Binary Quadratic Quantization: Beyond First-Order Quantization for Real-Valued Matrix Compression

Kyo Kuroki
Institute of Science Tokyo
kuroki.kyo@artic.iir.isct.ac.jp

Yasuyuki Okoshi
Institute of Science Tokyo
okoshi.yasuyuki@artic.iir.isct.ac.jp

Thiem Van Chu
Institute of Science Tokyo
thiem@artic.iir.isct.ac.jp

Kazushi Kawamura
Waseda University
kawamura.k.au@waseda.jp

Masato Motomura
Institute of Science Tokyo
motomura@artic.iir.isct.ac.jp

Abstract

This paper proposes a novel matrix quantization method, Binary Quadratic Quantization (BQQ). In contrast to conventional first-order quantization approaches—such as uniform quantization and binary coding quantization—that approximate real-valued matrices via linear combinations of binary bases, BQQ leverages the expressive power of binary quadratic expressions while maintaining an extremely compact data format. We validate our approach with two experiments: a matrix compression benchmark and post-training quantization (PTQ) on pretrained Vision Transformer-based models. Experimental results demonstrate that BQQ consistently achieves a superior trade-off between memory efficiency and reconstruction error than conventional methods for compressing diverse matrix data. It also delivers strong PTQ performance, even though we neither target state-of-the-art PTQ accuracy under tight memory constraints nor rely on PTQ-specific binary matrix optimization. For example, our proposed method outperforms the state-of-the-art PTQ method by up to 2.2% and 59.1% on the ImageNet dataset under the calibration-based and data-free scenarios, respectively, with quantization equivalent to 2 bits. These findings highlight the surprising effectiveness of binary quadratic expressions for efficient matrix approximation and neural network compression.

1 Introduction

Modern information systems increasingly demand efficiency in both computation and resource usage, driven by growing model sizes, data volumes, and deployment requirements across diverse hardware environments. In these systems, real-valued matrices frequently appear as weight parameters in deep neural networks (DNNs), as high-dimensional embeddings in retrieval systems, and as training datasets. Because such matrices are central to a wide range of data processing and applications, their efficient representation and compression is crucial for reducing the costs of storage, computation, and data movement—an essential step toward deploying models on edge devices, reducing memory usage in retrieval systems, or scaling to large datasets learning.

A widely adopted strategy for this purpose is quantization, which approximates continuous-valued data with discrete levels to save memory and enable faster computation. Most existing methods rely

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

on first-order scalar quantization approaches—such as Uniform Quantization (UQ) or Binary Coding Quantization (BCQ) [23]—that represent real-valued matrices as linear combinations of binary bases. While effective under moderate compression, such first-order methods often struggle to accurately reconstruct the original matrix with ultra-low-bit quantization, as the number of possible values for each element becomes extremely limited. Beyond these scalar quantization approaches, alternative techniques such as Vector Quantization (VQ) [15] and its variants (e.g., Product Quantization (PQ) [31] and Lattice Vector Quantization (LVQ) [1, 16, 60]), as well as low-rank approximations [13], have also been explored. While these methods can capture correlations among dimensions more effectively, they typically rely on codebooks or factorized components that contain unquantized real values. Consequently, although the index or low-rank representation is compact, their dependence on floating-point vectors limits hardware efficiency, unlike scalar quantization methods. Although several studies [11, 22, 64, 56] have explored combining low-rank approximation with scalar quantization, the use of extremely low-bit (e.g., binary) representations for the factorized components has not yet been fully explored. Furthermore, to our knowledge, no prior work has applied, as in BCQ, independent scaling factors for each binary matrix in a factorized representation. Incorporating such strategies could potentially enable even more efficient matrix approximation under extreme low-bit constraints. Motivated by these observations, we introduce Binary Quadratic Quantization (BQQ), a framework that represents matrices using quadratic combinations of binary variables, with independent scaling factors assigned to each binary matrix. Specifically, the target matrix is represented as a sum of binary matrix products, enabling expressive nonlinear approximations while maintaining an exceptionally compact data format. This approach pushes the boundaries of matrix quantization by addressing the limitations of traditional methods and offering a fundamentally new perspective on matrix approximation.

In this paper, we demonstrate the effectiveness of BQQ through comprehensive evaluations: (i) measuring the trade-off between quantization error and memory usage across various matrix datasets, (ii) assessing performance when applied to post-training quantization (PTQ) of DNNs. Beyond these applications, the generality of our framework suggests its potential for other scenarios where efficient matrix approximations are essential, such as accelerating approximate nearest neighbor (ANN) [2, 47]-based retrieval systems and improving the scalability of large-scale learning powered by abundant training data [70, 73].

Our main contributions are:

- We propose BQQ, a novel matrix quantization framework based on quadratic expressions of binary matrices, offering a new perspective on extreme matrix compression.
- Minimizing the quantization error under the BQQ formulation naturally leads to an NP-hard optimization problem. To address this issue, we develop an efficient solution based on polynomial unconstrained binary optimization (PUBO) and convex quadratic programming.
- We demonstrate that BQQ consistently achieves an excellent trade-off between memory usage and quantization error for compressing diverse matrix data.
- We show that BQQ also delivers state-of-the-art (SOTA) performance in weight PTQ of Vision Transformer [12]-based models (ViTs), even though our PTQ method is based mainly on minimizing weight reconstruction error, rather than explicitly minimizing activation error to achieve SOTA performance.

  To the best of our knowledge, this is the first study to achieve practical accuracy–such as 72% ImageNet top-1 accuracy on the DeiT-base model–using data-free PTQ for ViTs at a model size equivalent to 2-bit quantization.

2 Related Works

Quantization Quantization is a fundamental technique for reducing the precision of real-valued parameters, widely used for model compression and efficient processing. It converts continuous values into a limited set of discrete levels, with methods varying in granularity and complexity depending on hardware constraints and the acceptable accuracy-performance trade-off. UQ is the most commonly used quantization method, which approximates a real-valued matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ as:

$$\mathbf{W} \approx a \sum_{i=0}^{p-1} 2^i \mathbf{B}_i + b\mathbf{1}, \tag{1}$$

where $\mathbf{B}_i \in \{0,1\}^{m \times n}$, $a \in \mathbb{R}$ is the scaling factor, and $b \in \mathbb{R}$ is a bias (or zero-point). This form corresponds to p-bit quantization with uniform step sizes. On the other hand, non-uniform

quantization assigns quantization levels in a data-aware manner, allowing better alignment with the underlying distribution of matrices and reducing quantization error. One example is BCQ [23], which approximates a real-valued matrix as a sum of binary bases with individual scaling factors:

$$\mathbf{W} \approx \sum_{i=0}^{p-1} a_i \mathbf{B}_i, \tag{2}$$

where $a_i \in \mathbb{R}$ and $\mathbf{B}_i \in \{0,1\}^{m\times n}$ [6] or $\{-1,1\}^{m\times n}$ [62, 29, 37, 23, 65, 5, 53]. By introducing a bias term, the $\{0,1\}^{m\times n}$ and $\{-1,1\}^{m\times n}$ representations can be made equivalent and encompass UQ. This enables flexible quantization levels that can better capture the characteristics of $\mathbf{W}$. Alternatively, some methods apply a nonlinear transformation before quantization. One such method is logarithmic quantization [51], which approximates the logarithmic scale of the original weights: $\log_2|\mathbf{W}| \approx \sum_{i=0}^{p-1} 2^i \mathbf{B}_i + b\mathbf{1}$, where $\mathbf{B}_i \in \{0,1\}^{m\times n}$ and $b \in \mathbb{R}$. This technique is particularly effective when the distribution of matrix values is highly skewed. It also offers a hardware-friendly implementation of matrix multiplication, as the powers-of-two representation allows the operation to be performed using efficient bit-shift operations. However, because the sign information is lost in the logarithmic transformation, an additional bit is required to retain the original sign of each element.

Matrix Factorization Matrix factorization expresses a matrix $\mathbf{W}^{m\times n}$ exactly or approximately as a product $\mathbf{W} \approx \mathbf{YZ}$, and is a fundamental tool in signal processing, machine learning, and data analysis. The target matrix $\mathbf{W}^{m\times n}$, and the factor matrices $\mathbf{Y}^{m\times l}$ and $\mathbf{Z}^{l\times n}$ are subject to different constraints depending on the specific method. For example:

- **Singular Value Decomposition (SVD)** [19]: $\mathbf{W} \in \mathbb{R}^{m\times n}$, $\mathbf{Y} \in \mathbb{R}^{m\times l}$, $\mathbf{Z} \in \mathbb{R}^{l\times n}$, Note that singular value diagonal matrix and orthogonal matrix are described as one.
- **Non-negative Matrix Factorization (NMF)** [38]: $\mathbf{W} \in \mathbb{R}_{\geq 0}^{m\times n}$, $\mathbf{Y} \in \mathbb{R}_{\geq 0}^{m\times l}$, $\mathbf{Z} \in \mathbb{R}_{\geq 0}^{l\times n}$
- **Real/Binary Matrix Factorization (RBMF)** [58]: $\mathbf{W} \in \mathbb{R}^{m\times n}$, $\mathbf{Y} \in \mathbb{R}^{m\times l}$, $\mathbf{Z} \in \{0,1\}^{l\times n}$
- **Non-negative/Binary Matrix Factorization (NBMF)** [52]: $\mathbf{W} \in \mathbb{R}_{\geq 0}^{m\times n}$, $\mathbf{Y} \in \mathbb{R}_{\geq 0}^{m\times l}$, $\mathbf{Z} \in \{0,1\}^{l\times n}$
- **Binary Matrix Factorization (BMF)** [69]: $\mathbf{W} \in \{0,1\}^{m\times n}$, $\mathbf{Y} \in \{0,1\}^{m\times l}$, $\mathbf{Z} \in \{0,1\}^{l\times n}$
- **Boolean Matrix Factorization (BoolMF)** [49, 50]: $\mathbf{W} \in \{0,1\}^{m\times n}$, $\mathbf{Y} \in \{0,1\}^{m\times l}$, $\mathbf{Z}^{l\times n} \in \{0,1\}$, with Boolean product: $\mathbf{W}_{ij} \approx \bigvee_{k=1}^{l}(\mathbf{Y}_{ik} \wedge \mathbf{Z}_{kj})$

Such matrix factorization techniques are widely used not only in data analysis but also for matrix compression through low-rank approximation (i.e., $l \ll \min(m,n)$) [13]. Building on this idea, Low-Rank Adaptation (LoRA) [25] enables efficient fine-tuning of large pre-trained models by restricting weight updates to a low-rank subspace. More recently, [11, 64, 22] combines LoRA with quantization, further reducing memory usage while preserving model quality, and has become a widely adopted approach for resource-efficient fine-tuning.

3 Preliminaries

Polynomial Unconstrained Binary Optimization (PUBO) Polynomial Unconstrained Binary Optimization (PUBO) [17, 18] is a class of combinatorial optimization problems defined as the minimization of a multivariate polynomial over binary variables. Formally, it can be expressed as:

$$L(\boldsymbol{s}) = \sum_{i_1} J^{(1)}_{i_1} s_{i_1} + \sum_{i_1<i_2} J^{(2)}_{i_1 i_2} s_{i_1} s_{i_2} + \cdots + \sum_{i_1<i_2<\cdots<i_k} J^{(k)}_{i_1 i_2 \cdots i_k} \prod_{j=1}^{k} s_{i_j}, \tag{3}$$

where $\boldsymbol{s} \in \{0,1\}^N$ is a binary vector and $J^{(k)}$ denotes the k-th order interaction coefficients. In the special case where the degree $k = 2$, the problem reduces to the well-known Quadratic Unconstrained Binary Optimization (QUBO) formulation [54, 35]. QUBO is equivalent to minimizing the energy function of the Ising model [27] and has been widely studied in various fields, including physics, computer science, and artificial intelligence. In general, solving PUBO, including QUBO, problems is NP-hard, and thus, a broad range of heuristics [33, 20, 21, 7, 36, 46], optical computing [26, 24] and quantum computing [32, 30] have been proposed to tackle them efficiently.

Annealed Mean Field Descent One of the recent promising approaches for solving QUBO problems is Annealed Mean Field Descent (AMFD) [36], which is based on Mean Field Annealing [3]. It

aims to find minimum solutions by gradually annealing the temperature while optimizing a mean-field approximation to the canonical distribution: $P_{\mathrm{C}}(\boldsymbol{s}) = \frac{1}{Z}\exp\left(-\frac{L(\boldsymbol{s})}{T}\right),\ Z = \sum_{\boldsymbol{s}}\exp\left(-\frac{L(\boldsymbol{s})}{T}\right)$. Since computing the distribution is generally intractable due to the exponential number of configurations, it is approximated by an independent distribution for each variable (mean-field approximation): $P_{\mathrm{MF}}(\boldsymbol{s}) = \prod_{i=1}^{N} p_i(s_i)$, where $p_i(s_i)$ denotes the probability of taking value s_i. AMFD minimizes the Kullback–Leibler (KL) divergence between $P_{\mathrm{MF}}(\boldsymbol{s})$ and $P_{\mathrm{c}}(\boldsymbol{s})$ by gradient descent-based updates. At low temperatures, the canonical distribution concentrates on minimum states, thereby allowing the extraction of approximate minimum solutions. While AMFD derived an explicit form of the KL divergence for QUBO problems, this work extends the framework to general PUBO problems. We show that the KL divergence between $P_{\mathrm{MF}}(\boldsymbol{s})$ and $P_{\mathrm{C}}(\boldsymbol{s})$ can be written as:

$$D_{\mathrm{KL}}\left(P_{\mathrm{MF}}(\boldsymbol{s}) \,\|\, P_{\mathrm{C}}(\boldsymbol{s})\right) = \frac{L(\boldsymbol{x})}{T} + \ln Z + \sum_{i=1}^{N}\left[(1-x_i)\ln(1-x_i) + x_i \ln x_i\right], \tag{4}$$

where $x_i = \sum_{s_i=0}^{1} s_i p_i(s_i) = p_i(1)$ is the expectation of s_i under the mean-field distribution. Please refer to the App. A.1 for a detailed derivation of this formulation. Note that x_i is a real-valued variable ranging from 0 to 1, rather than a binary variable. Therefore, the KL divergence is differentiable with respect to x_i. One iteration of the AMFD algorithm is illustrated in Alg. 1. Note that the term $[(1-x_i)\ln(1-x_i) + x_i \ln x_i]$ in Eq. (4) is approximated by a second-order Taylor expansion around $x_i = 0.5$ to prevent numerical overflow when x_i is close to 0 or 1. This paper applies the extended AMFD to optimize quantized representations under the general PUBO setting.

4 Proposed Method

4.1 Binary Quadratic Quantization (BQQ)

A primary limitation of conventional first-order quantization methods like UQ and BCQ is the limited number of distinct values each element can take. For example, 1-bit quantization allows only two levels (e.g., $\{-1, +1\}$ or $\{0, 1\}$), and 2-bit quantization increases this to just four. Such coarse granularity restricts representational flexibility, especially under aggressive compression. While such methods are limited in expressiveness, binary matrix multiplication can yield outputs with a wider value range, enabling multi-bit representations even though each binary matrix individually encodes only minimal information. This property suggests a previously underexplored potential for approximating real-valued matrices through compositions of binary matrices, offering a new perspective beyond traditional quantization methods. Nonetheless, existing matrix decomposition approaches operate within fixed numerical domains—either real-to-real (e.g., SVD, NMF) or binary-to-binary (e.g., BMF, BoolMF). Hybrid methods like RBMF and NBMF bridge these domains but stop short of fully binary decompositions of real-valued matrices.

Motivated by this gap, we explore a novel quantization scheme that, unlike BCQ which uses linear combinations of binary matrices, is based on linear combinations of binary matrix products:

$$\boldsymbol{W} \approx \sum_{i=0}^{p-1} (\alpha_i \boldsymbol{Y}_i + \beta_i \mathbf{1}_Y)(\gamma_i \boldsymbol{Z}_i + \delta_i \mathbf{1}_Z), \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, $\boldsymbol{Y}_i \in \{0,1\}^{m\times l}$ and $\boldsymbol{Z}_i \in \{0,1\}^{l\times n}$ are binary matrices, while $\alpha_i, \gamma_i \in \mathbb{R}$ are scaling factors, and $\beta_i, \delta_i \in \mathbb{R}$ are bias terms. Also, $\mathbf{1}_Y, \mathbf{1}_Z, \mathbf{1}$ denote all-ones matrices with the same shape as $\mathbf{Y}, \mathbf{Z}, \mathbf{W}$, respectively. Notably, this formulation subsumes BCQ as a special case.

Algorithm 1 One Iteration of AMFD [36]

Input: L, $\boldsymbol{x}_{\text{old}}$, $\boldsymbol{x}_{\text{cur}}$, T, ΔT, η, ζ
Output: $\boldsymbol{x}_{\text{cur}}, \boldsymbol{x}_{\text{new}}, T$

$\boldsymbol{x}_{\text{fwd}} \leftarrow \boldsymbol{x}_{\text{cur}} + \zeta\left(\boldsymbol{x}_{\text{cur}} - \boldsymbol{x}_{\text{old}}\right)$ ▷ Forward point
$\boldsymbol{\Phi} \leftarrow \nabla L(\boldsymbol{x}_{\text{fwd}})$ ▷ Gradient of the first term in Eq. (4), scaled by T
$\boldsymbol{F} \leftarrow T\left(\boldsymbol{x}_{\text{cur}} - \mathbf{0.5}\right)$ ▷ Gradient of the other terms in Eq. (4) (approx. expr.) scaled by T
$\boldsymbol{x}_{\text{new}} \leftarrow \texttt{clip}\left(2\boldsymbol{x}_{\text{cur}} - \boldsymbol{x}_{\text{old}} - \eta\left(\boldsymbol{F} + \boldsymbol{\Phi}\right), 0, 1\right)$ ▷ Descent with acceleration and constraints
$T \leftarrow T - \Delta T$ ▷ Annealing

Specifically, when $l = \max(m, n)$, setting $\alpha_i \boldsymbol{Y}_i + \beta_i \mathbf{1}_Y$ as the identity matrix (if $m \geq n$), or setting $\gamma_i \boldsymbol{Z}_i + \delta_i \mathbf{1}_Z$ as the identity matrix (if $m \leq n$), recovers the standard BCQ structure. We now turn our attention to the generalized form of Eq. (5), referred to as Binary Quadratic Quantization (BQQ):

$$\boldsymbol{W} \approx \sum_{i=0}^{p-1} (r_i \boldsymbol{Y}_i \boldsymbol{Z}_i + s_i \boldsymbol{Y}_i \mathbf{1}_Z + t_i \mathbf{1}_Y \boldsymbol{Z}_i) + u\mathbf{1}, \tag{6}$$

where $r_i, s_i, t_i, u \in \mathbb{R}$ are scalar coefficients. Note that the all-one matrices are used only for notational convenience and do not need to be stored explicitly; only the binary matrices and scalar coefficients must be preserved. Also, the intermediate dimension l can be arbitrarily set, allowing the number of binary elements to be adjusted independently of the original matrix size.

4.2 Mixed Integer Programming for BQQ

To realize BQQ formulation (Eq. (6)), we consider minimizing the squared error between the original real-valued matrix and its approximation. The objective function is given by:

$$L_{\text{BQQ}} = \left\| \boldsymbol{W} - \left[\sum_{i=0}^{p-1} (r_i \boldsymbol{Y}_i \boldsymbol{Z}_i + s_i \boldsymbol{Y}_i \mathbf{1}_Z + t_i \mathbf{1}_Y \boldsymbol{Z}_i) + u\mathbf{1} \right] \right\|_2^2, \tag{7}$$

where the goal is to optimize $3p + 1$ real-valued coefficients and the elements of $2p$ binary matrices. This is a mixed-integer optimization problem and is NP-hard, making analytical solutions intractable.

To address this, we adopt the following strategy:

1. We apply greedy optimization independently to each index i in Eq. (7) to mitigate the increasing complexity from a growing number of binary variables.
2. We decouple the optimization of real-valued and binary variables, and alternate between convex quadratic optimization and PUBO.

To perform greedy optimization for each index i in Eq. (7), we first define the residual matrix $\boldsymbol{W}_{\text{res}}^{(i)}$ as the difference between the original matrix $\boldsymbol{W}$ and the partial reconstruction using variables up to index $i-1$: $\boldsymbol{W}_{\text{res}}^{(i)} = \boldsymbol{W} - \left[\sum_{j=0}^{i-1} (r_j \boldsymbol{Y}_j \boldsymbol{Z}_j + s_j \boldsymbol{Y}_j \mathbf{1}_Z + t_j \mathbf{1}_Y \boldsymbol{Z}_j + u_j \mathbf{1}) \right]$. Notably, as an exception, we set $\boldsymbol{W}_{\text{res}}^{(0)} = \boldsymbol{W}$. Then, the i-th subproblem can be formulated as the minimization of the following objective:

$$L_{\text{sub}}^{(i)} = \left\| \boldsymbol{W}_{\text{res}}^{(i)} - (r_i \boldsymbol{Y}_i \boldsymbol{Z}_i + s_i \boldsymbol{Y}_i \mathbf{1}_Z + t_i \mathbf{1}_Y \boldsymbol{Z}_i + u_i \mathbf{1}) \right\|_2^2. \tag{8}$$

Next, to minimize the objective function (Eq. (8)), we adopt an alternating optimization approach that separates continuous and binary variables. When the continuous coefficients are fixed, the problem

Algorithm 2 Subproblem Solving via AMFD

Input: Input matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, initial temperature T_{init}, final temperature T_{fin}, steps N_{step}, learning rate η, accelerating rate ζ, intermediate dimension l
Output: Binary matrices $\mathbf{Y} \in \{0,1\}^{m \times l}$, $\mathbf{Z} \in \{0,1\}^{l \times n}$, scaling factors $r_i, s_i, t_i, u_i \in \mathbb{R}$

Let $\mathbf{1} \in \{1\}^{m \times n}$, $\mathbf{1}_Y \in \{1\}^{m \times l}$, and $\mathbf{1}_Z \in \{1\}^{l \times n}$
Sample $\hat{\mathbf{Y}}_{\text{old}}, \hat{\mathbf{Z}}_{\text{old}} \sim \mathcal{U}(0,1)$ ▷ Initial expectation values
$\hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}}_{\text{old}} - \eta(\hat{\mathbf{Y}}_{\text{old}} - \mathbf{0.5})$, $\hat{\mathbf{Z}} \leftarrow \hat{\mathbf{Z}}_{\text{old}} - \eta(\hat{\mathbf{Z}}_{\text{old}} - \mathbf{0.5})$
$\mathbf{W} \leftarrow \mathbf{W} / (\max(\mathbf{W}) - \min(\mathbf{W}))$ ▷ Normalization
$\Delta T \leftarrow (T_{\text{init}} - T_{\text{fin}})/(N_{\text{step}} - 1)$
$T \leftarrow T_{\text{init}}$
$[r_i, s_i, t_i, u_i] \leftarrow \texttt{SFO}(\hat{\mathbf{Y}}, \hat{\mathbf{Z}}, \mathbf{W})$ ▷ SFO: scaling factors optimization using Eq. (10)
for $t = 1$ to N_{step} **do**
    $[\hat{\mathbf{Y}}_{\text{old}}, \hat{\mathbf{Z}}_{\text{old}}], [\hat{\mathbf{Y}}, \hat{\mathbf{Z}}], T \leftarrow \texttt{AMFD}\left(L_{\text{pubo}}^{(i)}(\mathbf{W}, [r_i, s_i, t_i, u_i]), [\hat{\mathbf{Y}}_{\text{old}}, \hat{\mathbf{Z}}_{\text{old}}], [\hat{\mathbf{Y}}, \hat{\mathbf{Z}}], T, \Delta T, \eta, \zeta\right)$
    ▷ AMFD: AMFD iteration using Alg. 1
    $[r_i, s_i, t_i, u_i] \leftarrow \texttt{SFO}(\hat{\mathbf{Y}}, \hat{\mathbf{Z}}, \mathbf{W})$
$\mathbf{Y} \leftarrow \texttt{step}(\hat{\mathbf{Y}} - \mathbf{0.5})$, $\mathbf{Z} \leftarrow \texttt{step}(\hat{\mathbf{Z}} - \mathbf{0.5})$ ▷ Binarization to the higher probability
$[r_i, s_i, t_i, u_i] \leftarrow (\max(\mathbf{W}) - \min(\mathbf{W})) \cdot \texttt{SFO}(\mathbf{Y}, \mathbf{Z}, \mathbf{W})$

Algorithm 3 Greedy Binary Quadratic Quantization

Input: Matrix $\boldsymbol{W}$, learning rate η, accelerating rate ζ, initial temperature T_{init}, final temperature T_{fin}, steps N_{step}, intermediate dimension l, binary matrix stacks p

Output: Binary matrices $[\boldsymbol{Y}_0, \boldsymbol{Y}_1, ..., \boldsymbol{Y}_{p-1}]$, $[\boldsymbol{Z}_0, \boldsymbol{Z}_1, ..., \boldsymbol{Z}_{p-1}]$, scaling factors $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t} \in \mathbb{R}^p$, $u \in \mathbb{R}$

$\boldsymbol{W}_{\text{res}} \leftarrow \boldsymbol{W}$

for $i = 0$ to $(p-1)$ **do**

  $\boldsymbol{Y}_i, \boldsymbol{Z}_i, r_i, s_i, t_i, u_i \leftarrow \mathtt{SS}(\boldsymbol{W}_{\text{res}}, T_{\text{init}}, T_{\text{fin}}, N_{\text{step}}, \eta, \zeta, l)$ ▷ SS: subproblem solving via Alg. 2

  $\boldsymbol{W}_{\text{res}} \leftarrow \boldsymbol{W}_{\text{res}} - (r_i \boldsymbol{Y}_i \boldsymbol{Z}_i + s_i \boldsymbol{Y}_i \boldsymbol{1}_Z + t_i \boldsymbol{1}_Y \boldsymbol{Z}_i + u_i \boldsymbol{1})$

$u \leftarrow \sum_{i=0}^{p-1} u_i$

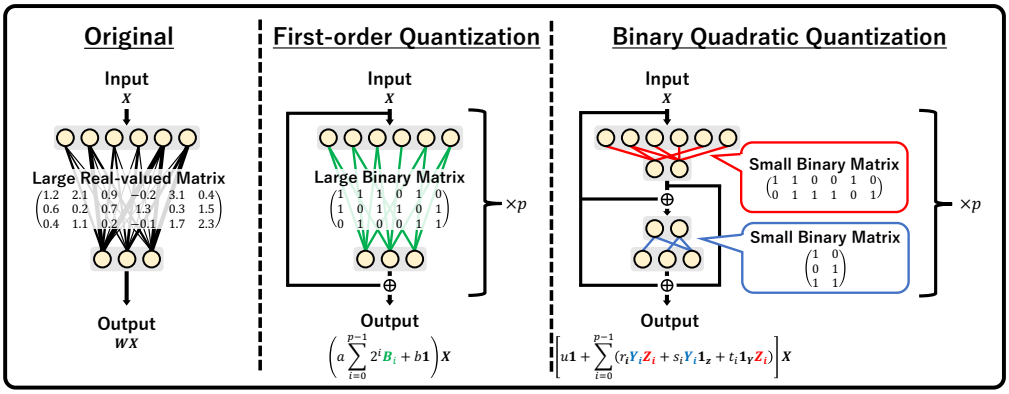

Figure 1: Comparison between BQQ and first-order quantization in a DNN layer.

reduces to optimizing over binary variables only. In this case, the objective can be reformulated as a PUBO by reducing powers of binary variables using $y^2 = y$, making each such term linear. The resulting objective takes the following form:

$$\begin{aligned} L_{\text{pubo}}^{(i)} = L_{\text{sub}}^{(i)} &+ r_i^2 \sum [\boldsymbol{Y}_i \boldsymbol{Z}_i - (\boldsymbol{Y}_i \odot \boldsymbol{Y}_i)(\boldsymbol{Z}_i \odot \boldsymbol{Z}_i)] \\ &+ s_i^2 n \sum [\boldsymbol{Y}_i - (\boldsymbol{Y}_i \odot \boldsymbol{Y}_i)] + t_i^2 m \sum [\boldsymbol{Z}_i - (\boldsymbol{Z}_i \odot \boldsymbol{Z}_i)] \\ &+ 2 r_i s_i \sum [\boldsymbol{Y}_i \boldsymbol{Z}_i - (\boldsymbol{Y}_i \odot \boldsymbol{Y}_i) \boldsymbol{Z}_i] + 2 r_i t_i \sum [\boldsymbol{Y}_i \boldsymbol{Z}_i - \boldsymbol{Y}_i (\boldsymbol{Z}_i \odot \boldsymbol{Z}_i)] , \end{aligned} \tag{9}$$

where $\sum$ denotes the elementwise sum over all entries of the corresponding matrix.

On the other hand, fixing the binary matrices, the continuous coefficients r_i, s_i, t_i, u_i can be optimized in closed form via the convexity of the ℓ_2 norm.

$$[r_i, s_i, t_i, u_i] = -[v_{r_i}, v_{s_i}, v_{t_i}, v_{u_i}] \boldsymbol{H}_{L_{\text{pubo}}}^{-1}, \tag{10}$$

where $\boldsymbol{H}_{L_{\text{pubo}}} \in \mathbb{R}^{4 \times 4}$ and $\boldsymbol{v} \in \mathbb{R}^4$ are the Hessian matrix and the first-order coefficients of Eq. (9) with respect to r_i, s_i, t_i, u_i, respectively. By incorporating the optimization of the scaling factors into a single iteration of the AMFD algorithm for PUBO, we aim to solve the subproblem. The complete procedure is presented in Alg. 2. Using the solution of this subproblem, we then approximate the original real-valued matrix in a greedy manner. This overall approach is described in Alg. 3.

4.3 Post-Training Quantization via BQQ

This subsection presents a model compression technique for deep neural networks (DNNs) based on BQQ. In particular, we focus on ViT-based models, which have recently achieved remarkable success in image processing but still struggle with ultra-low-bit quantization. Quantization methods for ViTs, as well as for general DNNs, can be categorized into two types: Quantization-Aware Training (QAT) [44, 39, 63], which integrates quantization into the training process using labeled data, and Post-Training Quantization (PTQ) [43, 41, 68, 71, 72], which applies quantization to a pretrained model using either unlabeled or limited data. Especially, in situations where training data is unavailable due to privacy constraints or data access limitations, the need to address these challenges has driven interest in data-free quantization techniques [40, 42]. Our work focuses on PTQ under two different scenarios. In the data-free setting, we perform only data-free weight quantization. In the calibration-based setting, we first apply data-free weight quantization, followed by correction

of bias and normalization parameters using a small amount of unlabeled calibration data. Previous studies have explored weight quantization to reduce model size and inference costs, as well as quantization of both weights and activations to further reduce inference costs. In this study, we focus exclusively on weight quantization to clarify the standalone effectiveness of BQQ. Note that, unlike standard first-order quantization, BQQ approximates the original matrix using a combination of binary matrices with altered shapes, as illustrated in Fig. 1.

Data-Free Quantization First, we apply a data-free quantization approach, directly quantizing the weights. Specifically, we formulate weight quantization as an optimization problem that minimizes the squared reconstruction error between a pretrained weight matrix $\mathbf{W}$ and its quantized counterpart via BQQ, as shown in Eq. (7). While prior methods often use channel-wise (i.e., column-wise) scaling factors to maintain accuracy, they result in a large number of scaling parameters. Instead, we adopt a group-wise quantization strategy [65], in which each weight matrix is divided into smaller submatrices, and BQQ is applied independently to each with its own set of scaling factors (see Fig.2). This can reduce the number of scaling parameters, thereby shrinking the model size.

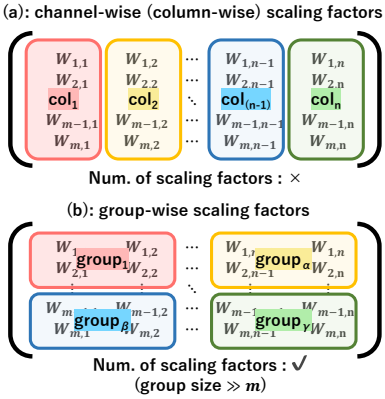

Figure 2: Weight scaling methods.

Correction of Bias and Normalization Parameters After quantizing all weight matrices, we optionally apply a lightweight correction step using a small set of unlabeled calibration inputs. Similar to [5], we refine only the bias and layer normalization parameters–while keeping all other parameters fixed–by minimizing the mean squared error between the output logits of the original f_{org} and quantized models f_{q}, as a form of knowledge distillation:

$\min_{\boldsymbol{\theta}} \|f_{\text{org}}(\boldsymbol{\theta}_{\text{org}}) - f_{\text{q}}(\boldsymbol{\theta})\|_2^2 \,/\, |f_{\text{org}}(\boldsymbol{\theta}_{\text{org}})|$, where $\boldsymbol{\theta}$ denotes the bias and normalization parameters. This correction step compensates for quantization-induced errors and helps recover lost accuracy without requiring full fine-tuning or access to labeled data.

5 Evaluation

Implementation Details As described in Eq. (6), BQQ decomposes a real-valued matrix of size $m \times n$ into binary matrices $\mathbf{Y}_i \in \{0,1\}^{m \times l}$ and $\mathbf{Z}_i \in \{0,1\}^{l \times n}$. To ensure a fair comparison with baseline methods like UQ and BCQ, we fix the intermediate dimension $l = \text{round}(mn/(m+n))$ for all binary matrices. This ensures the total number of binary parameters matches that of UQ and BCQ, making p in Eq. (6) the pseudo bit width. Another way to match the number of binary parameters is to adjust the ratio between the intermediate dimension l and the number of stacks p in Eq. (6); however, this paper adopts the approach described above. Unless otherwise noted, the hyperparameters used in Alg. 3 are set to the following values throughout all experiments: $T_{\text{init}} = 0.2$, $T_{\text{fin}} = 0.005$, $\eta = 0.06$, $\zeta = 4$, and $N_{\text{step}} = 50{,}000$. Also, the scaling factor and the bias for UQ are optimized via grid search to minimize the mean squared error (MSE), as described in App. A.2. For BCQ, we implement the method based on [62], referring to parts of the open-source code provided in [65].

Matrix Data Compression We evaluate the trade-off between approximation error and memory size across five types of real-valued matrices: (i) a random matrix sampled from a Gaussian distribution, (ii) a weight matrix from the DeiT-S model [59], (iii) an inter-city distance matrix from the TSPLIB dataset [55], (iv) a matrix composed of multiple 128-dimensional feature vectors extracted from the SIFT dataset [28], commonly used in ANN search, (v) a red channel matrix of an image from the ImageNet dataset [10]. Each matrix is standardized to have zero mean and a variance of one prior to quantization. We compare nine methods: (1) SVD, a low-rank approximation using SVD; (2) SVD + p-bit UQ, SVD low-rank approximation followed by p-bit UQ of the factorized matrices; (3) UQ; (4) BCQ; (5)VQ, vector quantization where groups of values are clustered using k-means and encoded as indices of a codebook; (6) VQ + p-bit UQ, VQ followed by p-bit UQ of centroids; (7)E_8 LVQ, lattice vector quantization using the E_8 lattice with 240 centroids of norm $\sqrt{2}$, representing each 8-dimensional input as a linear combination of centroids stored in 8 bits; (8) 8-bit UQ + JPEG; a combination of 8-bit UQ and JPEG-style compression [61], where discrete cosine transformation is applied before quantization to exploit spatial redundancy in images; (9) BQQ. The performance is measured in terms of MSE and the memory size of the quantized matrices.

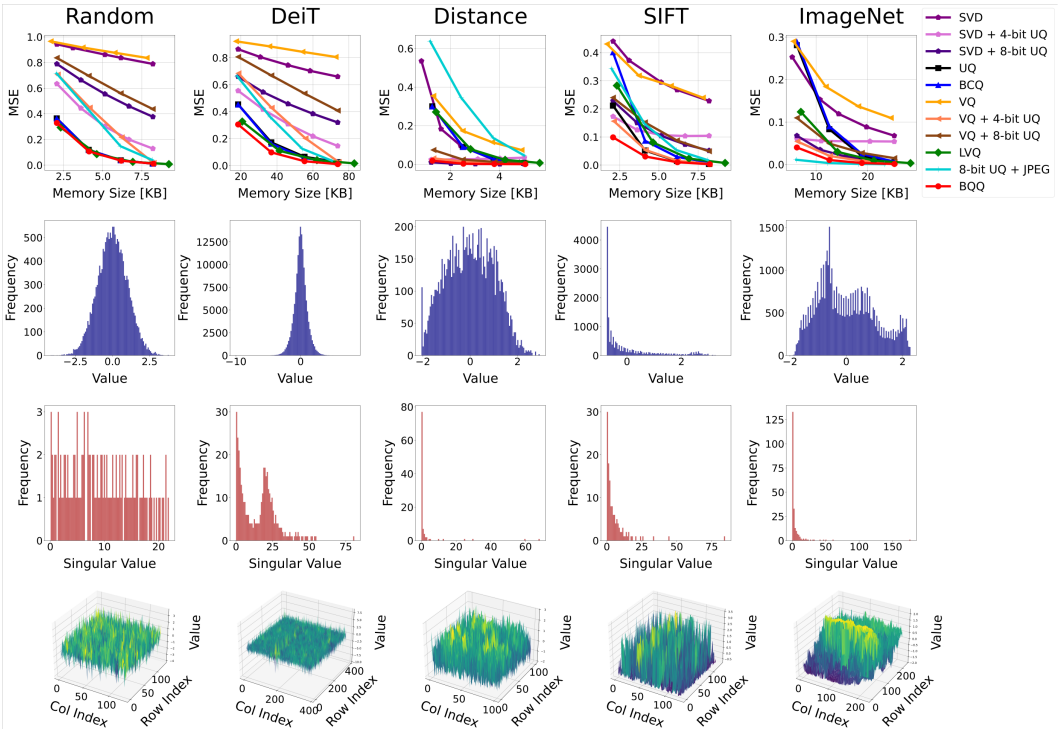

Figure 3: Comparison of the trade-off between reconstruction error (MSE) and memory size for five matrix datasets. Rows 1–4 show the trade-off curves, value distributions, singular value distributions, and 3D plots of the matrices, respectively.

Fig. 3 presents trade-off curves along with visualizations of value distributions, singular value distributions, and 3D surface plots derived from the original matrices. Across all datasets, BQQ consistently achieves a superior trade-off between compression rate and reconstruction accuracy, demonstrating its general effectiveness for matrix data approximation. Notably, the advantage of BQQ over UQ, BCQ, and LVQ becomes especially pronounced for matrices whose singular value spectrum is dominated by a few large components. In contrast, for matrices with relatively flat singular value distributions—i.e., those lacking dominant components—the gain over them is smaller. This suggests that BQQ particularly benefits from matrices with concentrated spectral energy. On the other hand, when compared to SVD, SVD + UQ, VQ, and VQ + UQ, the opposite trend is observed: BQQ exhibits greater advantage for matrices with more uniform singular value spectra. Admittedly, on the ImageNet dataset, BQQ yields a less favorable trade-off between memory size and reconstruction error compared to JPEG. However, since JPEG combines discrete cosine transform with quantization, integrating BQQ with transform-based approaches could potentially lead to further improvements in compression efficiency.

Post-Training Quantization for ViTs Following the methodology outlined in Sec. 4.3, we evaluate the performance of BQQ on pretrained ViTs. Specifically, we compare BQQ with several leading PTQ methods, including COMQ [68], FQ-ViT [43], PTQ4ViT [66], RepQ-ViT [41], ERQ [71], and PSAQ-ViT [40], using two representative ViT architectures: DeiT [59] and Swin Transformer [45]. All methods are tested under the 32-bit activation setting to ensure a fair comparison. Additionally, to appropriately assess the effectiveness of BQQ, we also compare it with variants that use the exact same process but replace the quantization method with UQ or BCQ, as baselines. In our quantization framework, we apply group-wise quantization with submatrices of size 384×384 for DeiTs and 96×96 for Swins based on the first block's channel number. If a weight matrix matches the group size, it is not subdivided further (i.e., layer-wise quantization). As a special case, the final classification layer uses a group size of 100×96. Also, the linear patch embedding layer in DeiT is grouped based on its embedding dimension (384×384 for DeiT-S and 784×784 for DeiT-B), while Swin's patch embedding layer, implemented as a convolutional layer, uses channel-wise UQ instead of BQQ. The same group sizes are applied in the UQ and BCQ baselines for fair comparison. In the case of bias and normalization parameter correction (denoted as c-UQ, c-BCQ, and c-BQQ for each quantization method), we optimize them using the Adam optimizer [34] with a learning rate of 0.001 for 15 epochs

Table 1: Comparison of ImageNet top-1 accuracy across various quantization methods on ViTs.

Method	W bit	Data Free	W scale	Top-1 Accuracy [%]			
				DeiT-S	DeiT-B	Swin-T	Swin-S
COMQ	2	×	column-wise	67.19	77.14	**74.05**	78.02
ERQ	2	×	column-wise	31.95	63.67	45.97	35.44
RepQ-ViT	2	×	column-wise	0.31	0.42	0.12	0.12
c-UQ	2	×	group-wise	52.21	60.57	67.49	74.16
c-BCQ	2	×	group-wise	60.13	73.37	68.09	73.97
c-BQQ	2*	×	group-wise	**69.41**	**77.94**	74.03	**78.47**
PSAQ-ViT	2	✓	column-wise	0.27	0.19	0.15	0.14
UQ	2	✓	group-wise	3.23	2.45	14.69	30.69
BCQ	2	✓	group-wise	10.83	12.99	18.62	34.84
BQQ	2*	✓	group-wise	**58.25**	**72.09**	**57.37**	**68.17**
COMQ	3	×	column-wise	**77.47**	80.47	79.31	**81.95**
ERQ	3	×	column-wise	75.56	79.73	77.99	80.87
RepQ-ViT	3	×	column-wise	58.26	68.80	21.41	69.57
FQ-ViT	3	×	column-wise	51.06	65.64	65.38	71.88
PTQ4ViT	3	×	layer-wise	70.22	75.42	70.74	73.46
c-UQ	3	×	group-wise	72.08	78.85	78.11	80.91
c-BCQ	3	×	group-wise	75.53	79.78	78.60	81.19
c-BQQ	3*	×	group-wise	77.33	**80.81**	**79.34**	81.86
PSAQ-ViT	3	✓	column-wise	52.76	66.40	65.87	72.53
UQ	3	✓	group-wise	42.28	58.56	70.90	75.72
BCQ	3	✓	group-wise	63.46	69.09	72.99	76.18
BQQ	3*	✓	group-wise	**75.61**	**79.90**	**77.33**	**80.36**
COMQ	4	×	column-wise	78.98	81.40	**80.89**	82.85
ERQ	4	×	column-wise	78.95	81.46	80.85	**82.99**
RepQ-ViT	4	×	column-wise	75.39	78.77	75.08	81.53
FQ-ViT	4	×	column-wise	76.23	79.92	78.81	81.89
PTQ4ViT	4	×	layer-wise	77.50	80.07	78.46	80.24
c-UQ	4	×	group-wise	78.15	81.01	80.42	82.40
c-BCQ	4	×	group-wise	78.67	81.22	80.46	82.47
c-BQQ	4*	×	group-wise	**79.12**	**81.47**	80.57	82.72
PSAQ-ViT	4	✓	column-wise	76.59	80.23	79.15	81.94
UQ	4	✓	group-wise	73.53	77.49	79.17	81.47
BCQ	4	✓	group-wise	75.82	78.58	79.63	81.72
BQQ	4*	✓	group-wise	**78.76**	**81.20**	**80.21**	**82.21**
Full Precision	32	-	-	79.83	81.80	81.37	83.21

$*$: Pseudo p^*-bit BQQ has a model size matching that of a p-bit quantized model, despite each matrix being 1 bit.

via a minibatch size of 16, and calibration data are randomly selected from the ImageNet [10] training dataset, with 2048 samples for DeiTs and 1024 samples for Swins, in accordance with [68] setting.

Tab. 1 summarizes the experimental results. Here, **W bit** denotes the bit width for weight quantization, while **W scale** indicates the granularity of scaling factors (e.g., per-layer, per-group, or per-column). Note that although each weight matrix in BQQ is binary, its configuration is designed to match the information content of a first-order p-bit quantized model, which we refer to as pseudo p^*-bit. The results for COMQ [68], FQ-ViT [43], and PTQ4ViT [66] are cited from [68], while those for RepQ-ViT [41], ERQ [71], and PSAQ-ViT [40] were obtained using publicly available implementations. As shown in the experimental results, BQQ consistently achieves SOTA performance regardless of whether calibration data is used, demonstrating a compelling trade-off between accuracy and model size. In particular, it shows notable improvements both in data-free settings and in configurations with model size equivalent to 2-bit quantization. To the best of our knowledge, this is the first study to achieve practically usable accuracy with a model size equivalent to 2-bit quantization in the absence of any data. In addition, while most existing methods preserve accuracy by using column-wise scaling factors–resulting in larger model size–BQQ adopts group-wise scaling, which can reduce parameter overhead. Despite using a more compact scaling strategy, it still achieves competitive accuracy.

6 Discussion

BQQ Effectiveness and Characteristics As shown in the matrix compression experiments, the advantage of BQQ over UQ, BCQ, and LVQ becomes more pronounced for matrices with skewed singular value distributions, where a few dominant singular values capture most of the spectral energy. Conversely, when the singular values are more uniformly distributed, the performance gap between these methods and BQQ becomes smaller. On the other hand, BQQ shows a clear advantage over SVD- and VQ-based methods when the singular value spectrum is relatively flat. This is likely because SVD and VQ are designed to capture and compress redundant patterns in the matrix, which works

well for low-rank or structured data. When such redundancy is absent—as in random-like matrices with weak spectral bias—their performance tends to degrade. Unlike SVD and VQ, UQ, BCQ, and LVQ quantize each element or vector independently to its nearest representative value. This makes it difficult to exploit pattern redundancy, but it also allows these methods to remain relatively stable across different spectral shapes. In fact, when the data lacks significant structure, such independent quantization can lead to more efficient compression than pattern-based approaches. Overall, BQQ integrates the strengths of both pattern-oriented and element-wise quantization strategies. It leverages the ability to capture structural redundancy—similar to SVD and VQ—while also benefiting from the stability and granularity of scalar quantization methods like UQ, BCQ, and LVQ. As a result, it achieves robust compression performance across a wide range of singular value distributions.

In addition, PTQ experiments on ViTs demonstrate that BQQ achieves SOTA performance in both data-free and calibration-based settings. While COMQ or ERQ slightly outperforms it in some cases, they adopt channel-wise quantization with more scaling parameters, whereas our group-wise approach yields a more compact model. Moreover, in contrast to most PTQ methods that optimize discrete parameters by minimizing output error, our approach optimizes binary parameters by simply minimizing the reconstruction error from the original weight matrix (i.e, no PTQ-specific binary variable optimization is performed). Despite this, BQQ matches or even surpasses PTQ-specialized methods, which is a noteworthy outcome. These results are likely due to BQQ's ability to capture structural redundancy often overlooked by first-order methods in overparameterized layers. It is also noteworthy that the matrix multiplication between weights and inputs can be performed using only addition operations, resulting in minimal computational overhead for inference (see App.A.4). This suggests that significant acceleration could be achieved with specialized hardware.

Further Potential and Limitations Despite the demonstrated effectiveness of BQQ, there remains significant room for further improvement. In the current implementation, we adopt a greedy optimization strategy as described in Alg. 3, which is suboptimal from a global perspective. Jointly optimizing all binary matrices and scaling factors could potentially lead to further reductions in quantization error. In addition, although our PTQ framework with BQQ is based on minimizing weight approximation error–except for the correction of bias and normalization parameters–it is generally more effective to minimize output quantization error, as demonstrated in many previous studies. Adapting BQQ to optimize binary matrices with respect to output error could therefore lead to even greater PTQ accuracy. Nevertheless, it is noteworthy that BQQ already achieves SOTA performance. Moreover, in our experiments, the intermediate dimension is fixed, as described in Sec. 5. However, this configuration may not be optimal under a fixed binary parameter budget. Exploring the optimal ratio between the intermediate dimension and the number of binary matrix stacks (p in Eq. (6)) could further improve approximation error (see App. A.9). While BQQ holds such potential, this work has certain limitations. Specifically, while an upper bound on the approximation error is provided (see App. A.10), it does not yet establish a theoretical guarantee that our method outperforms first-order quantization under specific conditions. Additionally, the quantization process still incurs a non-negligible computational cost (see App. A.5). Nonetheless, we believe that BQQ has the capacity to contribute to a wide range of applications beyond the experiments presented in this study, and could have a significant impact.

7 Conclusion

We introduced Binary Quadratic Quantization (BQQ), a novel quantization framework that approximates real-valued matrices as linear combinations of binary matrix products. Across both matrix compression and ViT-based PTQ tasks, BQQ consistently outperforms existing methods in terms of accuracy and compression ratio. These findings highlight the remarkable capability of second-order binary representations in capturing complex structures beyond the reach of first-order schemes, while maintaining an extremely compact data format. By providing an expressive and versatile framework for compressing real-valued matrices using binary bases, BQQ opens new possibilities for building efficient, scalable systems across a wide range of machine learning and information processing applications. We believe this work lays the groundwork for future research into quadratic binary representations and their role in high-performance model compression, retrieval systems, and large-scale learning on massive training data.

Acknowledgements

This work was supported by JST-ALCA-Next (Grant JPMJAN24F3), by JST PRESTO (Grant JPMJPR23P1), and by JST SPRING (Grant JPMJSP2180).

References

[1] Erik Agrell and Thomas Eriksson. Optimization of lattices for quantization. *IEEE Transactions on Information Theory*, 44(5):1814–1828, 1998. doi: 10.1109/18.709865.

[2] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, November 1998. ISSN 0004-5411. doi: 10.1145/293347.293348. URL https://doi.org/10.1145/293347.293348.

[3] Griff Bilbro, Reinhold Mann, Thomas Miller, Wesley Snyder, David van den Bout, and Mark White. Optimization by mean field annealing. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper_files/paper/1988/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper.pdf.

[4] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2019. URL https://api.semanticscholar.org/CorpusID:208290939.

[5] Adrian Bulat, Yassine Ouali, and Georgios Tzimiropoulos. QBB: Quantization with binary bases for LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Kw6MRGFx0R.

[6] Hong Chen, Chengtao Lv, Liang Ding, Haotong Qin, Xiabin Zhou, Yifu Ding, Xuebo Liu, Min Zhang, Jinyang Guo, Xianglong Liu, and Dacheng Tao. DB-LLM: Accurate dual-binarization for efficient LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8719–8730, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.516. URL https://aclanthology.org/2024.findings-acl.516/.

[7] Dmitry A. Chermoshentsev, Aleksei O. Malyshev, Mert Esencan, Egor S. Tiunov, Douglas Mendoza, Alán Aspuru-Guzik, Aleksey K. Fedorov, and Alexander I. Lvovsky. Polynomial unconstrained binary optimisation inspired by optical simulation. 6 2021.

[8] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044, 2019. URL https://api.semanticscholar.org/CorpusID:165163607.

[9] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL https://api.semanticscholar.org/CorpusID:3922816.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[13] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, Sep 1936. ISSN 1860-0980. doi: 10.1007/BF02288367. URL https://doi.org/10.1007/BF02288367.

[14] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

[15] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, USA, 1991. ISBN 0792391810.

[16] Jerry D. Gibson and Khalid Sayood. Lattice quantization. volume 72 of *Advances in Electronics and Electron Physics*, pages 259–330. Academic Press, 1988. doi: https://doi.org/10.1016/S0065-2539(08)60560-0. URL https://www.sciencedirect.com/science/article/pii/S0065253908605600.

[17] Fred Glover, Jin-Kao Hao, and Gary Kochenberger. Polynomial unconstrained binary optimisation – part 1. *Int. J. Metaheuristics*, 1(3):232–256, July 2011. ISSN 1755-2176.

[18] Fred Glover, Jin-Kao Hao, and Gary Kochenberger. Polynomial unconstrained binary optimisation-part 2. *Int. J. Metaheuristics*, 1(4):317–354, December 2011. ISSN 1755-2176. doi: 10.1504/IJMHEUR.2011.044356. URL https://doi.org/10.1504/IJMHEUR.2011.044356.

[19] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, April 1970. ISSN 0945-3245. doi: 10.1007/BF02163027. URL https://doi.org/10.1007/BF02163027.

[20] H. Goto, K. Tatsumura, and A. R. Dixon. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems. *Science Advances*, 5(eaav2372), 2019. doi: 10.1126/sciadv.aav2372.

[21] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura. High-performance combinatorial optimization based on classical mechanics. *Science Advances*, 7(eabe7953), 2021. doi: 10.1126/sciadv.abe7953.

[22] Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. LQ-loRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xw29VvOMmU.

[23] Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen. Network sketching: Exploiting binary structure in deep cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4040–4048. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.430. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.430.

[24] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara, K. Kawarabayashi, and H. Takesue. 100,000-spin coherent Ising machine. *Science Advances*, 7(eabh0952), 2021.

[25] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[26] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue. A coherent Ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016. doi: 10.1126/science.aah4243.

[27] Ernst Ising. Contribution to the Theory of Ferromagnetism. *Z. Phys.*, 31:253–258, 1925. doi: 10.1007/BF02980577.

[28] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011. doi: 10.1109/TPAMI.2010.57. URL https://inria.hal.science/inria-00514462.

[29] Yongkweon Jeon, Chungman Lee, Eulrang Cho, and Yeonju Ro. 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 (CVPR)*, pages 12329–12338, June 2022.

[30] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, May 2011. doi: 10.1038/nature10012.

[31] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

[32] T. Kadowaki and H. Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58:5355–5363, Nov. 1998. doi: 10.1103/PhysRevE.58.5355.

[33] Taro Kanao and Hayato Goto. Simulated bifurcation for higher-order cost functions. *Applied Physics Express*, 16(1), January 2023. ISSN 1882-0778. doi: 10.35848/1882-0786/acaba9. Publisher Copyright: © 2022 The Japan Society of Applied Physics.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2015. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

[35] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, July 2014. doi: 10.1007/s10878-014-9734-0. URL https://ideas.repec.org/a/spr/jcomop/v28y2014i1d10.1007_s10878-014-9734-0.html.

[36] Kyo Kuroki, Thiem Van Chu, Masato Motomura, and Kazushi Kawamura. Annealed mean field descent is highly effective for quadratic unconstrained binary optimization, 2025. URL https://arxiv.org/abs/2504.08315.

[37] Se Jung Kwon, Dongsoo Lee, Yongkweon Jeon, Byeongwook Kim, Bae Seong Park, and Yeonju Ro. Post-training weighted quantization of neural networks for language models, 2021. URL https://openreview.net/forum?id=2Id6XxTjz7c.

[38] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.

[39] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: accurate and fully quantized low-bit vision transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[40] Zhikai Li, Liping Ma, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Patch similarity aware data-free quantization for vision transformers. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, page 154–170, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_10. URL https://doi.org/10.1007/978-3-031-20083-0_10.

[41] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17227–17236, October 2023.

[42] Zhikai Li, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Psaq-vit v2: Toward accurate and general data-free quantization for vision transformers. *IEEE Transactions on Neural Networks and Learning Systems*, 35(12):17227–17238, 2024. doi: 10.1109/TNNLS.2023.3301007.

[43] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1173–1179. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/164. URL https://doi.org/10.24963/ijcai.2022/164. Main Track.

[44] Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. Oscillation-free quantization for low-bit vision transformers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[45] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. URL https://api.semanticscholar.org/CorpusID:232352874.

[46] Hengyuan Ma, Wenlian Lu, and Jianfeng Feng. Efficient combinatorial optimization via heat diffusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=psDrko9v1D.

[47] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, April 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

[48] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *ArXiv*, abs/1609.07843, 2016. URL https://api.semanticscholar.org/CorpusID:16299141.

[49] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. In *10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) 2006, Berlin, Germany, September 18-22. 2006*, 2006.

[50] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *Knowledge and Data Engineering, IEEE Transactions on*, 20:1348–1362, 11 2008. doi: 10.1109/TKDE.2008.53.

[51] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *CoRR*, abs/1603.01025, 2016. URL http://arxiv.org/abs/1603.01025.

[52] Daniel O'Malley, Velimir V. Vesselinov, Boian S. Alexandrov, and Ludmil B. Alexandrov. Nonnegative/binary matrix factorization with a d-wave quantum annealer. *PLOS ONE*, 13(12): e0206653, 2018. doi: 10.1371/journal.pone.0206653. URL https://doi.org/10.1371/journal.pone.0206653.

[53] Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. In *ICLR*, 2024. URL https://openreview.net/forum?id=gLARhFLE0F.

[54] Abraham P. Punnen. *The quadratic unconstrained binary optimization problem : theory, algorithms, and applications*. Springer, Cham, Switzerland, 2022. ISBN 9783031045202.

[55] G. Reinelt. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376–384, 1991.

[56] Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lkx3OpcqSZ.

[57] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. An adversarial winograd schema challenge at scale. 2019. URL https://api.semanticscholar.org/CorpusID:199370376.

[58] Martin Slawski, Matthias Hein, and Pavlo Lutsik. Matrix factorization with binary components. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/226d1f15ecd35f784d2a20c3ecf56d7f-Paper.pdf.

[59] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

[60] Vinay A. Vaishampayan, N. J. A. Sloane, and Sergio D. Servetto. Multiple-description vector quantization with lattice codebooks: Design and analysis. *IEEE Trans. Inf. Theory*, 47:1718–1734, 2001. URL https://api.semanticscholar.org/CorpusID:5327640.

[61] Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, April 1991. ISSN 0001-0782. doi: 10.1145/103085.103089. URL https://doi.org/10.1145/103085.103089.

[62] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S19dR9x0b.

[63] Sheng Xu, Yanjing Li, Teli Ma, Bo-Wen Zeng, Baochang Zhang, Peng Gao, and Jinhu Lv. Tervit: An efficient ternary vision transformer. *ArXiv*, abs/2201.08050, 2022. URL https://api.semanticscholar.org/CorpusID:246063758.

[64] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, XIAOPENG ZHANG, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Representation Learning*, volume 2024, pages 52401–52418, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e6c2e85db1f1039177c4495ccd399ac4-Paper-Conference.pdf.

[65] Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, and Yingyan Celine Lin. ShiftaddLLM: Accelerating pretrained LLMs via post-training multiplication-less reparameterization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=JNl6h3U3oW.

[66] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XII*, page 191–207, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19774-1. doi: 10.1007/978-3-031-19775-8_12. URL https://doi.org/10.1007/978-3-031-19775-8_12.

[67] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:159041722.

[68] Aozhong Zhang, Zi Yang, Naigang Wang, Yingyong Qi, Jack Xin, Xin Li, and Penghang Yin. Comq: A backpropagation-free algorithm for post-training quantization, 2024. URL https://arxiv.org/abs/2403.07134.

[69] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 391–400, 2007. doi: 10.1109/ICDM.2007.99.

[70] Zhenghao Zhao, Yuzhang Shang, Junyi Wu, and Yan Yan. Dataset quantization with active learning based adaptive sampling. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LX*, page 346–362, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73026-9. doi: 10.1007/978-3-031-73027-6_20. URL https://doi.org/10.1007/978-3-031-73027-6_20.

[71] Yunshan Zhong, Jiawei Hu, You Huang, Yuxin Zhang, and Rongrong Ji. Erq: error reduction for post-training quantization of vision transformers. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[72] Yunshan Zhong, You Huang, Jiawei Hu, Yuxin Zhang, and Rongrong Ji. Towards accurate post-training quantization of vision transformers via error reduction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47(4):2676–2692, January 2025. ISSN 0162-8828. doi: 10.1109/TPAMI.2025.3528042. URL https://doi.org/10.1109/TPAMI.2025.3528042.

[73] Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17205–17216, October 2023.

Appendix

A.1 Derivation of Equation (4)

In order to extend the QUBO formulation, which is the domain of application for AMFD, to a general PUBO formulation, we prove that the KL divergence between the mean-field approximate distribution and the canonical distribution for the PUBO formulation is given by Eq. (4).

Proof.
From the definition of KL divergence:

$$\begin{aligned} D_{\mathrm{KL}}\left(P_{\mathrm{MF}}\left(\boldsymbol{s}\right) \mid\mid P_{\mathrm{C}}\left(\boldsymbol{s}\right)\right) &= \sum_{\boldsymbol{s}} P_{\mathrm{MF}}\left(\boldsymbol{s}\right) \ln\left(\frac{P_{\mathrm{MF}}\left(\boldsymbol{s}\right)}{P_{\mathrm{C}}\left(\boldsymbol{s}\right)}\right) \\ &= \sum_{\boldsymbol{s}} \left[\left(\prod_{i=1}^{N} p_i\left(s_i\right)\right)\left(\frac{1}{T}L\left(\boldsymbol{s}\right) + \ln\left(Z\right) + \sum_{i=1}^{N} \ln\left(p_i\left(s_i\right)\right)\right)\right]. \end{aligned} \tag{S.1}$$

For the first term in Eq. (S.1):

$$\begin{aligned} &\sum_{\boldsymbol{s}} \left[\left(\prod_{j=1}^{N} p_j\left(s_j\right)\right) L(\boldsymbol{s})\right] \\ &= \sum_{\boldsymbol{s}} \left[\left(\prod_{j=1}^{N} p_j\left(s_j\right)\right)\left(\sum_{i_1} J^{(1)}_{i_1} s_i + \sum_{i_1<i_2} J^{(2)}_{i_1 i_2} s_{i_1} s_{i_2} + \sum_{i_1<i_2<i_3} J^{(3)}_{i_1 i_2 i_3} s_{i_1} s_{i_2} s_{i_3} + \cdots\right)\right] \\ &= \sum_{\boldsymbol{s}} \left(\prod_{j\neq i_1} p_j\left(s_j\right) \sum_{i_1} J^{(1)}_{i_1} s_{i_1} p_{i_1}\left(s_{i_1}\right)\right) + \sum_{\boldsymbol{s}} \left(\prod_{j\neq i_1,i_2} p_j\left(s_j\right) \sum_{i_1<i_2} J^{(2)}_{i_1 i_2} s_{i_1} p_{i_1}\left(s_{i_1}\right) s_{i_2} p_{i_2}\left(s_{i_2}\right)\right) \\ &+ \sum_{\boldsymbol{s}} \left(\prod_{j\neq i_1,i_2,i_3} p_j\left(s_j\right) \sum_{i_1<i_2<i_3} J^{(3)}_{i_1 i_2 i_3} s_{i_1} p_{i_1}\left(s_{i_1}\right) s_{i_2} p_{i_2}\left(s_{i_2}\right) s_{i_3} p_{i_3}\left(s_{i_3}\right)\right) + \cdots \\ &= \left(\prod_{j\neq i_1} \sum_{s_j=0}^{1} p_j\left(s_j\right)\right) \sum_{i_1} J^{(1)}_{i_1} \sum_{s_{i_1}=0}^{1} s_{i_1} p_{i_1}\left(s_{i_1}\right) \\ &+ \left(\prod_{j\neq i_1,i_2} \sum_{s_j=0}^{1} p_j\left(s_j\right)\right)\left(\sum_{i_1<i_2} J^{(2)}_{i_1 i_2} \sum_{s_{i_1}=0}^{1} s_{i_1} p_{i_1}\left(s_{i_1}\right) \sum_{s_{i_2}=0}^{1} s_{i_2} p_{i_2}\left(s_{i_2}\right)\right) \\ &+ \left(\prod_{j\neq i_1,i_2,i_3} \sum_{s_j=0}^{1} p_j\left(s_j\right)\right)\left(\sum_{i_1<i_2<i_3} J^{(3)}_{i_1 i_2 i_3} \sum_{s_{i_1}=0}^{1} s_{i_1} p_{i_1}\left(s_{i_1}\right) \sum_{s_{i_2}=0}^{1} s_{i_2} p_{i_2}\left(s_{i_2}\right) \sum_{s_{i_3}=0}^{1} s_{i_3} p_{i_3}\left(s_{i_3}\right)\right) \cdots \\ &= \sum_{i=1}^{N} J^{(1)}_{i_1} x_i + \sum_{i=1}^{N}\sum_{j<i} J^{(2)}_{i_1 i_2} x_i x_j + \sum_{i=1}^{N}\sum_{j<i}\sum_{k<j} J^{(3)}_{i_1 i_2 i_3} x_i x_j x_k \cdots + \sum_{i_1<i_2<\cdots<i_k} J^{(k)}_{i_1 i_2 \cdots i_k} \prod_{j=1}^{k} x_{i_j} \\ &= L(\boldsymbol{x}). \end{aligned} \tag{S.2}$$

For the second term in Eq. (S.1):

$$\begin{aligned} &\sum_{\boldsymbol{s}} \left(\prod_{i=1}^{N} p_i\left(s_i\right)\right) \ln(Z) \\ &= \left(\prod_{i=1}^{N} \sum_{s_i=0}^{1} p_i\left(s_i\right)\right) \ln(Z) \\ &= \ln(Z) \quad \left(\because \sum_{s_j=0}^{1} p_j\left(s_j\right) = 1\right). \end{aligned} \tag{S.3}$$

For the third term in Eq. (S.1):

$$
\begin{aligned}
&\sum_{\boldsymbol{s}}\left[\left(\prod_{i=1}^{N} p_i\left(s_i\right)\right)\sum_{i=1}^{N}\ln\left(p_i\left(s_i\right)\right)\right] \\
&=\sum_{\boldsymbol{s}}\left[\sum_{i=1}^{N}\left(\prod_{j\neq i} p_j\left(s_j\right)\right)p_i\left(s_i\right)\ln\left(p_i\left(s_i\right)\right)\right] \\
&=\sum_{i=1}^{N}\left(\prod_{j\neq i}\sum_{s_j=0}^{1} p_j\left(s_j\right)\right)\sum_{s_i=0}^{1} p_i\left(s_i\right)\ln\left(p_i\left(s_i\right)\right) \\
&=\sum_{i=1}^{N}\sum_{s_i=0}^{1} p_i\left(s_i\right)\ln\left(p_i\left(s_i\right)\right) \quad \left(\because \sum_{s_j=0}^{1} p_j\left(s_j\right)=1\right) \\
&=\sum_{i=1}^{N}\left[x_i\ln x_i+(1-x_i)\ln(1-x_i)\right] \quad \left(\because x_i=\sum_{s_i=0}^{1} s_i p_i(s_i)=p_i(1)\right). \qquad \text{(S.4)}
\end{aligned}
$$

From Eq. (S.1)–(S.4), we obtain the KL divergence expression in Eq. (4). Therefore, the claim is proven. □

A.2 Uniform Quantization with Grid Search

Here, we provide a detailed explanation of UQ algorithm introduced in Sec. 5. As shown in Alg. S.1, the scaling factor is determined via grid search so as to minimize the MSE between the original matrix and the quantized matrix. We optimized with $N_{\text{split}} = 100$ for all experiments.

Algorithm S.1 MSE-Aware Uniform Quantization with Grid Search

Input: Input matrix $\boldsymbol{W}$, bit width N_{bit}, the number of grid divisions N_{split}
Output: Quantized matrix $\boldsymbol{Q}$, scaling factor a, bias b, dequantized matrix $\boldsymbol{W}_q$

$L \leftarrow 2^{N_{\text{bit}}}$ ▷ The number of quantization levels
$\mu \leftarrow \text{mean}(\boldsymbol{W})$ ▷ Search range setting
$w_{\min} \leftarrow \min(\boldsymbol{W})$
$w_{\max} \leftarrow \max(\boldsymbol{W})$
Initialize $\varepsilon \leftarrow \infty$, $\boldsymbol{Q} \leftarrow$ None, $(\alpha, \beta) \leftarrow$ None
for each $r_{\max} \in \text{linspace}(\mu, w_{\max}, N_{\text{split}})$ **do**
  for each $r_{\min} \in \text{linspace}(w_{\min}, \mu, N_{\text{split}})$ **do**
    $\boldsymbol{W}_c \leftarrow \text{clip}(\boldsymbol{W}, r_{\min}, r_{\max})$ ▷ Clip $\boldsymbol{W}$ to $[r_{\min}, r_{\max}]$
    $\boldsymbol{Q}_c \leftarrow \left\lfloor \frac{\boldsymbol{W}_c - r_{\min}}{r_{\max} - r_{\min}} \cdot (L-1) \right\rceil$ ▷ Quantize to $[0, L-1]$
    $\boldsymbol{W}_c \leftarrow \frac{\boldsymbol{Q}_c}{L-1} \cdot (r_{\max} - r_{\min}) + r_{\min}$ ▷ Dequantize
    $e \leftarrow \frac{1}{|\boldsymbol{W}|}\sum(\boldsymbol{W} - \boldsymbol{W}_c)^2$ ▷ Compute quantization error
    if $e < \varepsilon$ **then**
      $\varepsilon \leftarrow e$ ▷ Save best quantization setting
      $\boldsymbol{Q} \leftarrow \boldsymbol{Q}_c$
      $a \leftarrow \frac{r_{\max} - r_{\min}}{L-1}$, $b \leftarrow r_{\min}$
      $\boldsymbol{W}_q \leftarrow \boldsymbol{W}_c$
return $\boldsymbol{Q}$, a, b, $\boldsymbol{W}_q$

A.3 E_8 Lattice Vector Quantization Using 240 Centroids of Norm $\sqrt{2}$

Here, we describe the E_8 LVQ algorithm introduced in Sec. 5. In this work, we construct a codebook consisting of all 240 E_8 lattice vectors with Euclidean norm $\sqrt{2}$ as centroids. For a given 8-dimensional vector, the nearest centroid is selected to approximate it. To minimize the MSE, after selecting the centroid based on cosine similarity, the optimal scalar scaling factor is computed. Each 8-dimensional vector is thus associated with a single 8-bit index, which is equivalent to approximately one bit of scalar quantization. By iteratively applying this procedure to the residual errors, the

quantization can be extended to multiple bits. However, maintaining a separate scaling factor for each 8-dimensional vector would make the memory overhead non-negligible. Therefore, in this work, the scaling factors are quantized using 2-bit uniform quantization, which was empirically found to achieve the best trade-off between memory efficiency and quantization error. The resulting residual-based multi-bit LVQ algorithm is summarized in Alg. S.2.

Algorithm S.2 E_8 Lattice Vector Quantization with Residual Greedy Search

Input: Input matrix $\mathbf{W} \in \mathbb{R}^{M \times N}$, number of quantization bits N_{bits}, scale bits s_{bits}
Output: Quantized and reconstructed matrix $\mathbf{W_q}$, code vector indices $\mathbf{k}^*$, scaling factors $\boldsymbol{\alpha}$

Flatten $\mathbf{W}$ to $\mathbf{W}_{\text{flat}}$
Pad $\mathbf{W}_{\text{flat}}$ with zeros so that its length is a multiple of 8
Reshape $\mathbf{W}_{\text{flat}}$ to $\mathbf{D} \in \mathbb{R}^{n \times 8}$
Construct the E_8 codebook $\mathbf{C} \in \mathbb{R}^{240 \times 8}$ consisting of all lattice vectors with norm $\sqrt{2}$
Initialize reconstruction $\mathbf{D}_{\text{total}} \leftarrow 0$
for bit $= 1$ to N_{bits} **do**
    Normalize codebook and data: $\mathbf{C}_{\text{norm}} = \mathbf{C}/\|\mathbf{C}\|_2$, $\mathbf{D}_{\text{norm}} = \mathbf{D}/\|\mathbf{D}\|_2$
    Compute cosine similarity matrix:

$$\mathbf{S} = \mathbf{D}_{\text{norm}} \mathbf{C}_{\text{norm}}^{\top}$$

    Select closest code vector index for each row:

$$k_r^* = \arg \max_{k=1,\ldots,240} S_{r,k}, \quad r = 1, \ldots, n$$

    Select corresponding codes: $\mathbf{C}_{\text{selected}} \leftarrow \mathbf{C}[k^*]$
    Compute scalar coefficients $\boldsymbol{\alpha} \in \mathbb{R}^n$ for each row of $\mathbf{D}$: $\alpha_r = \dfrac{\sum_{j=1}^{8} D_{r,j}\, C_{\text{selected},r,j}}{\sum_{j=1}^{8} C_{\text{selected},r,j}^2}$
    Quantize $\boldsymbol{\alpha}$ using uniform quantization with s_{bits}
    Reconstruct partial matrix: $\mathbf{D}_{\text{hat}} \leftarrow \boldsymbol{\alpha} \cdot \mathbf{C}_{\text{selected}}$
    Update residual: $\mathbf{D} \leftarrow \mathbf{D} - \mathbf{D}_{\text{hat}}$
    Accumulate reconstruction: $\mathbf{D}_{\text{total}} \leftarrow \mathbf{D}_{\text{total}} + \mathbf{D}_{\text{hat}}$
Reshape $\mathbf{D}_{\text{total}}$ to original shape of $\mathbf{W}$: $\mathbf{W_q} \leftarrow \mathbf{D}_{\text{total}}$
return $\mathbf{W_q}, \mathbf{k}^*, \boldsymbol{\alpha}$

A.4 Inference Computational Cost Analysis of the BQQ Layer in DNN

Since p-bit quantization typically incurs approximately p times the computational cost of 1-bit quantization, we analyze the computational cost based on the 1-bit case for both the conventional first-order quantization and the proposed BQQ layer. We focus on a single linear layer, which is where weights in ViTs are concentrated. Let the input be a real-valued matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, and the 1-bit quantized weight for the first-order baseline be $\mathbf{W}_q \in \{0,1\}^{m \times n}$. For the BQQ method, the weights are represented as $\mathbf{Y} \in \{0,1\}^{m \times l}$ and $\mathbf{Z} \in \{0,1\}^{l \times n}$, where l is the intermediate rank.

Here, `AND` refers to bitwise operations between binary weights and real-valued inputs (not binary-binary operations), `ADD` denotes real-valued addition, and `MUL` represents real-valued multiplication. As described in the experimental setting in Sec. 5, we set $l = \text{round}\left(\frac{mn}{m+n}\right)$ for the BQQ layer. Note that the cost of computing the zero-point bias is omitted because it is a common term and does not change as the bit width increases.

First-Order 1-Bit Quantization

$$\text{OUT}_{\text{FOQ}} = a\mathbf{W}_q\mathbf{X}$$

The matrix multiplication involves the following computational cost:

$$\text{COST}_{\text{FOQ}} = mnd\,\texttt{AND} + md(n-1)\,\texttt{ADD} + md\,\texttt{MUL}$$

BQQ 1*-Bit Quantization The output is computed as:

$$\text{OUT}_{\text{BQQ}} = (r\mathbf{YZ} + s\mathbf{Y}\mathbf{1}_Z + t\mathbf{1}_Y\mathbf{Z})\,\mathbf{X}$$

The computational steps are broken down as follows:

$$\begin{aligned}
\mathbf{A} &\leftarrow \mathbf{ZX} : lnd\,\texttt{AND} + (n-1)ld\,\texttt{ADD} \\
\mathbf{B} &\leftarrow \mathbf{1}_Z\mathbf{X} : (n-1)d\,\texttt{ADD} \\
\mathbf{C} &\leftarrow \mathbf{Y}(r\mathbf{A} + s\mathbf{B}) : 2ld\,\texttt{MUL} + ld\,\texttt{ADD} + mld\,\texttt{AND} + (l-1)md\,\texttt{ADD} \\
\mathbf{D} &\leftarrow \mathbf{1}_Y \cdot t\mathbf{A} : (l-1)d\,\texttt{ADD} + ld\,\texttt{MUL} \\
\text{Out} &\leftarrow \mathbf{C} + \mathbf{D} : md\,\texttt{ADD}
\end{aligned}$$

The total computational cost becomes:

$$\text{COST}_{\text{BQQ}} = ld(m+n)\,\texttt{AND} + d[(m+n+1)l + n - m - 2]\,\texttt{ADD} + 3ld\,\texttt{MUL}$$

Relative Cost Ratio We compare the computational cost between the first-order 1-bit quantization and the BQQ method. The relative ratio is given by:

$$\begin{aligned}
\frac{\text{COST}_{\text{BQQ}}}{\text{COST}_{\text{FOQ}}} &= \frac{ld(m+n)\,\texttt{AND} + d[(m+n+1)l + n - m - 2]\,\texttt{ADD} + 3ld\,\texttt{MUL}}{mnd\,\texttt{AND} + md(n-1)\,\texttt{ADD} + md\,\texttt{MUL}} \\
&= 1 + \frac{d[(n+l-2)\,\texttt{ADD} + (3l-m)\,\texttt{MUL}]}{mnd\,\texttt{AND} + md(n-1)\,\texttt{ADD} + md\,\texttt{MUL}}, \quad \left(\text{where } l = \frac{mn}{m+n}\right)
\end{aligned}$$

yielding a computational complexity ratio of $\mathcal{O}(1)$, since the first-order quantization has $\mathcal{O}(mnd)$ operations while the BQQ method has $\mathcal{O}((m+n)ld)$ operations, and for $l \approx \frac{mn}{m+n}$, their ratio becomes of order one:

$$\frac{\text{COST}_{\text{BQQ}}}{\text{COST}_{\text{FOQ}}} = \mathcal{O}\left(\frac{(m+n)ld}{mnd}\right) = \mathcal{O}(1), \quad \text{when } l \approx \frac{mn}{m+n}.$$

Practical Examples For the DeiT model in Sec. 5 with $m = n = 384$, $l = 192$:

$$1 + \frac{574\,\texttt{ADD} + (3 \cdot 192 - 384)\,\texttt{MUL}}{384^2\,\texttt{AND} + 384 \cdot 383\,\texttt{ADD} + 384\,\texttt{MUL}} < 1.0052$$

For the Swin model with $m = n = 96$, $l = 48$:

$$1 + \frac{142\,\texttt{ADD} + (3 \cdot 48 - 96)\,\texttt{MUL}}{96^2\,\texttt{AND} + 96 \cdot 95\,\texttt{ADD} + 96\,\texttt{MUL}} < 1.0207$$

These results demonstrate that the inference computational cost of the BQQ layer is nearly equivalent to that of the conventional first-order 1-bit quantization.

A.5 BQQ Execution Time for PTQ

We report the quantization time required by our proposed method, BQQ, under a data-free setting. All experiments were conducted using the following environment:

- Python 3.9.19
- PyTorch 2.6.0 with CUDA 12.4
- Four NVIDIA GeForce RTX 4090 GPUs
- AMD EPYC 7313 16-Core Processor

During quantization, we parallelized the process by assigning each matrix to a separate GPU thread, enabling concurrent quantization of multiple layers. Quantization time was measured using Python's `time` module. We evaluated quantization time using a small model (DeiT-S, 22M parameters) and a large model (DeiT-B, 86M parameters).

Tab. S.1 summarizes the computation time for pseudo 2*-bit, 3*-bit, and 4*-bit quantization using BQQ. The reported times indicate the total time required to quantize the entire model. Note that the larger the pseudo bit width, the longer it takes because of greedy optimization for each pseudo bit index, as shown in Alg. 3 Since processing is performed in parallel on each GPU, speedup is possible by increasing the number of GPUs, but computation time is a barrier to scaling up the applicability of PTQ to large-scale models such as large language models.

Table S.1: Execution time of BQQ on DeiT-S and DeiT-B.

Model	#layers	#parameters	BQQ Time [min]		
			2* bit	3* bit	4* bit
DeiT-S	12	22M	13	17	21
DeiT-B	12	86M	32	45	57

A.6 Effect of N_{step} on Accuracy and Computation Time

We investigate the effect of the number of optimization steps N_{step} (and the total quantization time) on the final ImageNet top-1 accuracy and computation time under the data-free quantization setting. The results on DeiT-Small (DeiT-S) and DeiT-Base (DeiT-B) models are shown in Fig. S.1. As expected, increasing N_{step} generally results in longer quantization time but also enables more precise optimization, which tends to improve final accuracy. In particular, it is noteworthy that increasing N_{step} beyond 50,000—the setting used in the main manuscript—yields even more accurate results. For instance, with DeiT-S, an accuracy of 60.24% is achieved at pseudo 2*-bit precision, which is 1.99% higher than the accuracy reported in the main manuscript. Similarly, with DeiT-B, an accuracy of 72.91% is obtained at pseudo 2*-bit precision, representing a 0.82% improvement. As shown in the figure, in highly compressed settings such as pseudo 2*-bit quantization, a large N_{step} is essential to prevent significant accuracy degradation. In contrast, for pseudo 3*-bit or 4*-bit quantization, accuracy remains relatively stable even with smaller N_{step} values.

These observations suggest that more aggressive compression schemes are more sensitive to the quality of optimization, as even small quantization errors can have a greater impact on final accuracy. Therefore, more rigorous optimization is required in such cases. Notably, the time required for accuracy to reach saturation appears largely independent of the pseudo bit-width, with convergence observed in approximately 10 minutes for DeiT-S and 20 minutes for DeiT-B—durations that are not prohibitive in practical applications.

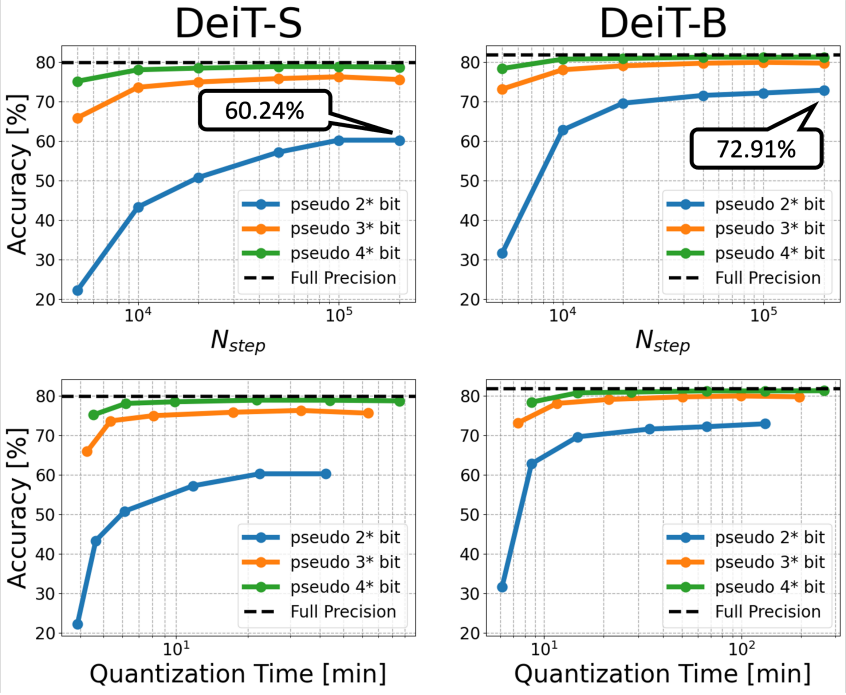

Figure S.1: Effect of the number of optimization steps N_{step} (top) and total quantization time (bottom) on final ImageNet top-1 accuracy in the data-free quantization setting.

A.7 Accuracy Under Extreme Compression

We present additional results beyond those in the main manuscript, focusing on further compression–specifically, setting the model size to pseudo 1.5* bit. In the main experiments, the factorization was performed using binary matrices $\boldsymbol{Y}_i \in \{0,1\}^{m\times l}$ and $\boldsymbol{Z}_i \in \{0,1\}^{l\times n}$, where the intermediate dimension was set to $l = \text{round}\left(\frac{mn}{m+n}\right)$. Here, we increase the intermediate dimension to $l = \text{round}\left(1.5 \cdot \frac{mn}{m+n}\right)$ (i.e., 1.5 times the original parameter count), and fix the number of stacks (denoted as p in Eq. (6)) to 1. This results in an effectively 1.5-bit quantized model, noting that all weight elements remain binary (1-bit). Tab. S.2 reports the top-1 accuracy on ImageNet after compression. While the accuracy significantly deteriorates in the data-free setting, it is remarkable that with only a small amount of calibration data, the accuracy remains reasonably high. Notably, for DeiT-B, the top-1 accuracy exceeds 70%, demonstrating that even under such severe compression, practical accuracy can be retained—an impressive outcome.

Table S.2: ImageNet top-1 accuracy of BQQ under extreme compression.

Method	**W bit**	**Data Free**	**ImageNet Top-1 Accuracy [%]**			
			DeiT-S	**DeiT-B**	**Swin-T**	**Swin-S**
c-BQQ	1.5*	×	53.67	71.36	61.25	69.89
BQQ	1.5*	✓	7.41	35.21	10.82	20.06

A.8 Evaluation on Language Models

Here, we evaluate our quantization methods on several compact language models that are well-suited for edge deployment: Qwen2.5-0.5B, Qwen2.5-1.5B, and DeepSeek-R1-Distill-Qwen1.5B. We compare three approaches: (1) BQQ, which performs data-free quantization; (2) tuned-BQQ (t-BQQ), which first applies BQQ and then fine-tunes only the continuous parameters; and (3) GPTQ [14], a standard post-training quantization (PTQ) method. For both t-BQQ and GPTQ, the calibration data consist of the full training split of WikiText-2.

Tab. S.3 reports the perplexity on WikiText-2 [48] and the accuracy on six downstream tasks: PIQA [4], Winogrande (WinoG) [57], ARC-Easy (ArcE) and ARC-Challenge (ArcC) [9], HellaSwag (HellaS) [67], and BoolQ [8]. The table also includes the average accuracy across all tasks. BQQ achieves higher average accuracy than GPTQ on the 1.5B models, particularly under low-bit settings (2-bit and 3-bit). Notably, even without any calibration data, BQQ attains comparable or superior performance to GPTQ. In contrast, for the smaller 0.5B model, BQQ struggles to maintain accuracy. We attribute this to the model-dependent discrepancy between the quantization distributions that minimize activation error and those that minimize weight error. When this discrepancy becomes large, the performance of BQQ tends to degrade.

Although the performance of neural network quantization is not always guaranteed to be superior, we emphasize that the current work proposes BQQ as a general binary quantization framework rather than a dedicated quantization technique for neural networks. In our implementation, BQQ quantizes weights by minimizing reconstruction error in the weight space, without relying on output-based error signals that are commonly used in many neural network quantization methods. This implies that BQQ does not yet exploit task-specific loss functions or activation statistics. We therefore believe that adapting BQQ to incorporate such feedback—especially minimizing the downstream output error—could lead to significant further improvements in model performance, and this represents a highly promising avenue for future research.

A.9 Quantization Error vs. Binary Matrix Stack-to-Intermediate Dimension Ratio

In the main manuscript, we fixed the intermediate dimension as $l = \text{round}\left(\frac{mn}{m+n}\right)$ for all experiments. However, this setting is not necessarily optimal. The same compression ratio can also be achieved by varying the number of stacked binary matrices and the intermediate dimension. In this subsection, we investigate how the quantization error (MSE) changes with respect to the number of stacks p and

Table S.3: WikiText-2 perplexity and downstream task accuracy.

Model	Method	Bit	PPL	ArcE	ArcC	BoolQ	HellaS	PiQA	WinoG	Avg.
Qwen2.5-0.5B	BQQ	2	17106.9	28.3	20.3	37.8	26.3	54.0	50.5	36.2
	t-BQQ	2	**49.5**	**37.1**	**24.2**	39.7	**29.9**	**56.4**	**52.1**	**39.9**
	GPTQ	2	4392.5	25.3	21.3	**43.2**	25.7	52.7	49.0	36.2
	BQQ	3	1808.8	31.5	19.1	38.0	26.5	54.4	50.5	36.7
	t-BQQ	3	**19.4**	45.2	**27.2**	43.0	**39.7**	**62.1**	**56.1**	45.6
	GPTQ	3	21.8	**45.8**	21.0	**56.5**	34.0	**63.6**	56.0	**46.1**
	BQQ	4	71.7	38.8	21.6	43.9	30.5	61.2	51.2	41.2
	t-BQQ	4	14.7	51.4	**30.5**	**52.7**	**45.6**	**66.6**	54.3	50.2
	GPTQ	4	**14.4**	**62.9**	27.9	**57.4**	38.9	**68.2**	**56.2**	**51.9**
	baseline	16	13.1	64.6	29.5	58.8	40.6	70.2	56.4	53.4
Qwen2.5-1.5B	BQQ	2	688.3	33.8	19.1	57.7	27.2	56.6	51.9	41.0
	t-BQQ	2	**23.6**	**45.1**	**28.5**	**62.0**	**40.2**	**61.2**	**53.8**	**48.5**
	GPTQ	2	922.4	26.0	21.1	45.0	26.0	51.6	52.2	37.0
	BQQ	3	14.6	**63.6**	32.0	**63.9**	41.7	71.1	57.9	55.0
	t-BQQ	3	**11.8**	61.8	**38.3**	60.6	**57.0**	**72.0**	**59.0**	**58.1**
	GPTQ	3	12.1	59.5	30.0	**60.7**	43.6	70.0	57.1	53.5
	BQQ	4	10.5	72.4	38.5	69.1	47.7	74.8	60.7	60.5
	t-BQQ	4	9.9	73.0	**45.1**	65.5	**62.9**	73.8	61.3	**63.6**
	GPTQ	4	**9.7**	**74.5**	40.0	**70.8**	49.3	**75.2**	**63.1**	62.1
	baseline	16	9.3	71.5	45.1	73.0	67.8	76.1	63.4	66.1
Deepseek-R1-Distill Qwen-1.5B	BQQ	2	921.2	29.8	18.8	47.9	26.5	53.6	49.3	37.6
	t-BQQ	2	**46.1**	**41.1**	**22.7**	**52.5**	**31.0**	**58.8**	**52.6**	**43.1**
	GPTQ	2	872.4	26.6	20.2	47.6	25.7	52.9	49.1	37.0
	BQQ	3	59.2	**56.1**	30.6	51.2	33.8	**63.2**	52.9	48.0
	t-BQQ	3	**28.5**	50.7	**32.9**	**59.3**	**41.2**	61.7	**55.7**	**50.3**
	GPTQ	3	59.6	52.3	25.6	54.0	32.8	62.4	51.9	46.5
	BQQ	4	**39.4**	**59.3**	31.8	66.5	36.0	64.8	**57.5**	**52.7**
	t-BQQ	4	**21.7**	54.1	31.9	65.2	**43.9**	63.8	56.0	52.5
	GPTQ	4	43.4	58.5	**32.2**	**66.9**	36.0	**65.9**	56.3	52.6
	baseline	16	40.4	56.1	34.6	68.6	44.8	65.8	55.6	54.2

the normalized intermediate dimension l_{scale}, where the actual intermediate dimension is defined as $l = \text{round}\left(l_{\text{scale}} \frac{mn}{m+n}\right)$. We conducted experiments by sweeping over p and l_{scale} and measuring the resulting MSE.

Tab. S.4 summarizes the experimental results. These results show that the optimal balance between the number of stacks and the intermediate dimension varies across datasets, indicating that fixing $l = \text{round}\left(\frac{mn}{m+n}\right)$ is not always optimal. Therefore, adaptively determining this ratio can potentially lead to more efficient compression.

A.10 Theoretical Upper Bound of BQQ

In this subsection, we provide a theoretical analysis of the approximation error inherent in the BQQ formulation. To derive a concrete upper bound, we consider a particular case of Eq. (5) by setting $\beta_i = -0.5\alpha_i$, $\delta_i = 1$, and $\gamma_i = -0.5$. Although this specific case does not yield a closed-form optimal solution, an approximate one can be obtained by aligning the binary components with the sign patterns of the singular vectors obtained from SVD, combined with appropriate scaling. This leads to a theoretical upper bound on the square root of the subproblem error, denoted by $\sqrt{L_{\text{sub}}^{(i)}}$ (Eq. (8)). The overall error of BQQ is then obtained by summing over all stack indices i: BQQ total error $= \sum_i L_{\text{sub}}^{(i)}$. Based on the Eckart–Young–Mirsky theorem and the triangle inequality,

Table S.4: Quantization error (MSE) with varying the number of stacked binary matrices p and the intermediate dimension scaling l_{scale}. Note that the MSE values are scaled by a factor of 10^3.

		Random		DeiT		Distance		SIFT		ImageNet	
#stacks (p)	**l_{scale}**	**Size [KB]**	**MSE**	**Size [KB]**	**MSE**	**Size [KB]**	**MSE**	**Size [KB]**	**MSE**	**Size [KB]**	**MSE**
1	1	2.1	**324.3**	18.4	**298.1**	1.3	14.6	2.1	**97.8**	6.3	42.7
2	0.5	2.1	329.9	18.5	309.4	1.3	**9.2**	2.1	108.1	6.3	**29.7**
4	0.25	2.1	337.1	18.5	317.3	1.3	12.2	2.1	110.8	6.3	49.6
1	2	4.1	106.4	36.9	101.9	2.5	7.7	4.1	46.3	12.6	20.5
2	1	4.1	**105.3**	36.9	**95.8**	2.5	**2.3**	4.1	**30.0**	12.6	10.9
4	0.5	4.1	108.4	36.9	100.8	2.6	2.5	4.1	34.4	12.6	**8.9**
8	0.25	4.2	114.2	37.0	105.6	2.5	3.5	4.2	35.9	12.6	14.1
2	1.5	6.2	**33.2**	55.3	**30.6**	3.8	1.3	6.2	10.9	18.8	4.2
3	1	6.2	34.4	55.3	31.0	3.8	**0.7**	6.2	**9.7**	18.9	3.5
6	0.5	6.2	36.0	55.4	33.2	3.8	0.9	6.2	11.5	18.9	**2.9**
12	0.25	6.3	38.4	55.4	35.7	3.7	1.2	6.3	12.2	19.0	4.7
2	2	8.2	11.5	73.8	10.9	5.0	0.6	8.2	4.9	25.1	2.1
4	1	8.2	**11.2**	73.8	**10.0**	5.1	**0.2**	8.2	**3.2**	25.1	1.1
8	0.5	8.3	11.9	73.8	10.9	5.1	0.3	8.3	3.8	25.2	**1.0**
16	0.25	8.4	13.0	73.9	12.0	5.0	0.4	8.4	4.1	25.3	1.6

we obtain the following bound:

$$\min \sqrt{L_{\text{sub}}^{(i)}} \leq \min \left\| \boldsymbol{W}^{(i)} - [\alpha_i(\boldsymbol{Y}_i - 0.5 \cdot \mathbf{1}_Y)(\boldsymbol{Z}_i - 0.5 \cdot \mathbf{1}_Z)] \right\| \tag{S.5}$$

$$= \min \left\| \boldsymbol{W}^{(i)} - \boldsymbol{W}_{\text{svd}}^{(i)} + \boldsymbol{W}_{\text{svd}}^{(i)} - [\alpha_i(\boldsymbol{Y}_i - 0.5 \cdot \mathbf{1}_Y)(\boldsymbol{Z}_i - 0.5 \cdot \mathbf{1}_Z)] \right\| \tag{S.6}$$

$$\leq \min \left\| \boldsymbol{W}^{(i)} - \boldsymbol{W}_{\text{svd}}^{(i)} \right\| + \left\| \boldsymbol{W}_{\text{svd}}^{(i)} - [\alpha_i(\boldsymbol{Y}_i - 0.5 \cdot \mathbf{1}_Y)(\boldsymbol{Z}_i - 0.5 \cdot \mathbf{1}_Z)] \right\| \tag{S.7}$$

$$\leq \min \sqrt{\sum_{j=l+1}^{\min(m,n)} \sigma_j^2} + \left\| \boldsymbol{W}_{\text{svd}}^{(i)} - \alpha_i \cdot \operatorname{sgn}\left(\boldsymbol{U}_{\text{svd}}^{(i)}\right) \operatorname{sgn}\left(\boldsymbol{V}_{\text{svd}}^{(i)}\right) \right\| \tag{S.8}$$

$$= \sqrt{\sum_{j=l+1}^{\min(m,n)} \sigma_j^2} + \left\| \boldsymbol{W}_{\text{svd}}^{(i)} - \frac{\left\langle \boldsymbol{W}_{\text{svd}}^{(i)}, \operatorname{sgn}\left(\boldsymbol{U}_{\text{svd}}^{(i)}\right) \operatorname{sgn}\left(\boldsymbol{V}_{\text{svd}}^{(i)}\right) \right\rangle}{\left\| \operatorname{sgn}\left(\boldsymbol{U}_{\text{svd}}^{(i)}\right) \operatorname{sgn}\left(\boldsymbol{V}_{\text{svd}}^{(i)}\right) \right\|^2} \cdot \operatorname{sgn}\left(\boldsymbol{U}_{\text{svd}}^{(i)}\right) \operatorname{sgn}\left(\boldsymbol{V}_{\text{svd}}^{(i)}\right) \right\|, \tag{S.9}$$

where σ_j and $\boldsymbol{W}_{\text{svd}}^{(i)}$ denote the singular values and the reconstructed matrix after applying an l-rank approximation to the SVD of $\boldsymbol{W}^{(i)}$, respectively. Furthermore, $\boldsymbol{U}_{\text{svd}}^{(i)}$ and $\boldsymbol{V}_{\text{svd}}^{(i)}$ represent the left and right singular vectors obtained from the SVD of $\boldsymbol{W}^{(i)}$, where $\boldsymbol{U}_{\text{svd}}^{(i)}$ already incorporates the top l singular values.

This result indicates that the approximation error of the BQQ formulation is closely related to the magnitude of the discarded singular values. Consequently, BQQ achieves higher representational fidelity for matrices with rapidly decaying singular spectra, while its approximation quality degrades when the low-rank truncation leaves substantial residual energy. This theoretical property can also be observed in the trends shown in Fig. 3.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions of this work, including the proposed quantization framework, the associated optimization method, and its empirical performance. The claims are consistent with the theoretical and experimental results, and appropriately scoped to reflect the evaluated settings and assumptions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are discussed in the subsection "**Further Potential and Limitations**" in Section 6. We specifically address the suboptimality of the greedy optimization strategy, the lack of a theoretical upper bound on the approximation error, and the relatively high quantization time. Additionally, we consider possible directions for improving the method through broader adaptation and structural refinement, in line with the NeurIPS Code of Ethics on transparency and reproducibility.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: This work has derived the KL divergence between the mean-field approximation distribution and the canonical distribution in the PUBO cost function (Equation 4) to extend the previous AMFD algorithm from QUBO to PUBO, as shown in Appendix A.1.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided all necessary details for experimental reproducibility, including the optimization algorithms used and the full set of hyper-parameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is not currently publicly available. We plan to release it after publication for reproducing the main results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the experimental settings and details in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The paper reports results from a single run with a fixed random seed, following common practice in recent quantization studies. While error bars and statistical significance are not reported, the results were consistently observed across multiple settings and benchmarks. We leave more extensive statistical analysis as future work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments were run on local CPU/GPU machines. Appendix A.5 includes hardware details and measured runtimes for several experiments, providing a sufficient indication of the compute requirements for reproduction.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research presented in the paper fully conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: None of social impacts we feel must be specifically highlighted here.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No high-risk data or models are released in this work, so safeguards are not applicable.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: This work makes use of existing assets, including datasets, pretrained models, and quantization methods. All such assets are credited in the paper, and their licenses and terms of use are properly respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This work introduces new code, which is not released publicly at this time. However, relevant implementation details are clearly described in the paper. Therefore, the question regarding documentation alongside released assets is not applicable.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human subjects were involved in this work.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: No LLMs were used in the development of the proposed method, experiments, or analysis, and they had no influence on the contributions of this work.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.