quantization.

batch normalization layers. Xu et al. [41] propose to use a generative model to construct fake data for accuracy improvement. Subsequent research efforts employ a set of calibration data and perform layer-by-layer or block-by-block quantization. The general idea is to minimize layer-wise [23], [24], [26], [33], [42] or block-wise [22] squared error between outputs of pre-trained and quantized models with respect to the associated quantized weights. To carry out the minimization procedure, a class of PTQ methods such as AdaRound [23], BRECQ [22], QDrop [27], adopt back-propagation to minimize the layer-wise or block-wise quantization error. AdaQuant [26], [42] proposed to solve integer programming for bit-allocation while jointly learning a scaling factor.

Some initiatives have been undertaken to develop backpropagation-free algorithms for weight quantization. OBC [24], also known as OPTQ [43] for quantizing large language models (LLMs), adapts the Optimal Brain Surgeon framework to the setting of layer-wise PTQ. OBC quantizes one weight using an analytical formula at a time and updates all remaining weights after each step. Compared with OBC, our proposed COMQ method is simpler because it only updates one parameter in each step and keeps the others untouched. Additionally, unlike OBC, COMQ does not require inverting the Hessian matrix of the layer-wise squared error in quantizing the weight. Zhang et al. [33] introduced GPFQ to efficiently learn layer-wise bit-codes sequentially, assuming that the floating scaling factors are already learned. In its practical implementation, determining the appropriate scalars requires a trial-and-error process. In parallel to our research, Behdin et al. explored a cyclic coordinate descent approach, QuantEase [44], for PTQ. While their work focuses on LLMs, our research targets ViTs with a different algorithmic implementation. Specifically, we propose a greedy update order and introduce learnable scaling factors, which we believe offer unique benefits in terms of performance.

Developing tailored PTQ methods for ViTs while maintaining good performance poses a significant challenge and has garnered considerable attention [38], [45], [46], [47]. For examples, PTQ4ViT [45] proposes twin uniform quantization to cope with the unbalanced distributions of activation values and a Hessianguided metric to search for scaling factors. APQ-ViT [46] proposes a calibration scheme that perceives the overall quantization disturbance block-wise. FQ-ViT [47] introduces Powers-of-Two scale and Log-Int-Softmax quantizers for the activation quantization. RepQ-ViT [38] proposes scale reparameterization methods for post-LayerNorm and post-Softmax activations, and improves the accuracy of 4-bit PTQ of ViTs to a usable level. More recently, PTQ methods have also been applied to the emerging diffusion models [48], [49] and large language models [43], [50].

B. COORDINATE DESCENT

The coordinate descent method is a simple yet effective optimization algorithm widely applied to large-scale problems [35], [36], [37]. The algorithm minimizes the objective functions along the coordinate directions iteratively. For the objective function $f(x_1, \ldots, x_d)$, the coordinate descent method starts with the initial value $(x_1^0, \ldots, x_n^0)$. In the k-th iteration, the method sequentially solves the following

problem for $i = 1, \ldots, n$

$$x_i^k := \arg\min_{x_i} f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^{k-1}, \ldots, x_n^{k-1}).$$

In the PTQ problem, each quantized weight is represented as $\boldsymbol{W}_q = \delta \cdot \boldsymbol{Q}$. We consider the scale δ and all elements in $\boldsymbol{Q}$ as coordinates. In one step, we update one element in $\boldsymbol{Q}$ or the scale δ while fixing all other coordinates. Each subproblem in the coordinate descent method is a univariate optimization problem and has a closed-form solution in our settings. The order of solving subproblems affects the optimization performance. Cyclic order is the most widely used, cyclically iterating through the coordinates. However, the minimization problem for quantization is non-convex and exhibits a discrete nature, challenging the optimality of cyclic order. In this work, we propose to update the parameters in a carefully designed order to achieve a reduced quantization loss compared to that of cyclic order [34].

III. PROPOSED METHOD

This section presents our proposed COMQ method for post-training quantization. Our discussions mainly focus on linear layers for simplicity. A convolutional layer can be equivalently converted to a linear layer, hence the proposed method can be also applied. Regarding transformers [51], it is natural to conceptualize the key, query, and value components of a self-attention layer as three separate linear layers.

We consider a linear layer with matrix weight $\boldsymbol{W}$ and matrix input $\boldsymbol{X}$. For the weight $\boldsymbol{W}$, we aim to find the quantized weights $\boldsymbol{W}_q$ that minimize the following function

$$\min_{\boldsymbol{W}_q \in \mathcal{W}} \|\boldsymbol{X}\boldsymbol{W}_q - \boldsymbol{X}\boldsymbol{W}\|^2, \tag{1}$$

where $\mathcal{W}$ is an appropriate set of all feasible quantized weights. The matrix input $\boldsymbol{X}$ is the feature generated from the pre-trained model and does not depend on the quantized weights from the previous layer. We propose to solve the problem (1) by the coordinate descent algorithm. We regard all elements in the quantized weight and the scale factor as coordinates. In each step, we only solve a univariate optimization problem with respect to only one coordinate. By iteratively computing the quantized weights and corresponding scale factors, we finally find a proper quantized weight $\boldsymbol{W}_q$ that has a small quantization error and maintains good accuracy.

In this section, we will show our COMQ method for two quantization scenarios: per-layer quantization and per-channel quantization. All quantized weights with in the same layer share a common scale factor for the per-layer quantization, while different columns of the quantized weight use different scale factors for the per-channel quantization. Throughout the paper, we consider the b-bit asymmetric uniform quantization [52], which takes the bit-code set of $\mathbb{S} = \{z, z+1, \ldots, z+2^b-1\}$, with z being the so-called zero point.

A. PER-LAYER QUANTIZATION

In this subsection, we present our COMQ method for b-bit per-layer quantization, where the whole quantized weight matrix $\boldsymbol{W}_q$ shares the scale factor δ. In this setting, the quantized weight $\boldsymbol{W}_q$ can be decomposed as

$$\boldsymbol{W}_q = \delta \cdot \boldsymbol{Q} \in \mathbb{R}^{m\times n},$$

where $\boldsymbol{Q} \in \mathbb{S}^{m\times n}$ is the integer bit-code matrix for $\boldsymbol{W}_q$. The optimization problem (1) reduces to

$$\min_{\delta\in\mathbb{R}, \boldsymbol{Q}\in\mathbb{S}^{m\times n}} \|\delta\boldsymbol{X}\boldsymbol{Q} - \boldsymbol{X}\boldsymbol{W}\|^2. \tag{2}$$

It holds that $\boldsymbol{X}\boldsymbol{Q} = (\boldsymbol{X}\boldsymbol{q}_1, \ldots, \boldsymbol{X}\boldsymbol{q}_n)$ and $\boldsymbol{X}\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{w}_1, \ldots, \boldsymbol{X}\boldsymbol{w}_n)$, where $\boldsymbol{q}_j$ and $\boldsymbol{w}_j$ are the j-th column of $\boldsymbol{Q}$ and $\boldsymbol{W}$ respectively. Therefore, we can rewrite the problem (2) as

$$\min_{\delta, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_n} \sum_{j=1}^{n} \|\delta\boldsymbol{X}\boldsymbol{q}_j - \boldsymbol{X}\boldsymbol{w}_j\|^2. \tag{3}$$

When fixing the scale factor δ, the terms $\|\delta\boldsymbol{X}\boldsymbol{q}_j - \boldsymbol{X}\boldsymbol{w}_j\|^2$ for $j = 1, \ldots, n$ are *independent* of each other. Hence, we can update the columns $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ *in parallel*. We apply the coordinate descent method to solve the above problem (2). Specifically, we regard the scaling factor δ and all elements in $\boldsymbol{Q}$ as individual coordinates. In each iteration, we can update the bit-codes across all columns in the same row, i.e., row-wise update, before updating δ:

$$\{Q_{1,j}\}_{j\in[n]} \Rightarrow \{Q_{2,j}\}_{j\in[n]} \Rightarrow \cdots \Rightarrow \{Q_{m,j}\}_{j\in[n]} \Rightarrow \delta. \tag{4}$$

1) INITIALIZATION

At the beginning of our COMQ method, the scale factor δ^0 should be properly initialized to capture the range of the weight matrix $\boldsymbol{W}$. Typically, the maximum value of $|\boldsymbol{W}|$ is the default selection in most quantization problems. However, this choice overlooks the impact of outliers in the weights on the overall quantization error. In order to smooth out these outliers, we consider the average infinity norm of weights across all columns $\boldsymbol{W}$. For the b bits uniform quantization, we use the initialization $\delta^0 = \frac{1}{2^{b-1}} \frac{\sum_{1\le i\le n} \|\boldsymbol{w}_i\|_\infty}{n}$. Then the matrix $\boldsymbol{Q}$ is initialized as $\boldsymbol{Q}^0 = \frac{\boldsymbol{W}}{\delta^0}$. Note that the initialization $\boldsymbol{Q}^0$ above is not an actual bit-code matrix as $\boldsymbol{W}$ is float, but it will become feasible after the 1st iteration as will be shown below.

2) Q-UPDATE

Let $\boldsymbol{Q}^{k-1}, \delta^{k-1}$ be the parameters after $k-1$ iterations. Without loss of generality, we focus on updating an arbitrary column of $\boldsymbol{Q}$, denoted by $\boldsymbol{q}$, where the column index j is omitted for notational simplicity. Suppose the coordinates q_t^k for $1 \le t \le i-1$, have been updated in the k-th iteration. Note that $\boldsymbol{X}\boldsymbol{q} = \sum_{t=1}^{m} q_t \boldsymbol{x}_t$, with $\boldsymbol{x}_t$ being the t-th column of $\boldsymbol{X}$. Coordinate descent calls for solving the following problem

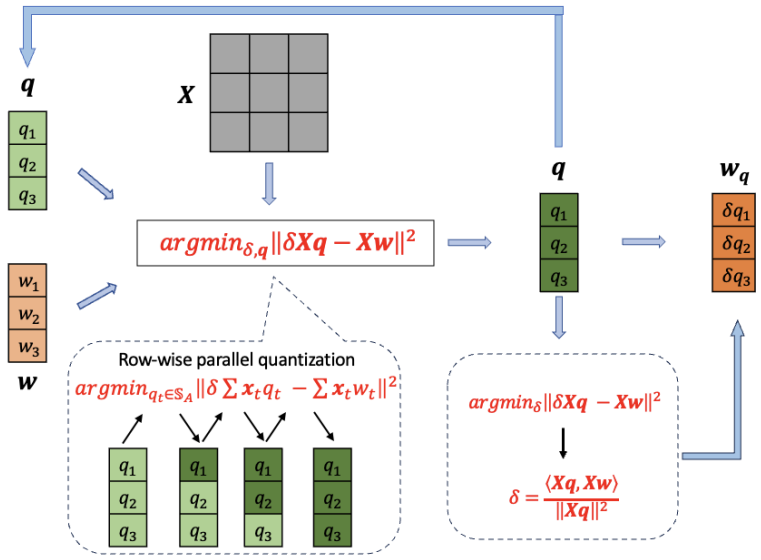


FIGURE 2. The workflow of COMQ for per-channel quantization.

to obtain q_i^k:

$$\begin{aligned} & q_i^k \\ &= \arg\min_{q_i \in \mathbb{S}} \left\| \delta^{k-1} \left(q_i \boldsymbol{x}_i + \sum_{t=1}^{i-1} q_t^k \boldsymbol{x}_t + \sum_{t=i+1}^{m} q_t^{k-1} \boldsymbol{x}_t \right) - \boldsymbol{X}\boldsymbol{w} \right\|^2 \\ &= \arg\min_{q_i \in \mathbb{S}} \left\| q_i \delta^{k-1} \boldsymbol{x}_i - \boldsymbol{s}_i^k \right\|^2, \end{aligned} \tag{5}$$

where $\boldsymbol{s}_i^k = \boldsymbol{X}\boldsymbol{w} - \delta^{k-1}(\sum_{t=1}^{i-1} q_t^k \boldsymbol{x}_t + \sum_{t=i+1}^{m} q_t^{k-1} \boldsymbol{x}_t)$ is a constant vector. (5) is a quadratic minimization problem if the variable q_t is in the continuous domain. The global optimizer of (5) over $\mathbb{R}$ is $\frac{\langle \delta^{k-1} \boldsymbol{x}_i, \boldsymbol{s}_i^k \rangle}{\|\delta^{k-1} \boldsymbol{x}_i\|^2}$. Therefore, we obtain the updated coordinate q_i^k as

$$q_i^k = \text{clip}\left(\left\lfloor \frac{\langle \boldsymbol{x}_i, \boldsymbol{s}_i^k \rangle}{\delta^{k-1} \|\boldsymbol{x}_i\|^2} \right\rceil, z, z + 2^b - 1 \right).$$

Hereby we define the quantization residuals at the i-th step (for the column):

$$\begin{aligned} \boldsymbol{u}_i^k &:= \sum_{t=1}^{i-1} (w_t - \delta^{k-1} q_t^k) \boldsymbol{x}_t + \sum_{t=i}^{m} (w_t - \delta^{k-1} q_t^{k-1}) \boldsymbol{x}_t \\ &= \boldsymbol{s}_i^k - \delta^{k-1} q_i^{k-1}. \end{aligned} \tag{6}$$

To efficiently implement the $\boldsymbol{Q}$-update as in (4), we take advantage of vectorized operations to update $\{Q_{i,j}\}_{j \in [n]}$ across all columns for each i. To this end, we denote by $\boldsymbol{w}_{i,:} \in \mathbb{R}^n$ and $\boldsymbol{q}_{i,:} \in \mathbb{R}^n$ the i-th row of $\boldsymbol{W}$ and $\boldsymbol{Q}$, respectively, and denote by $\boldsymbol{x}_{:,i} \in \mathbb{R}^m$ the i-th column of $\boldsymbol{X}$. Then the vectorized update of $\{Q_{i,j}^k\}_{j \in [n]}$ or $\boldsymbol{q}_{i,:}^k$ for the k-th iteration of COMQ proceeds as follows: with $\boldsymbol{U}_0^k = \boldsymbol{X}(\boldsymbol{W} - \delta^{k-1}\boldsymbol{Q}^{k-1})$, we iterate for $i = 1, \ldots, m$:

$$\begin{aligned} \boldsymbol{U}_i^k &= \boldsymbol{U}_{i-1}^k - \boldsymbol{x}_{:,i} \otimes (\boldsymbol{w}_{i,:} - \delta^{k-1} \boldsymbol{q}_{i,:}^{k-1}) \\ \boldsymbol{q}_{i,:}^k &= \text{clip}\left(\left\lfloor \frac{(\boldsymbol{U}_i^k + \boldsymbol{x}_{:,i} \otimes \boldsymbol{w}_{i,:})^\top \boldsymbol{x}_{:,i}}{\delta^{k-1} \boldsymbol{x}_{:,i}^\top \boldsymbol{x}_{:,i}} \right\rceil, z, z + 2^b - 1 \right) \\ \boldsymbol{U}_i^k &= \boldsymbol{U}_i^k + \boldsymbol{x}_{:,i} \otimes (\boldsymbol{w}_{i,:} - \delta^{k-1} \boldsymbol{q}_{i,:}^k). \end{aligned} \tag{7}$$

Here $\boldsymbol{U}_i^k$ maintains the quantization residuals across all columns.

3) δ-UPDATE

After obtaining the new bit-code matrix $\boldsymbol{Q}^k$, the scale factor can be updated by solving the following problem

$$\min_{\delta} \|\delta \boldsymbol{X}\boldsymbol{Q}^k - \boldsymbol{X}\boldsymbol{W}\|^2.$$

It is a standard convex quadratic optimization problem and has a closed-form solution

$$\delta^k = \frac{\langle \boldsymbol{X}\boldsymbol{Q}^k, \boldsymbol{X}\boldsymbol{W} \rangle}{\|\boldsymbol{X}\boldsymbol{Q}^k\|^2}. \tag{8}$$

We summarize the COMQ algorithm for linear layers in Alg. 1. For a general neural network with multiple linear layers, we can apply the COMQ method in Alg. 1 to each linear layer to obtain the quantized neural network.

Algorithm 1 COMQ for the per-Layer Quantization of One Linear Layer

Input: Pre-trained weights $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, feature matrix $\boldsymbol{X}$, and iteration number K.

1: **Initialize** δ^0 and $\boldsymbol{Q}^0$
2: **for** $k = 1, \ldots, K$ **do**
3: **for** $i = 1, \ldots, m$ **do**
4: Update the coordinates $\{Q^k_{i,j}\}_{j\in[n]}$ or $\boldsymbol{q}^k_{i,:}$ as in (7)
5: **end for**
6: Update the scaling factor δ^k as in (8)
7: **end for**
8: Let $\boldsymbol{W}_q = \delta^K \boldsymbol{Q}^K$

Output: Quantized weight $\boldsymbol{W}_q$.

B. PER-CHANNEL QUANTIZATION

Per-layer quantization uses the same scale factor for the whole weight matrix. However, the different columns of weight matrix may have very different value ranges, and this causes a large quantization error. For the per-channel quantization, each column has its own scale factor, which yields smaller quantization errors and less accuracy drop.

The per-channel quantization aims to quantize the j-th column $\boldsymbol{w}_i$ of $\boldsymbol{W}$ to $\delta_j \boldsymbol{q}_j$, where $\boldsymbol{q}_j$ is bit-code vector. For the weight matrix $\boldsymbol{W}$, the quantized weight $\boldsymbol{W}_q \in \mathbb{R}^{m\times n}$ in per-channel quantization format is

$$\boldsymbol{W}_q = \boldsymbol{Q}\mathrm{diag}(\delta) = \boldsymbol{Q}\mathrm{diag}(\delta_1, \ldots, \delta_n) = (\delta_1 \boldsymbol{q}_1, \ldots, \delta_n \boldsymbol{q}_n).$$

Same as for per-layer quantization, the j-th column of $\boldsymbol{W}$ can be quantized independently by solving the following optimization problem:

$$\delta_j, \boldsymbol{q}_j = \arg\min_{\delta,\boldsymbol{q}} \|\delta \boldsymbol{X}\boldsymbol{q} - \boldsymbol{X}\boldsymbol{w}_j\|^2,\ j \in [n]. \tag{9}$$

1) INITIALIZATION

The scale factor is initialized as $\delta^0_j = \lambda \frac{\max(\boldsymbol{w}_j) - \min(\boldsymbol{w}_j)}{2^b - 1}$ for some $0 < \lambda \leq 1$. The λ is to ensure we do not quantize most values to zero, especially for ultra-low bit quantization. The $\boldsymbol{q}^0_j$ is initialized as $\frac{\boldsymbol{w}_j}{\delta^0_j}$.

2) Q-UPDATE

Let $\delta^{k-1} \in \mathbb{R}^n$ and $\boldsymbol{Q}^{k-1} \in \mathbb{S}^{m\times n}$ be the scaling factors and bit-code matrix produced by $(k-1)$-th iteration. Or equivalently, suppose $\boldsymbol{W}^{k-1}_q = \boldsymbol{Q}^{k-1}\mathrm{diag}(\delta^{k-1})$ is the current quantized weight matrix. The $\boldsymbol{Q}$-Update for per-channel quantization is substantially similar to the per-layer setting. The updated bit-code $Q^k_{i,j}$ is given by

$$Q^k_{i,j} = \mathrm{clip}\left(\left\lfloor \frac{\left\langle \boldsymbol{x}_i, \boldsymbol{s}^k_{i,j}\right\rangle}{\delta^{k-1}_j \|\boldsymbol{x}_i\|^2} \right\rceil, z_j, z_j + 2^b - 1\right),$$

where $\boldsymbol{s}^k_{i,j} = \boldsymbol{X}\boldsymbol{w}_j - \delta^{k-1}_j(\sum_{t=1}^{i-1} Q^k_{t,j}\boldsymbol{x}_t + \sum_{t=i+1}^{m} Q^{k-1}_{t,j}\boldsymbol{x}_t)$, and z_j is the zero-point for quantizing the j-th column. For the row-wise update of $\boldsymbol{Q}^k$, we simply follow (7) with minor adaptions to the column-wise scaling. Using the same notations, we denote by $\boldsymbol{w}_{i,:} \in \mathbb{R}^n$ and $\boldsymbol{q}_{i,:} \in \mathbb{R}^n$ the i-th row of $\boldsymbol{W}$ and $\boldsymbol{Q}$, respectively, and denote by $\boldsymbol{x}_{:,i} \in \mathbb{R}^m$ the i-th column of $\boldsymbol{X}$. With $\boldsymbol{U}^k_0 = \boldsymbol{X}(\boldsymbol{W} - \boldsymbol{W}^{k-1}_q)$, we iterate for $i = 1, \ldots, m$:

$$\begin{aligned}
&\boldsymbol{U}^k_i \\
&= \boldsymbol{U}^k_{i-1} - \boldsymbol{x}_{:,i} \otimes (\boldsymbol{w}_{i,:} - \delta^{k-1} \odot \boldsymbol{q}^{k-1}_{i,:}) \\
&\boldsymbol{q}^k_{i,:} \\
&= \mathrm{clip}\left(\left\lfloor \frac{(\boldsymbol{U}^k_i + \boldsymbol{x}_{:,i} \otimes \boldsymbol{w}_{i,:})^\top \boldsymbol{x}_{:,i}}{\boldsymbol{x}^\top_{:,i}\boldsymbol{x}_{:,i}} \oslash \delta^{k-1} \right\rceil, z, z + 2^b - 1\right) \\
&\boldsymbol{U}^k_i \\
&= \boldsymbol{U}^k_i + \boldsymbol{x}_{:,i} \otimes (\boldsymbol{w}_{i,:} - \delta^{k-1} \odot \boldsymbol{q}^k_{i,:}),
\end{aligned} \tag{10}$$

where $\boldsymbol{U}^k_i$ maintains the quantization residuals and z is the vector of zero-points across all columns.

3) δ-UPDATE

Having obtained $\boldsymbol{Q}^k$, we update the scaling factors as

$$\delta^k_j = \frac{\left\langle \boldsymbol{X}\boldsymbol{q}^k_j, \boldsymbol{X}\boldsymbol{w}_j\right\rangle}{\|\boldsymbol{X}\boldsymbol{q}^k_j\|^2}. \tag{11}$$

To summarize, the workflow for per-channel quantization is depicted in Fig. 2, and Alg. 2 describes COMQ for layer-wise PTQ with per-channel scaling.

Algorithm 2 COMQ for the per-Channel Quantization of One Linear Layer

Input: Pre-trained weights $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, feature matirx $\boldsymbol{X}$, and iteration number K.

1: **Initialize** δ^0, $\boldsymbol{Q}^0$ as described in Section III-B
2: **for** $k = 1, \ldots, K$ **do**
3: **for** $i = 1, \ldots, m$ **do**
4: Update the coordinates $\{Q^k_{i,j}\}_{j\in[n]}$ or $\boldsymbol{q}^k_{i,:}$ as in (10)
5: **end for**
6: Update the scaling factors δ^k as in (11)
7: **end for**
8: Compute $\boldsymbol{W}_q = (\delta^K_1 \boldsymbol{q}^K_1, \ldots, \delta^K_n \boldsymbol{q}^K_n)$

Output: Quantized weight $\boldsymbol{W}_q$.

C. GREEDY COMQ

Coordinate descent update the coordinates of a vector in the cyclic order (or index order) by default, which may not be optimal. To further reduce the quantization error and improve the performance of the quantized model, we propose a greedy update rule to determine the update order of the coordinates. The quantization error for a column with weight vector $\boldsymbol{w}$ can be written as $\|\delta\boldsymbol{X}\boldsymbol{q} - \boldsymbol{X}\boldsymbol{w}\|^2 = \left\|\sum_{i=1}^m \delta q_i \boldsymbol{x}_i - w_i\boldsymbol{x}_i\right\|^2$. The importance of quantization target $w_i\boldsymbol{x}_i$ can be heuristically measured by its magnitude. Hence, we choose to first update the coordinates with a larger magnitude. This order ensures that the most significant coordinates are first updated, and we have a sufficient quantization error decrease for each step. In practice, we sort $\|w_1\boldsymbol{x}_1\|, \ldots, \|w_m\boldsymbol{x}_m\|$ in descending order

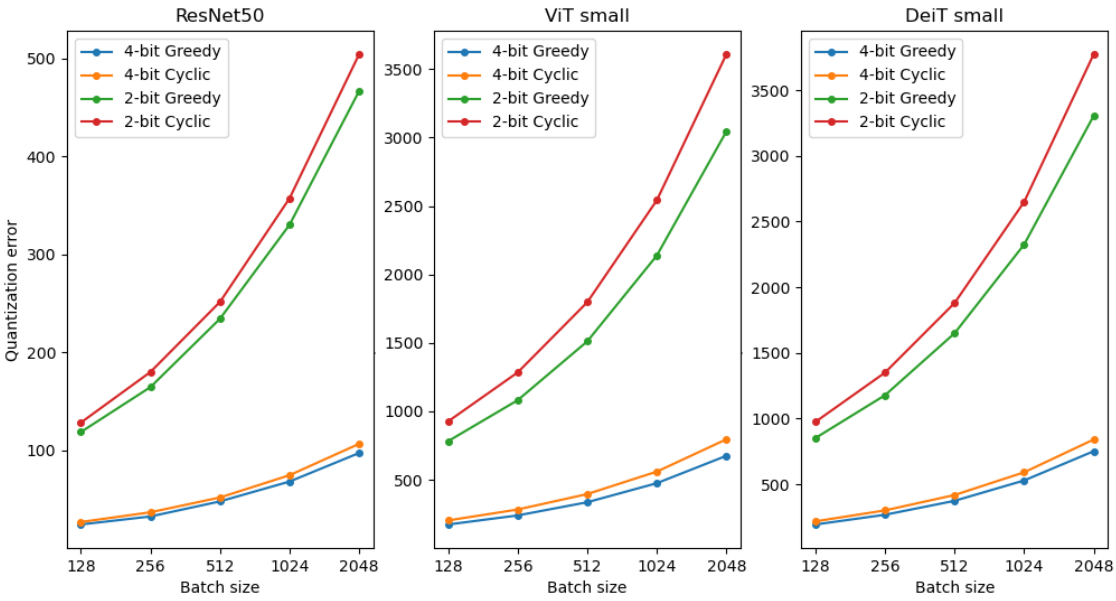


FIGURE 3. Comparisons of layer-wise quantization errors for cyclic and greedy orders.

to get the coordinate update order $i_1, \ldots, i_m$, permuting the index set $[m]$. Therefore, the greedy COMQ for quantizing the weight matrix $\boldsymbol{W}$ requires column-wise sorting of $(\boldsymbol{v} \odot |\boldsymbol{w}_1|, \ldots, \boldsymbol{v} \odot |\boldsymbol{w}_n|)$, where $\boldsymbol{v} = (\|\boldsymbol{x}_1\|, \ldots, \|\boldsymbol{x}_m\|)^\top$ is the column-wise Euclidean norm of $\boldsymbol{X}$ in descending order. Then, we permute the columns of $\boldsymbol{W}$ and rows of $\boldsymbol{X}$ individually according to the sorted indices, followed by the quantization process. After that, the inverse permutations are performed on $\boldsymbol{W}$ and $\boldsymbol{X}$ to restore the original index order.

To demonstrate the superiority of our proposed greedy order over the cyclic order, we compared the empirical layer-wise quantization errors $\|\boldsymbol{X}\boldsymbol{W}_q - \boldsymbol{X}\boldsymbol{W}\|$ across different architectures in Fig. 3. Additionally, we also compared their performance on ImageNet in Tab. 8 in the Appendix.

IV. EXPERIMENTS

In this part, we demonstrate the superiority of our method on various neural architectures, including ResNets [53], MobileNetV2 [54], and ViTs [29]. Sec. IV-A provides the implementation details. In Sec. IV-B, we compare the proposed COMQ method with the state-of-the-art PTQ methods. In Sec. IV-C, we conducted extensive ablation studies to comprehensively analyze the properties of our algorithm, including the influence of various batch sizes, the number of iterations and the runtime of different batch sizes.

A. IMPLEMENTATION DETAILS

We validate the performance of COMQ on the ImageNet [55] dataset. Evaluation metrics are the precision of the top-1 and top-5 quantized models in the validation data set. With the pre-trained float models from PyTorch and BRECQ [22], we quantize various CNNs on ImageNet including ResNet18 [53], ResNet50 [53], MobileNetV2 [54] and Vision Transformers including ViT [29], DeiT [30] and Swin [31]. In addition to vision models, we further validate the generality of COMQ on large language models (LLMs), including the OPT [56] family (from 2.7B to 13B) and the LLaMA2 [57] family (7B and 13B). For ImageNet experiments, we adopt a batch size of 2048 for ResNet-18, ResNet-50, and DeiT, and a batch size of 1024 for MobileNetV2, ViTs, and Swin. For language model evaluation, we sample 128 random segments, each containing 2048 tokens, from the WikiText2 [58] dataset. We ran our experiments on an Nvidia RTX 3090 GPU with 24G GPU memory for convolutional neural nets and on an Nvidia A40 GPU with 48G GPU memory for ViTs and LLMs.

1) THE CHOICE OF PARAMETER

As mentioned in per-channel quantziation, a scalar parameter λ is introduced to enable a gradual reduction in the quantization step. Empirically determined, the optimal λ is set to 0.7 for 2-bit quantization and 0.85 for 3-bit quantization. For 4-bit quantization, λ remains fixed at 1, indicating the use of the standard quantization step.

Furthermore, K is also a very important parameter in our algorithm. To strike a balance between runtime efficiency and performance, we typically maintain $K = 3$ across all bits for per-layer quantization. However, for per-channel quantization, there are slight variations among different bit configurations. Specifically, K is set to 2 for 2-bit and 3-bit quantization. For 4-bit quantization, K is increased to 4 to enhance performance.

B. COMPARISON WITH THE STATE-OF-THE-ARTS

1) VISION TRANSFORMER

We evaluated COMQ with per-channel quantization in Alg. 2 on various architectures of vision transformers, such as ViT,

TABLE 1. Comparison of ImageNet Top-1 accuracy (%) on ViTs using *per-channel* weight-only uniform quantization.

Method	WBit	ViT-S	ViT-B	DeiT-S	DeiT-B	Swin-T	Swin-S
Baseline	32	81.39	84.53	79.85	81.99	81.38	83.21
FQ-ViT [47]	4	-	81.09	76.23	79.92	78.81	81.89
PTQ4ViT [45]	4	72.34	72.06	77.50	80.07	78.46	80.24
Ours	4	**80.35**	**83.86**	**78.98**	**81.40**	**80.89**	**82.85**
FQ-ViT [47]	3	-	34.31	51.06	65.64	65.38	71.88
PTQ4ViT [45]	3	38.77	19.85	70.22	75.42	70.74	73.46
Ours	3	**77.08**	**81.73**	**77.47**	**80.47**	**79.31**	**81.95**
Ours	2	**52.44**	**65.69**	**67.19**	**77.14**	**74.05**	**78.02**

TABLE 2. Comparison of ImageNet Top-1 accuracy (%) on ViTs using *per-channel* full uniform quantization.

Method	Bit (W/A)	ViT-S	ViT-B	DeiT-S	DeiT-B	Swin-S
BRECQ [22]	4/4	12.36	9.68	63.73	72.31	72.74
QDrop [27]	4/4	21.42	47.30	68.27	72.60	79.58
APQ-ViT [46]	4/4	47.95	41.41	43.55	67.48	77.15
RepQ-ViT [38]	4/4	65.05	68.48	69.03	75.61	79.45
Ours	4/4	**71.47**	**78.27**	**72.17**	**78.72**	**81.19**
Ours	2/4	**30.11**	**45.33**	**53.20**	**71.90**	**75.37**

DeiT, and Swin, and compared its performance with previous methods PTQ4ViT [43] and FQ-ViT [47]. These methods exhibit good performance in 4-bit quantization but perform poorly in lower 3-bit quantization and do not work for 2-bit quantization. Tab. 1 compares our method with these methods for weight quantization. Tab. 2 further quantizes the activations into 4-bit. Across all ViT models and precision levels, COMQ has better performance than the existing PTQ methods. In particular, we achieve a remarkable high accuracy for 3-bit quantization. To be noted, our method is the first to push the precision down to 2-bit (2W32A and 2W4A) quantization while maintaining high accuracy. Note that for the activation quantization, we adopted the reparameterization method proposed in [38] and incorporated it into COMQ.

TABLE 3. Comparison of ImageNet Top-1 accuracy (%) on ResNets and MobileNetV2 with *per-layer* weight-only uniform quantization.

Method	WBit	ResNet18	ResNet50	MobileNetV2
Baseline	32	69.76	76.13	72.49
AdaRound [23]	4	66.56	-	-
Bit-spilt [59]		68.31	70.56	-
AdaQuant [26]		68.12	74.68	44.78
Ours†		68.76	75.19	70.51
Ours		**69.26**	**75.50**	**70.49**
Bit-spilt [59]	3	64.77	66.98	-
AdaQuant [26]		59.21	64.98	12.56
Ours†		65.63	68.23	62.36
Ours		**65.72**	**71.64**	**62.43**

2) CONVOLUTIONAL NEURAL NETWORKS

Beyond Vision Transformer, we also evaluated COMQ with per-layer and per-channel quantization on CNN models and compared it with state-of-the-art uniform post-training quantization methods, including Bit-split [59], AdaRound [23], AdaQuant [42], BRECQ [22] and OBQ [24].

a: PER-LAYER QUANTIZATION

Per-layer quantization is more computationally efficient but per-layer quantization in existing methods typically leads to high performance degradation. In contrast, our greedy COMQ for per-layer quantization in Alg. 1 yields promising results, particularly on ResNets. With the float models from PyTorch, we compare COMQ with state-of-the-art uniform per-layer PTQ methods, such as AdaQuant [23] and Bit-split [59]. The results are presented in Table 3. For 4-bit, COMQ achieves a mere 1.0% accuracy drop in ResNet18 and a 0.94% accuracy drop in ResNet50, outperforming other methods for per-layer PTQ. More remarkably, for 3-bit quantization, COMQ achieves a 4.13% and an 7.9% accuracy drop on ResNet18 and ResNet50, respectively, surpassing all competing methods. The cyclic COMQ for per-layer quantization is highlighted by a † symbol.

TABLE 4. Comparison of ImageNet Top-1 accuracy (%) on ResNets using *per-channel* weight-only uniform quantization.

Method	Bit (W/A)	ResNet18	ResNet50
Baseline	32/32	71.00	76.63
Bit-split [59]	4/32	69.11	75.58
AdaRound [23]	4/32	68.71	75.23
FlexRound [60]	4/32	70.28	75.95
BRECQ [22]	4/32	70.70	76.29
OBQ [24]	4/32	70.42	76.09
Ours	4/32	**70.83**	**76.38**
Bit-split [59]	3/32	66.75	73.24
AdaRound [23]	3/32	68.07	73.42
FlexRound [23]	3/32	68.65	74.38
BRECQ [22]	3/32	**69.81**	75.61
OBQ [24]	3/32	68.96	74.23
Ours	3/32	69.63	**75.73**
AdaRound [23]	2/32	55.96	47.95
FlexRound [23]	2/32	62.57	63.67
BRECQ [22]	2/32	**66.30**	**72.40**
OBQ [24]	2/32	63.15	68.49
Ours	2/32	64.52	70.32

b: PER-CHANNEL QUANTIZATION

We evaluate greedy COMQ for per-channel quantization described in Alg. 2 and compared its performance with state-of-the-art PTQ methods with uniform quantization, such as Bit-split [59], AdaRound [23], AdaQuant [42], BRECQ [22], and OBQ [24]. The results are shown in Tab. 4 and Tab. 5. For 4-bit quantization, our method almost attains lossless accuracy and outperforms the existing methods on both ResNet18 and ResNet50. For 3-bit, Greedy COMQ exhibits a comparable accuracy on ResNet18 and slightly outperforms existing methods on ResNet50. Even for 2-bit quantization, our method exhibits promising results on both models. Although the results are inferior to those of BRECQ, we remark that the implementations of BRECQ and other similar algorithms rely on costly back-propagation. For instance, the runtime of COMQ for quantizing ResNet18 on an Nvidia RTX 3090 GPU is merely 2 minutes, in stark contrast to the nearly 50 minutes required by BRECQ.

TABLE 5. Comparison of ImageNet Top-1 accuracy (%) on ResNets using *per-channel* full quantization.

Method	Bit (W/A)	ResNet18	ResNet50
Baseline	32/32	71.00	76.63
Bit-split [59]	4/4	67.56	73.71
AdaRound [23]	4/4	69.20	72.79
FlexRound [60]	4/4	69.26	75.08
BRECQ [22]	4/4	69.60	75.05
QDrop [27]	4/4	69.62	75.45
Ours	4/4	**69.70**	**75.46**

3) LARGE LANGUAGE MODEL

We further evaluate COMQ under per-channel quantization across a range of large language model architectures, specifically including the OPT and LLaMA2 model families. The performance of COMQ is benchmarked against prior representative quantization methods, namely Round-To-Nearest (RTN) and GPTQ [43]. We report the perplexity (PPL) of quantized models on standard language generation tasks, using the WikiText2 [58] and C4 [61] datasets, with detailed results presented in Table 6 and Appendix 9. Table 6 reports the perplexity of quantized OPT and LLaMA-2 models on the C4 dataset. On the OPT models, COMQ consistently outperforms all baseline methods under both 4-bit and 3-bit quantization, yielding substantial improvements across all evaluated model size. For the LLaMA2 family, COMQ achieves highly competitive results at 4-bit quantization, and demonstrates consistent superiority over existing methods under the more challenging 3-bit quantization.

C. ABLATION STUDY

In this subsection, we examine how various factors influence the performance and efficacy of our algorithm. We evaluate our method on CNNs with float models from PyTorch and Vit-B with float model from open source Timm.

TABLE 6. Perplexity of quantized OPT and LLaMA2 models on C4. We report C4 perplexity in this table. The results on Wikitext2 can be found in the Appendix.

Datasets	Bit (W/A)	C4 (PPL↓)				
Model		OPT			LLaMA2	
Size		2.7B	6.7B	13B	7B	13B
Baseline	FP16	13.16	11.74	11.20	6.97	6.46
RTN	4/32	18.43	14.36	13.36	7.71	6.83
OPTQ [43]	4/32	15.00	13.18	12.26	7.37	6.70
Ours	4/32	**13.81**	**12.07**	**11.44**	**7.34**	**6.70**
RTN	3/32	1.1e4	5.2e3	3.1e3	402.35	12.51
OPTQ [43]	3/32	18.17	17.14	13.34	9.81	8.02
Ours	3/32	**17.48**	**13.64**	**12.61**	**9.05**	**8.01**

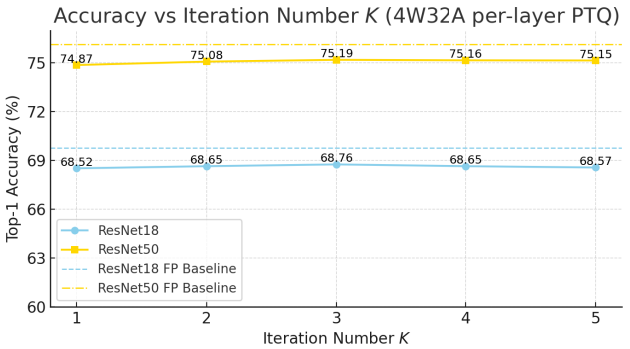


FIGURE 4. Accuracy vs Iteration Number *K* for 4W32A per-layer PTQ.

1) BATCH SIZE

All experiments are for per-channel PTQ. The number of operations (dot product and rounding) required by COMQ depends on the number of weights, not on the batch size. The larger batch size will result only in dot products performed in a higher dimension, which can still be efficient. We see that COMQ also performs reasonably well with a small batch size as shown by Tab 7.

TABLE 7. Accuracy vs Batchsize for 4W32A per-channel PTQ.

Batch size	128	256	512	1024	2048	FP Baseline
ResNet18	69.34	69.28	69.47	69.54	69.75	69.76
ResNet50	76.01	76.05	76.04	76.08	76.09	76.13
ViT-B	83.53	83.51	83.73	83.86	83.79	84.53

2) ITERATION NUMBER K

All experiments are for per-layer PTQ. As shown in Fig 4, more iterations do not necessarily equate to better results. Typically, the optimal solution is achieved when $K = 3$, additional iterations may not deliver significant improvements.

3) CYCLIC AND GREEDY

We compare greedy update order and cyclic update order for the per-channel quantization in Alg. 2. We evaluate the

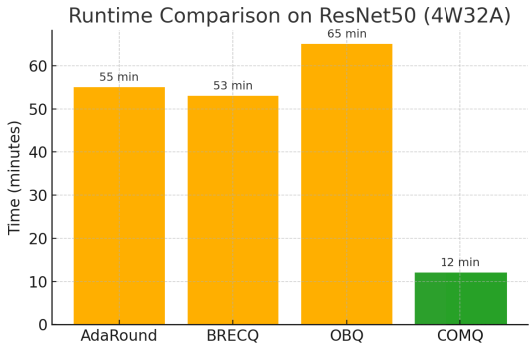


FIGURE 5. Runtime of 4W32A COMQ (greedy), which is faster than the gradient-based methods.

two update orders on five widely used models: ResNet18, ResNet50, ViT-S, DeiT-S, and Swin-T. The pre-trained ResNet family models are sourced from the PyTorch platform, and the pre-trained vision transformer models are trained on ImageNet using the open-source Timm library. The results are shown in Tab 8. We can find that the greedy update order outperforms the cyclic update order in all test cases across all models and precisions, demonstrating that the proposed greedy update order greatly improves the performance of quantized models. Furthermore, the performance improvement is more significant for larger models at lower bit-widths.

TABLE 8. ImageNet results for cyclic and greedy COMQ with weight quantization.

Method	Bits	RN18	RN50	ViT-S	DeiT-S	Swin-T
FP	32	71.00	76.63	81.39	81.99	81.38
Cyclic	4	70.71	76.29	80.16	78.94	80.85
Greedy	4	**70.83**	**76.38**	**80.35**	**78.98**	**80.89**
Cyclic	3	69.53	75.58	76.58	77.20	78.81
Greedy	3	**69.63**	**75.73**	**77.08**	**77.47**	**79.31**
Cyclic	2	64.24	69.21	49.27	65.46	73.05
Greedy	2	**64.52**	**70.32**	**52.44**	**67.19**	**74.05**

4) RUNTIME

We compared several currently top-performing methods on ResNet50, and the results are presented in the Fig 5. COMQ demonstrates significantly improved running time compared to other methods.

V. CONCLUDING REMARKS

In conclusion, our research presented COMQ, a novel coordinate-wise minimization algorithm designed for the post-training quantization (PTQ) of convolutional neural nets and transformers. COMQ solves the minimization of the layer-wise squared reconstruction error, treating all quantization parameters within the same layer, including weights and floating-point scalars, as variables in the error function. One notable feature of COMQ is its efficiency in each iteration, which involves only dot products and rounding operations. This simplicity distinguishes COMQ from existing PTQ approaches, making it a low-cost alternative. Importantly, the algorithm requires no hyper-parameter tuning to achieve state-of-the-art performance in image classification tasks. Our experiments demonstrate that COMQ surpasses existing methods, especially in the ultra-low bit-width regime, showcasing superior uniform PTQ results on ImageNet for Vision Transformers. This highlights the effectiveness of COMQ in achieving optimal quantization outcomes with minimal computational overhead, thus contributing to the advancement of PTQ techniques for DNNs. Future work includes the extension of PTQ based on prediction difference metric [62] and for multimodal models such as vision-language models [63]. It is also possible to combine per-layer and per-channel quantization strategies into a mix-precision quantization framework [64].

APPENDIX A
MORE EXPERIMENTS ON LLMS

Table 6 shows the perplexity of quantized OPT and LLaMA2 models on Wikitext2. On the OPT models, COMQ achieves competitive performance under the 4-bit quantization and consistently outperforms all baseline methods under the more challenging 3-bit quantization, indicating its superior adaptability in ultra-low-bit scenarios. Regarding the LLaMA2 family, COMQ attains the best overall perplexity results across both 3-bit and 4-bit quantization, further demonstrating its effectiveness across diverse model architectures and quantization granularities.

TABLE 9. Perplexity of quantized OPT and LLaMA2 models on Wikitext2.

Datasets	Bit (W/A)	Wikitext2 (PPL↓)				
Model		OPT			LLaMA2	
Size		2.7B	6.7B	13B	7B	13B
Baseline	FP16	12.47	10.85	10.13	5.47	4.88
RTN	4/32	16.92	12.10	11.32	6.11	5.20
OPTQ [43]	4/32	**12.87**	11.39	**10.31**	5.83	5.13
Ours	4/32	12.93	**11.17**	10.33	**5.75**	**5.08**
RTN	3/32	1.6e4	5.8e3	3.4e3	539.48	10.68
OPTQ [43]	3/32	16.88	14.86	11.61	8.37	6.44
Ours	3/32	**15.53**	**11.75**	**11.56**	**7.02**	**5.84**

APPENDIX B
MORE ABLATION STUDY

Table 10 shows the performance of our algorithm on Swin ViTs for different values of λ in 2 bits. It is clear that the

TABLE 10. ImageNet accuracy for different λ initialization. $\lambda = 0.71$ is empirically (near-) optimal for Swin ViTs.

λ	Bits	Swin-T	Swin-S
1	2	65.05	70.06
0.71	2	**74.05**	**78.02**
FP	32	81.38	83.21

results of $\lambda = 0.71$ (near-optimal λ) are much better than that of $\lambda = 1$.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 60, May 2017, pp. 84–90.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[4] J. Jumper, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, Jul. 2021.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Tech. Rep., 2018.

[7] J. Ho, A. N. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 6840–6851.

[8] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, and X. Chang, "Block-wisely supervised neural architecture search with knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1989–1998.

[9] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1398–1406.

[10] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.

[11] M. Rastegari, V. Ordóñez, J. Redmon, and A. Farhadi, "XNOR-net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland. Springer, Jan. 2016, pp. 525–542.

[12] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura, "Mixed precision DNNs: All you need is a good parametrization," 2019, *arXiv:1905.11452*.

[13] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, Jan. 2016.

[15] P. Wang and J. Cheng, "Fixed-point factorized networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jan. 2016, pp. 4012–4020.

[16] P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, and J. Xin, "Blended coarse gradient descent for full quantization of deep neural networks," *Res. Math. Sci.*, vol. 6, no. 1, pp. 1–23, Jan. 2019.

[17] P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, and J. Xin, "BinaryRelax: A relaxation approach for training deep neural networks with quantized weights," *SIAM J. Imag. Sci.*, vol. 11, no. 4, pp. 2205–2223, Jan. 2018.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[19] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2019.

[20] Z. Li, B. Yang, P. Yin, Y. Qi, and J. Xin, "Feature affinity assisted knowledge distillation and quantization of deep neural networks on label-free data," *IEEE Access*, vol. 11, pp. 78042–78051, 2023.

[21] P. Yin, S. Zhang, Y. Qi, and J. Xin, "Quantization and training of low bit-width convolutional neural networks for object detection," 2016, *arXiv:1612.06052*.

[22] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "BRECQ: Pushing the limit of post-training quantization by block reconstruction," 2021, *arXiv:2102.05426*.

[23] M. Nagel, R. A. Amjad, M. v. Baalen, C. Louizos, and T. Blankevoort, "Up or down? Adaptive rounding for post-training quantization," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2020, pp. 7197–7206.

[24] E. Frantar and D. Alistarh, "Optimal brain compression: A framework for accurate post-training quantization and pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 4475–4488.

[25] L. Chen, B. Peng, Z. Li, W. Tan, Y. Ren, J. Xiao, and S. Pu, "Bit-shrinking: Limiting instantaneous sharpness for improving post-training quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 16196–16205.

[26] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," 2020, *arXiv:2006.10518*.

[27] X. Wei, R. Gong, Y. Li, X. Liu, and F. Yu, "QDrop: Randomly dropping quantization for extremely low-bit post-training quantization," 2022, *arXiv:2203.05740*.

[28] N. Wang, C.-C. C. Liu, S. Venkataramani, S. Sen, C.-Y. Chen, K. El Maghraoui, V. V. Srinivasan, and L. Chang, "Deep compression of pre-trained transformer models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 14140–14154.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jeǵou, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 10347–10357.

[31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002.

[32] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 28092–28103.

[33] J. Zhang, Y. Zhou, and R. Saab, "Post-training quantization for neural networks with provable guarantees," *SIAM J. Math. Data Sci.*, vol. 5, no. 2, pp. 373–399, Jun. 2023.

[34] P. Wang, W. Chen, X. He, Q. Chen, Q. Liu, and J. Cheng, "Optimization-based post-training quantization with bit-split and stitching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2119–2135, Feb. 2023.

[35] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, Jan. 2012.

[36] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, Jun. 2015.

[37] C. Hsieh, K. Chang, C. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proc. 25th Int. Conf. Mach. Learn.*, Jan. 2008, pp. 408–415.

[38] Z. Li, J. Xiao, L. Yang, and Q. Gu, "RepQ-ViT: Scale reparameterization for post-training quantization of vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2023, pp. 17181–17190.

[39] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1325–1334.

[40] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "ZeroQ: A novel zero shot quantization framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13166–13175.

[41] S. Xu, H. Li, B. Zhuang, J. Liu, J. Cao, C. Liang, and M. Tan, "Generative low-bitwidth data free quantization," in *Proc. 16th Eur. Conf., Comput. Vis. (ECCV)*, Jan. 2020, pp. 1–17.

[42] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 4466–4475.

[43] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," 2022, *arXiv:2210.17323*.

[44] K. Behdin, A. Acharya, A. Gupta, Q. Song, S. Zhu, S. Keerthi, and R. Mazumder, "QuantEase: Optimization-based quantization for language models," 2023, *arXiv:2309.01885*.

[45] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "PTQ4 ViT: Post-training quantization for vision transformers with twin uniform quantization," 2021, *arXiv:2111.12293*.

[46] Y. Ding, H. Qin, Q. Yan, Z. Chai, J. Liu, X. Wei, and X. Liu, "Towards accurate post-training quantization for vision transformer," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 5380–5388.
[47] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "FQ-ViT: Post-training quantization for fully quantized vision transformer," 2021, *arXiv:2111.13824*.
[48] Y. He, L. Liu, J. Liu-Zeng, W. Wu, H. Zhou, and B. Zhuang, "PTQD: Accurate post-training quantization for diffusion models," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023.
[49] Y. Shang, Z. Yuan, B. Xie, B. Wu, and Y. Yan, "Post-training quantization on diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 1972–1981.
[50] G. Xiao, J. Lin, M. Seznec, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2022, pp. 38087–38099.
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jun. 2017, pp. 5998–6008.
[52] J. Choi, Z. Wang, S. Venkataramani, P. I-Jen Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2016, pp. 770–778.
[54] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, Jun. 2009, pp. 248–255.
[56] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "OPT: Open pre-trained transformer language models," 2022, *arXiv:2205.01068*.
[57] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, and S. Bhosale, "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.
[58] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv:1609.07843*.
[59] P. Wang, Q. Chen, X. He, and J. Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, Jul. 2020, pp. 9847–9856.
[60] J. H. Lee, J. Kim, S. J. Kwon, and D. Lee, "FlexRound: Learnable rounding based on element-wise division for post-training quantization," 2023, *arXiv:2306.00317*.
[61] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 1–67, Jan. 2019.
[62] J. Liu, L. Niu, Z. Yuan, D. Yang, X. Wang, and W. Liu, "PD-quant: Post-training quantization based on prediction difference metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jan. 2022, pp. 24427–24437.
[63] M. Wortsman, T. Dettmers, L. Zettlemoyer, A. S. Morcos, A. Farhadi, and L. Schmidt, "Stable and low-precision training for large-scale vision-language models," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023.
[64] A. Chauhan, U. Tiwari, and N. R. Vikram, "Post training mixed precision quantization of neural networks using first-order information," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2023, pp. 1335–1344.

AOZHONG ZHANG received the B.S. degree from Sun Yat-sen University. He is currently pursuing the Ph.D. degree in mathematics with the State University of New York at Albany. He was a Visiting Student with the Multimedia Laboratory of Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, from 2021 to 2022. His research interests include efficient machine learning, quantization, and efficient finetuning.

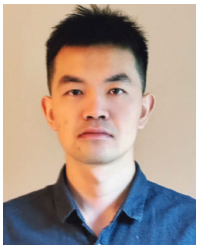
ZI YANG received the Ph.D. degree in mathematics from UC San Diego, in 2021. He is currently an Assistant Professor with the Department of Mathematics and Statistics, University at Albany, SUNY. Before joining UAlbany, he was a Postdoctoral Scholar with the Department of Electrical and Computer Engineering, UC Santa Barbara.

NAIGANG WANG received the B.S. degree in materials science from Jilin University, and the Ph.D. degree in materials science from the University of Florida. He is currently a Research Scientist with IBM T. J. Watson Research Center. He has over 50 journal and conference papers and over 90 filed patents. His current research interests include advancing hardware-friendly machine learning algorithms and AI accelerators, particularly by leveraging the capabilities of low-precision and sparsity to accelerate the training and inference processes of deep learning models.

YINGYONG QI received the first Ph.D. degree in speech and hearing sciences from The Ohio State University, in 1989, and the second Ph.D. degree in electrical and computer engineering from The University of Arizona, in 1993. He held a Faculty position with The University of Arizona, from 1989 to 1999. He was a Visiting Scientist with the Research Laboratory of Electronics, Massachusetts Institute of Technology, from 1995 to 1996, and a Visiting Scientist with the Visual Computing Laboratory, Hewlett Packard, Palo Alto, in 1998. He is currently the Senior Director of technology at Qualcomm and a Researcher at the Department of Mathematics, University of California at Irvine. He has published over 100 scientific papers and U.S. patents during his tenure at university and industry. His research interests include speech processing, computer vision, and machine learning. He received the Klatt Memorial Award in speech science from the Acoustical Society of America, in 1991, the First Award from the National Institute of Health, in 1992, and the AASFAA Outstanding Faculty Award from Arizona Association of Scholar and Faculty of Asian American, in 1998. He was in the first place of Qualcomm Invention Network Competition, in 2010. More recently, he led a team winning the 3rd place of IEEE Low Power Image Recognition Competition, sponsored by Google, CVPR, in 2019.

JACK XIN received the Ph.D. degree in mathematics from the Courant Institute of Mathematical Sciences, New York University, in 1990. He was a Faculty with The University of Arizona, from 1991 to 1999, and The University of Texas at Austin, from 1999 to 2005. He is currently a Chancellor's Professor of mathematics with UC at Irvine, Irvine. His research interests include applied analysis and computational methods, and their applications in multi-scale problems and data science. He is a fellow of the Guggenheim Foundation, American Mathematical Society, American Association for the Advancement of Science, the Society for Industrial and Applied Mathematics, and Asia-Pacific Artificial Intelligence Association. He was elected as a member of the National Academy of Artificial Intelligence in 2025. He was a recipient of Qualcomm Faculty Award, from 2019 to 2022 and Qualcomm Gift Award, from 2023 to 2024.

XIN LI (Fellow, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, in 1996, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2000. He was a member of Technical Staff with Sharp Laboratories of America, Camas, WA, USA, from August 2000 to December 2002. From January 2003 to August 2023, he was a Faculty Member with the Lane Department of Computer Science and Electrical Engineering, West Virginia University. In 2023, he joined the Department of Computer Science, University at Albany. His current research interests include computer vision, NeuroAI, multimodal AI, foundation models, and autism research. He was elected as a fellow of IEEE in 2017 for his contributions to image interpolation, restoration, and compression.



PENGHANG YIN received the Ph.D. degree in applied mathematics from the University of California at Irvine, in 2016. From 2016 to 2019, he was an Assistant Adjunct Professor with the Department of Mathematics, University of California at Los Angeles. He is currently an Assistant Professor with the Department of Mathematics and Statistics, State University of New York at Albany. His research interests include applied and computational mathematics and deep learning. He won the Kovalevsky Outstanding Ph.D. Thesis Award.