Unifying Uniform and Binary-coding Quantization for Accurate Compression of Large Language Models

Seungcheol Park[1], **Jeongin Bae**[2], **Beomseok Kwon**[2], **Minjun Kim**[1], **Byeongwook Kim**[2], **Se Jung Kwon**[2], **U Kang**[1*], **Dongsoo Lee**[2]
Seoul National University[1] NAVER Cloud[2]
{ant6si, minjun.kim, ukang}@snu.ac.kr

Abstract

How can we quantize large language models while preserving accuracy? Quantization is essential for deploying large language models (LLMs) efficiently. Binary-coding quantization (BCQ) and uniform quantization (UQ) are promising quantization schemes that have strong expressiveness and optimizability, respectively. However, neither scheme leverages both advantages. In this paper, we propose **UniQuan**$_F$ (**Uni**fied **Quan**tization with **F**lexible Mapping), an accurate quantization method for LLMs. UniQuan$_F$ harnesses both strong expressiveness and optimizability by unifying the flexible mapping technique in UQ and BCQ's non-uniform quantization levels. We propose unified initialization, and local and periodic mapping techniques to optimize the parameters in UniQuan$_F$ precisely. After optimization, our unification theorem removes computational and memory overhead, allowing us to utilize the superior accuracy of UniQuan$_F$ without extra deployment costs induced by the unification. Experimental results demonstrate that UniQuan$_F$ outperforms existing UQ and BCQ methods, achieving up to 4.60% higher accuracy on GSM8K benchmark.

1 Introduction

How can we compress large language models without compromising accuracy? Reducing the size of large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Dubey et al., 2024) is crucial for deploying them in real-world applications since they require expensive computational and memory costs. Quantization algorithms (Xu et al., 2018; Kwon et al., 2022; Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2024) efficiently compress LLMs via bit-width reduction of weights by encoding them with a small set of values, namely, quantization levels. Each quantization scheme has its own quantization parameters that determine the quantization levels; accurately optimizing these parameters is important because it guarantees that the quantization levels are well aligned with the model's weight distribution.

Uniform quantization (UQ) (Lee et al., 2023; Lin et al., 2024; Shao et al., 2024) and binary-coding quantization (BCQ) (Xu et al., 2018; Kwon et al., 2022) are promising quantization schemes that ensure fast inference of quantized LLMs. While UQ divides its quantization levels uniformly, BCQ results in non-uniform quantization levels through the addition and subtraction of values assigned per each bit. Research on quantizing LLMs with UQ is actively ongoing, but no study has yet explored BCQ on LLMs as fast acceleration kernels (Park et al., 2024a; You et al., 2024) that support BCQ scheme have been released recently.

We compare the expressiveness and optimizability of quantization schemes for LLMs in Figure 1. UQ demonstrates strong optimizability, efficiently reducing quantization errors, by leveraging various advanced methods such as FlexRound (Lee et al., 2023) and OmniQuant (Shao et al., 2024). However, its expressiveness is constrained, failing to adapt its quantization levels to the weight distribution due to their uniform spacing. Conversely, BCQ offers strong expressiveness through non-uniform quantization levels, but its optimizability is limited due to the absence of precise quantization techniques. ALTERNATING (Xu et al., 2018) is the only BCQ method applicable to LLMs, but its major drawback is that it does not consider the input distribution. In summary, neither scheme fully achieves both strong expressiveness and optimizability.

In this paper, we propose **UniQuan**$_F$ (**Uni**fied **Quan**tization with **F**lexible Mapping), an accurate quantization method for LLMs. We find that the strong optimizability of UQ originates from its transformation process, and the strong expressiveness of BCQ originates from its generalized

*Corresponding author

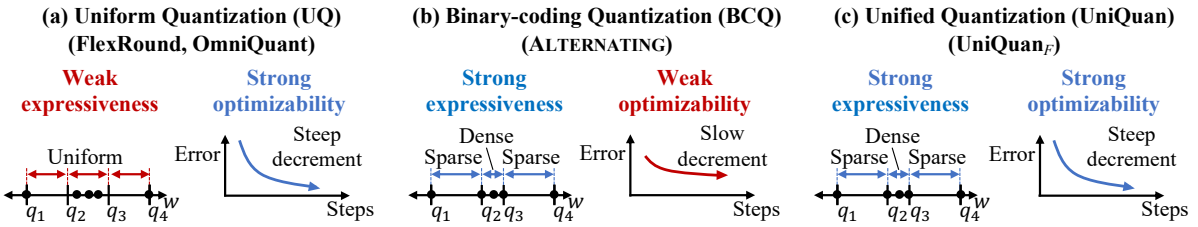


Figure 1: A comparison of (a) uniform, (b) binary-coding, and (c) unified quantization schemes. UQ has strong optimizability swiftly reducing errors during optimization, and BCQ has strong expressiveness adapting its non-linear quantization levels (q, bar) according to the distribution of weights (w, circle). UniQuan combines the advantages of both schemes by unifying UQ and BCQ.

mapping process that maps weights to non-uniform quantization levels (see Section 3.1). Building on this observation, we define **UniQuan** (**Uni**fied **Quan**tization) which unifies UQ and BCQ schemes by integrating UQ's transformation process into BCQ's mapping process, thereby harnessing both strong optimizability and expressiveness as shown in Figure 1(c). In UniQuan$_F$, we unify FlexRound and ALTERNATING, the best-performing UQ and BCQ methods, respectively, following the quantization process of UniQuan. We improve the accuracy of UniQuan$_F$ with two main ideas: 1) unified initialization for joint initialization of quantization parameters from FlexRound and ALTERNATING, and 2) local and periodic mapping to accelerate the slow mapping process of BCQ. We further remove the extra deployment cost caused from the unification by integrating the two-step inference process into a single step with unification theorem (see Theorem 1). As a result, UniQuan$_F$ exhibits the best performance without any additional memory and computational costs at deployment.

Our main contributions are as follows:

- **Algorithm.** We propose UniQuan$_F$, an accurate quantization method for LLMs. UniQuan$_F$ unifies the best-performing UQ and BCQ methods to leverage both of their advantages.
- **Analysis.** We analyze that UniQuan$_F$ exhibits both strong expressiveness and optimizability.
- **Experiments.** We show that UniQuan$_F$ outperforms UQ and BCQ methods on various benchmarks, showing up to 4.60% higher accuracy.

The source code for implementing UniQuan$_F$ is publicly available at `https://github.com/snudm-starlab/UniQuanF`.

The rest of this paper is organized as follows. Section 2 defines LLM quantization problem and provides preliminaries. We propose UniQuan$_F$ in Section 3. Section 4 presents the experimental results, followed by a discussion of related works

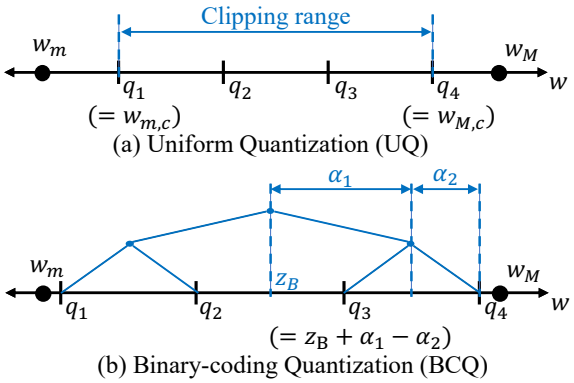


Figure 2: Illustrations of quantization levels (q) assigned for a weight group $\boldsymbol{w}$ under (a) UQ and (b) BCQ schemes, where w_m and w_M are the minimum and the maximum weights in $\boldsymbol{w}$, respectively. UQ has evenly-spaced quantization levels within a clipping range while BCQ has non-uniform quantization levels determined by its scale factors $\boldsymbol{\alpha}$ and a shifting factor z_B.

in Section 5. Finally, we conclude the paper with a summary of our findings.

2 Preliminary

We introduce the LLM quantization problem and describe preliminaries. We describe the frequently used terminologies in Appendix A.

2.1 LLM Quantization Problem

We have a pre-trained LLM F, a desired bit-width k, and a sample dataset $\mathbb{D}$. Our objective is to find an accurate k-bit quantized model $\widehat{F}$. The target model is a Transformer-based (Vaswani et al., 2017) LLM (Jiang et al., 2023; Dubey et al., 2024) with N blocks, where quantization is applied to the weight matrices in each block. We divide each weight matrix into multiple weight groups that share quantization parameters. Given a group $\boldsymbol{w} \in \mathbb{R}^g$ of g weight values and a desired bit-width k, a quantizer Q quantizes $\boldsymbol{w}$ into $\widehat{\boldsymbol{w}} = Q(\boldsymbol{w}, k; \Theta)$ where Θ is a set of quantization parameters. In this paper, we focus on two quantization schemes: uni-

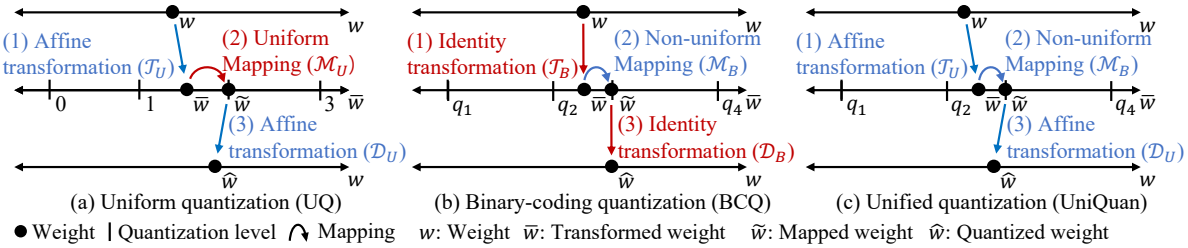


Figure 3: Quantization processes for a weight w in (a) UQ, (b) BCQ, and (c) UniQuan schemes. Blue-colored processes are parameterized functions which are the source of UQ's strong optimizability and BCQ's powerful expressiveness, respectively. UniQuan takes the advantages of both UQ and BCQ schemes by combining the parameterized functions in both schemes. See Section 3.1 for details.

form quantization (UQ) and binary-coding quantization (BCQ) which are supported by the state-of-the-art inference kernel (Park et al., 2024a). We directly compare their accuracy since they require the same memory and computational costs.

2.2 Uniform Quantization (UQ)

Uniform quantization (UQ) (Lee et al., 2023; Shao et al., 2024) is a quantization scheme that has uniformly spaced quantization levels. We explain UQ using Round-to-nearest (RTN) (Gupta et al., 2015), a representative method of UQ. Figure 2(a) illustrates the quantization levels in UQ that are evenly distributed in a clipping range $[w_{m,c}, w_{M,c}]$, where $w_{m,c}$ and $w_{M,c}$ are the minimum and maximum values in the clipping range, respectively. RTN begins with a grid search to find the proper $w_{m,c}$ and $w_{M,c}$ to precisely approximate the original weights. The quantization parameters $\Theta_R = \{\Delta, z_U\}$ of RTN are derived based on its clipping range; scale factor $\Delta = (w_{M,c} - w_{m,c})/(2^k - 1)$ and zero-point $z_U = \lfloor -w_{m,c}/\Delta \rceil$, where $\lfloor \cdot \rceil$ is the rounding function. Then, RTN quantizer Q_R quantizes $\boldsymbol{w} \in \mathbb{R}^g$ into $\widehat{\boldsymbol{w}} = Q_R(\boldsymbol{w}, k; \Theta_R)$ as follows:

$$\widetilde{\boldsymbol{w}} = \mathit{Clip}\left(\lfloor \boldsymbol{w}/\Delta + z_U \mathbf{1}_g \rceil, 0, 2^k - 1\right),$$
$$\widehat{\boldsymbol{w}} = \Delta(\widetilde{\boldsymbol{w}} - z_U \mathbf{1}_g),$$

where $\mathit{Clip}(\cdot, m, M)$ is an element-wise clipping function with a min-max range $[m, M]$, and $\widetilde{\boldsymbol{w}}$ is a vector of low-bit integers assigned for each weight. $\mathbf{1}_g$ is a vector of size g filled with ones.

2.3 Binary-coding Quantization (BCQ)

Binary-coding quantization (BCQ) is a non-uniform quantization scheme with adaptive quantization levels. Its quantization parameters $\Theta_B = \{\boldsymbol{\alpha}, z_B\}$ consists of a vector $\boldsymbol{\alpha} \in \mathbb{R}^k$ of scale factors and a shifting factor $z_B \in \mathbb{R}$. Figure 2(b) illustrates the quantization levels in BCQ that are determined through the summation and subtraction of scale factors $\boldsymbol{\alpha}$ after shifting with z_B. In this example, BCQ quantizer defines the set $\{z_B + \alpha_1 + \alpha_2, z_B + \alpha_1 - \alpha_2, z_B - \alpha_1 + \alpha_2, z_B - \alpha_1 - \alpha_2\}$ of four quantization levels forming a binary tree centered at z_B with widths α_1 and α_2.

We assign a binary code $\boldsymbol{c} \in \{-1, +1\}^k$ for each weight w, indicating whether the weight selects the positive or negative value for each scale factor in $\boldsymbol{\alpha}$. Then, BCQ quantizer Q_B quantizes a weight group $\boldsymbol{w} \in \mathbb{R}^g$ into $\widehat{\boldsymbol{w}} = Q_B(\boldsymbol{w}; \Theta_B)$ as follows:

$$\widehat{\boldsymbol{w}} = \boldsymbol{C}\boldsymbol{\alpha} + z_B \mathbf{1}_g,$$
$$\mathit{where}\ \ \boldsymbol{C} = \arg\min_{\boldsymbol{C}'} ||\boldsymbol{w} - (\boldsymbol{C}'\boldsymbol{\alpha} + z_B \mathbf{1}_g)||_2^2.$$

$\boldsymbol{C} \in \{-1, +1\}^{g \times k}$ is the binary code matrix for $\boldsymbol{w}$ where its ith row contains the binary code for the ith weight in $\boldsymbol{w}$. Existing works (Xu et al., 2018) first set z_B as 0, then search the scale factors $\boldsymbol{\alpha}$ and binary codes $\boldsymbol{C}$ through an alternating update process (see Algorithm 2 in Xu et al. (2018)).

The main advantage of BCQ is its strong expressiveness; any UQ is representable in the form of BCQ (see Appendix C in Park et al. (2024a)). However, BCQ methods are far less accurate in quantizing LLMs than UQ methods due to the lack of accurate BCQ-based quantization methods. In this paper, we propose UniQuan$_F$ which transfers UQ's accurate quantization methods to BCQ to fully exploit BCQ's strong expressiveness.

3 Proposed Method

We propose **UniQuan$_F$** (**Uni**fied **Quan**tization with **F**lexible Mapping), an accurate quantization method for LLMs. We first propose **UniQuan** (**Uni**fied **Quan**tization), a framework unifying the quantization processes of UQ and BCQ schemes to exploit their strong expressiveness and optimizability. Then, we present UniQuan$_F$-0, a naïve method that unifies FlexRound and ALTERNATING following UniQuan, but with several limitations. Finally,

Table 1: Comparison of quantization methods. UniQuan$_F$ incorporates the advantages of FlexRound and ALTERNATING by exploiting $\mathcal{T}_F$, $\mathcal{D}_R$, and $\mathcal{M}_B^*$, overcoming the challenges in UniQuan$_F$-0. $\circ$ is a function composition operator. See Sections 3.2 for details.

Method	Optimization		Deployment	
	Initialization	Quantization	Parameters	Inference
FlexRound	Grid search	$\mathcal{D}_R \circ \mathcal{M}_U \circ \mathcal{T}_F$	$\widetilde{\boldsymbol{w}}, \Theta_R$	$\mathcal{D}_R(\widetilde{\boldsymbol{w}}; \Theta_R)$
ALTERNATING	Alternating update	$\mathcal{M}_B$	$\boldsymbol{C}, \Theta_B$	$\mathcal{R}_B(\boldsymbol{C}; \Theta_B)$
UniQuan$_F$-0	-	$\mathcal{D}_R \circ \mathcal{M}_B \circ \mathcal{T}_F$	$\boldsymbol{C}, \Theta_B, \Theta_R$	$\mathcal{D}_R(\mathcal{R}_B(\boldsymbol{C}; \Theta_B); \Theta_R)$
UniQuan$_F$ (Proposed)	Unified initialization	$\mathcal{D}_R \circ \mathcal{M}_B^* \circ \mathcal{T}_F$	$\boldsymbol{C}, \Theta_B^*$	$\mathcal{R}_B(\boldsymbol{C}; \Theta_B^*)$

we propose UniQuan$_F$ which achieves high accuracy without requiring extra costs at deployment, by addressing the limitations in UniQuan$_F$-0.

3.1 Unification of UQ and BCQ

We define a general representation of the quantization processes in UQ and BCQ schemes to clarify the source of their superior capabilities. Given a quantizer Q with quantization parameters Θ, the general representation is defined as follows:

$$\widehat{\boldsymbol{w}} = Q(\boldsymbol{w}; \Theta) = \mathcal{D}(\mathcal{M}(\mathcal{T}(w; \Theta); \Theta); \Theta),$$

where $\mathcal{T}$ is a transformation process which transforms weights $\boldsymbol{w}$ to transformed weights $\bar{\boldsymbol{w}}$, $\mathcal{M}$ is a mapping process which maps transformed weights $\bar{\boldsymbol{w}}$ to the corresponding quantization levels, and $\mathcal{D}$ is a detransformation process which reverts mapped weights $\widetilde{\boldsymbol{w}}$ to the original weight space, yielding quantized weights $\widehat{\boldsymbol{w}}$. We illustrate the quantization processes in diverse quantization schemes in Figure 3 following this general representation.

Optimizability of UQ. As shown in Figure 3(a), we formulate the quantizer Q_U of UQ as follows:

$$Q_U(\boldsymbol{w}, k; \Theta_U) = \mathcal{D}_U(\mathcal{M}_U(\mathcal{T}_U(\boldsymbol{w}; \Theta_U), k); \Theta_U),$$

where $\mathcal{M}_U(\boldsymbol{w}, k) = Clip(\lfloor \boldsymbol{w} \rceil, 0, 2^k - 1)$ is a uniform mapping function that maps transformed weights $\widetilde{\boldsymbol{w}}$ to uniform quantization levels. $\mathcal{T}_U$ and $\mathcal{D}_U$ are affine transformation functions with quantization parameters Θ_U. There are two main methods belonging to UQ: RTN and FlexRound. In RTN, the quantizer Q_R is formulated as follows:

$$Q_R(\boldsymbol{w}, k; \Theta_R) = \mathcal{D}_R(\mathcal{M}_U(\mathcal{T}_R(\boldsymbol{w}; \Theta_R), k); \Theta_R),$$

where $\Theta_R = \{\Delta, z_U\}$, $\mathcal{T}_R(\boldsymbol{w}; \Theta_R) = \boldsymbol{w}/\Delta + z_U \mathbf{1}_g$ and $\mathcal{D}_R(\boldsymbol{w}; \Theta_R) = \Delta(\boldsymbol{w} - z_U \mathbf{1}_g)$. In FlexRound, the main idea is to allow each weight to explore diverse quantization levels to find the optimal one. This is achieved by replacing the transformation function $\mathcal{T}_R$ with $\mathcal{T}_F$ defined as follows:

$$\mathcal{T}_F(\boldsymbol{w}; \Theta_F) = \boldsymbol{w} \oslash (\Delta \cdot \boldsymbol{s} \cdot s_r) + z_U \mathbf{1}_g, \quad (1)$$

where $\Theta_F = \{\Delta, z_U, \boldsymbol{s}, s_r\}$ is a set of quantization parameters of FlexRound, and $\oslash$ denotes an element-wise division. $\boldsymbol{s} \in \mathbb{R}^g$ and $s_r \in \mathbb{R}$ are element-wise and row-wise scale factors to promote the exploration of weights, respectively.

In summary, each quantization method has its own Θ_U, $\mathcal{T}_U$, and $\mathcal{D}_U$, and they are the origin of strong optimizability. We formulate the quantizers of various UQ methods in Appendix E.6.

Expressiveness of BCQ. As shown in Figure 3(b), we formulate the quantizer Q_B of BCQ as follows:

$$Q_B(\boldsymbol{w}; \Theta_B) = \mathcal{D}_B(\mathcal{M}_B(\mathcal{T}_B(\boldsymbol{w}); \Theta_B)),$$

where $\mathcal{T}_B(\boldsymbol{w}) = \mathcal{D}_B(\boldsymbol{w}) = \boldsymbol{w}$ are identity functions. $\mathcal{M}_B(\boldsymbol{w}; \Theta_B) = Q_B(\boldsymbol{w}; \Theta_B)$ is a non-uniform mapping function that maps transformed weights $\widetilde{\boldsymbol{w}}$ to the nearest non-uniform quantization levels. $\mathcal{M}_B$ enables the mapping between weights and non-uniform quantization levels, enabling strong expressiveness. ALTERNATING (Xu et al., 2018) is the only quantization method in BCQ scheme applicable for LLMs with the same quantizer Q_B defined above.

UniQuan (Unified Quantization). We propose UniQuan, a framework unifying the quantization processes of UQ and BCQ schemes. UniQuan formulates the unified quantizer Q_I with quantization parameters $\Theta_I = \{\Theta_B, \Theta_U\}$ as follows (see Figure 3(c)):

$$Q_I(\boldsymbol{w}; \Theta_I) = \mathcal{D}_U(\mathcal{M}_B(\mathcal{T}_U(\boldsymbol{w}; \Theta_U); \Theta_B); \Theta_U).$$

Q_I has strong expressiveness by mapping weights to the non-uniform quantization levels using $\mathcal{M}_B$. Also, Q_I has strong optimizability by incorporating useful methods in UQ by replacing Θ_U, $\mathcal{T}_U$, and

$\mathcal{D}_U$ with those of the method to incorporate. We elaborate on incorporating FlexRound (Lee et al., 2023), the best-performing UQ method, into BCQ in Sections 3.2 and 3.3. We cover the incorporation of AWQ (Lin et al., 2024), OmniQuant (Shao et al., 2024), and GPTQ (Frantar et al., 2023) into BCQ in Section E.6.

3.2 UniQuan$_F$-0

We introduce a basic method UniQuan$_F$-0 which naively unifies the best-performing UQ and BCQ methods, FlexRound (Lee et al., 2023) and ALTERNATING (Xu et al., 2018), respectively. The quantizer Q_{I_F} of UniQuan$_F$-0 is defined as follows:

$$Q_{I_F}(\boldsymbol{w}; \Theta_{I_F}) = \mathcal{D}_R(\mathcal{M}_B(\mathcal{T}_F(\boldsymbol{w}; \Theta_F); \Theta_B); \Theta_R). \quad (2)$$

Note that FlexRound begins its optimization process by initializing its quantization parameters with a grid search process to find a proper clipping range. FlexRound enhances its optimizability by utilizing $\mathcal{T}_F$ in Equation 1 and optimizes its quantization parameters Θ_F using stochastic gradient descent (SGD). After optimization, FlexRound stores low-bit weights $\widetilde{\boldsymbol{w}} = \mathcal{M}_U(\mathcal{T}_F(\boldsymbol{w}; \Theta_F), k)$ and quantization parameters $\Theta_R = \{\Delta, z_U\}$, and approximates the original weights $\boldsymbol{w}$ as $\widehat{\boldsymbol{w}} = \mathcal{D}_R(\widetilde{\boldsymbol{w}}; \Theta_R)$ in the detransformation step; $\boldsymbol{s}$ and s_r in Θ_F are used only for transformation. On the other hand, ALTERNATING initializes its quantization parameters in $\Theta_B = \{\boldsymbol{\alpha}, z_B\}$ with alternating update (Xu et al., 2018). Then, it stores the binary code matrix $\boldsymbol{C}$ and quantization parameters Θ_B. It approximates weights $\boldsymbol{w}$ as $\widehat{\boldsymbol{w}} = \mathcal{R}_{\mathcal{B}}(\boldsymbol{C}; \Theta_B) = \boldsymbol{C}\boldsymbol{\alpha} + z_B \mathbf{1}_g$ for inference.

UniQuan$_F$-0 inherits the strong optimizability from FlexRound's $\mathcal{T}_F$ and $\mathcal{D}_R$, and the strong expressiveness from ALTERNATING's $\mathcal{M}_B$. However, UniQuan$_F$-0 has three points of improvements as follows:

C1. Joint initialization. FlexRound and ALTERNATING are based on different initialization methods which seem incompatible. How can we jointly initialize the quantization parameters considering their dependency?

C2. Slow mapping process. The quantization levels of UniQuan$_F$-0 are changed during optimization process where its quantization parameters are updated. Therefore, it is crucial to update the mappings between weights and quantization levels during optimization using $\mathcal{M}_B$. However, existing method (Xu et al., 2018) exploits a binary search tree (BST) to implement $\mathcal{M}_B$ which finds the mappings through a GPU-unfriendly sequential process, and it requires intractable time for updating mappings during optimization. How can we expedite the mapping process?

C3. Expensive deployment cost. UniQuan$_F$-0 suffers from memory and computational overheads at deployment, since UniQuan$_F$-0 require executing both inference procedures $\mathcal{D}_R$ and $\mathcal{R}_B$ (see Table 1).How can we decrease the inference cost?

Algorithm 1 UniQuan$_F$ for a block f

Input: A Transformer block f, f's input $\boldsymbol{X}$, a set Φ of weight groups in f, the input $\widehat{\boldsymbol{X}}$ of a quantized f obtained by preceding quantized blocks, number E of epochs, and number $|\mathbb{D}|$ of data points

Output: Optimized quantization parameters Θ_B^* and a binary-code matrix $\boldsymbol{C}$ for each group

Initialize Θ_{I_F} for each weight group ▷ **(I1)** Unified Initialization
for epoch e in 1, 2, ..., E **do**
  for s in 1, 2, ..., $|\mathbb{D}|$ **do**
    Quantize Φ into $\widehat{\Phi}$ using Equation 2 ▷ **(I2)** Local and periodic mapping
    $\mathcal{L} \leftarrow ||f(\boldsymbol{X}_s; \Phi) - f(\widehat{\boldsymbol{X}}_s; \widehat{\Phi})||_F^2$
    Perform backpropagation using $\mathcal{L}$
    Update Θ_{I_F} for each group
  end for
end for
Convert each Θ_{I_F} into $\Theta_B^* = \{\boldsymbol{\alpha}^*, z_B^*\}$ ▷ **(I3)** Unification Theorem
Find $\boldsymbol{C}$ using Equation 4 for each group

3.3 UniQuan$_F$

We propose **UniQuan$_F$** (**Uni**fied **Quan**tization with **F**lexible Mapping), which improves UniQuan$_F$-0 with the following main ideas.

I1. Unified initialization. We unify the initialization algorithms in FlexRound and ALTERNATING to jointly initialize the quantization parameters, considering their dependency.

I2. Local and periodic mapping. We efficiently update mappings by exploiting the locality and the temporal sparsity of changes in mappings.

I3. Unification theorem. We unify the inference process of UniQuan$_F$-0 into a single BCQ's inference process, removing additional memory and computational costs at deployment.

UniQuan$_F$ quantizes an LLM through a block-wise optimization strategy, quantizing sequentially from the bottom-most Transformer block upwards. Algorithm 1 details the process of quantizing a Transformer block f with a set Φ of weight groups, to obtain a binary-code $\boldsymbol{C}$ and optimized quantization parameters Θ_B^* for each group. We provide inputs $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ of unquantized and quantized blocks, respectively. We iterate the optimization process for E epochs using $|\mathbb{D}|$ data points.

First, for each weight group, we initialize the quantization parameters $\Theta_{I_F} = \Theta_B \cup \Theta_F$ using the unified initialization technique (line 1) which we describe later in this subsection. Subsequently, we iteratively optimize the quantization parameters to minimize the distance between the outputs of the quantized and unquantized blocks (lines 2-9). In each optimization step, we quantize the weight groups in Φ using UniQuan$_F$'s quantizer $Q_{I_F}^*$ with its quantization parameters Θ_{I_F} as follows:

$$\begin{aligned} &Q_{I_F}^*(\boldsymbol{w}; \Theta_{I_F}) \\ &\quad = \mathcal{D}_R(\mathcal{M}_B^*(\mathcal{T}_F(\boldsymbol{w}; \Theta_F); \Theta_B); \Theta_R), \end{aligned} \tag{3}$$

where $\mathcal{M}_B^*$ represents the local and periodic mapping technique, which we describe later in this subsection, to accelerate mappings to non-uniform quantization levels. $\mathcal{T}_F$ is the transformation function of FlexRound in Equation 1. We quantize each weight group in parallel, and the quantized weight groups are stored in $\widehat{\Phi}$ (line 4). We measure the reconstruction loss $\mathcal{L}$ which represents the distance between the outputs of the original and quantized blocks, and then optimize the quantization parameters Θ_{I_F} via stochastic gradient descent (lines 5-7). After the iterative optimization process, we have the optimized quantization parameters $\Theta_{I_F} = \Theta_B \cup \Theta_F$ which generate the outputs closely resembling those of the original block. We leverage the unification theorem (Theorem 1) to merge the optimized quantization parameters into a single set of BCQ's quantization parameters $\Theta_B^* = \{\boldsymbol{\alpha}^*, z_B^*\}$ (line 10). Finally, we obtain a binary-code matrix $\boldsymbol{C}$ using Θ_B^* for each weight group as follows:

$$\boldsymbol{C} = \arg\min_{\boldsymbol{C}'} ||\boldsymbol{w} - (\boldsymbol{C}'\boldsymbol{\alpha}^* + z_B^* \mathbf{1}_g)||_2^2. \tag{4}$$

Then, the quantization process for a block is completed (line 11). We obtain an entirely quantized model by sequentially applying Algorithm 1 from the bottom to the top blocks.

Table 2: The percentage of index changes per optimization step and throughout all 2,560 steps. Minimal changes in each step add up to substantial shifts overall.

Step size	**Index change**			
	0	**1**	**2**	**>2**
Single step	99.97%	0.01%	0.00%	0.01%
All steps (2,560)	89.30%	8.37%	1.36%	0.97%

Unified Initialization. How can we jointly initialize the quantization parameters of FlexRound and ALTERNATING? FlexRound initializes Δ and z_U through a grid search process, and ALTERNATING utilizes an alternating update process to find $\boldsymbol{\alpha}$, fixing z_B as 0. We propose a unified initialization process which initializes the quantization parameters $\Theta_{I_F} = \{\Delta, z_U, \boldsymbol{s}, s_r, \boldsymbol{\alpha}, z_B\}$ by unifying both techniques, considering their dependency.

In unified initialization, we perform alternating updates to find $\boldsymbol{\alpha}$ and z_B in each iteration of a grid search process for Δ and z_U. As a result, we find $\boldsymbol{\alpha}$ and z_B according to the Δ and z_U during the grid search process. As in FlexRound, we initialize $\boldsymbol{s}$ and s_r as $\mathbf{1}_g$ and 1, respectively. We detail the unified initialization algorithm in Appendix E.1.

Local and Periodic Mapping. How can the mappings between weights and quantization levels be swiftly updated during optimization? Table 2 shows that while over 8% of weights change indices of the mapped quantization levels during the full optimization, 99.97% remain stable at each step, and most changes involve adjacent levels. Hence, we propose a local and periodic mapping which evaluates only neighboring levels and remaps periodically. We set a hyperparameter p as a remapping period, and in every p steps, we find the closest quantization level among the previous level and its two neighbors. We detail the process of local and periodic mapping in Appendix E.4.

Unification Theorem. How can we eliminate the memory and computational overheads at deployment induced by the unification? The deployment cost of UniQuan is expensive since the unified quantization process in Equation 3 requires two-step inference procedures with $\mathcal{D}_R$ and $\mathcal{R}_B$. We propose a unification theorem that integrates the two-step inference process of UniQuan$_F$-0 into a single BCQ inference process as follows:

Theorem 1 (Unification Theorem)**.** *Given a reconstruction function $\mathcal{R}_B(\boldsymbol{C}; \Theta_B)$ and a detransformation function $\mathcal{D}_R(\widetilde{\boldsymbol{w}}; \Theta_R)$ where $\widetilde{\boldsymbol{w}} =$*

Table 3: Comparison of the average accuracy of 0-shot and 5-shot on MMLU, and the perplexity on WikiText2 (Wiki) benchmarks. Higher accuracies and lower perplexities indicate better performance. Bold and underlined texts indicate the best and second-best performance, respectively. UniQuan$_F$ shows the best performance in most cases.

# Bits	Scheme†	Method	Mistral 7B		Llama-3 8B		Llama-3 70B	
			MMLU Avg. (↑)	Wiki (↓)	MMLU Avg. (↑)	Wiki (↓)	MMLU Avg. (↑)	Wiki (↓)
Full precision			61.40	5.25	62.77	6.14	77.52	2.86
4	UQ	RTN	55.20	6.04	54.67	7.80	67.74	3.85
		OmniQuant	56.28	5.67	56.02	<u>7.37</u>	76.62	3.34
		FlexRound	<u>58.60</u>	<u>5.52</u>	<u>58.59</u>	8.22	**77.24**	<u>3.31</u>
	BCQ	ALTERNATING	24.65	9.8e4	22.95	1.3e5	44.07	11.06
		UniQuan$_F$	**59.22**	**5.51**	**61.43**	**7.01**	<u>76.97</u>	**3.19**
3	UQ	RTN	28.59	36.07	24.35	120.50	40.61	24.42
		OmniQuant	32.82	9.24	26.34	42.06	71.13	<u>5.23</u>
		FlexRound	<u>53.45</u>	<u>6.41</u>	<u>50.22</u>	<u>10.11</u>	<u>72.44</u>	5.81
	BCQ	ALTERNATING	24.19	1.1e4	25.51	8.4e4	24.06	1.5e3
		UniQuan$_F$	**53.68**	**6.20**	**53.46**	**8.75**	**74.79**	**4.24**

† UQ and BCQ methods have the same costs when using BCQ kernels (Park et al., 2024a)

$\mathcal{R}_B(\boldsymbol{C};\Theta_B)$, there is a set Θ_B^* of unified quantization parameters such that $\mathcal{R}_B(\boldsymbol{C};\Theta_B^*) = \mathcal{D}_R(\mathcal{R}_B(\boldsymbol{C};\Theta_B);\Theta_R)$ for any $\boldsymbol{C}$, Θ_B, and Θ_R.

Proof. See Appendix E.5. □

As a result, UniQuan$_F$ exploits the UQ's optimizability without any extra memory and computational costs at deployment.

Discussion. UniQuan$_F$ is interpreted as a quantization framework that augments the BCQ's quantization process with the linear transformation process of UQ. By defining the transformation function in UniQuan$_F$ as a linear function, several key advantages emerge. First, it enables the seamless integration of well-established UQ methods (Lee et al., 2023; Lin et al., 2024; Shao et al., 2024) into the BCQ framework, thereby reducing the time and effort required to develop new methods. Second, through the unification theorem, the inference process is consolidated into a single step, allowing the system to benefit from the improved accuracy of the unified approach without incurring additional memory or computational costs at deployment. One might consider introducing non-linear transformation functions to further enhance the expressiveness of the unified quantization process. However, this approach presents notable drawbacks since the aforementioned two key advantages do not hold for non-linear transformation functions; we cannot exploit existing UQ methods and unification theorem. Therefore, we incorporate linear transformation processes into the BCQ's quantization process, rather than non-linear transformation processes, to enhance efficiencies in method development and deployment.

4 Experiments

We perform experiments to answer the following questions. Additional analyses on UniQuan$_F$ are discussed in Appendix F.

- **Q1. General knowledge evaluation.** How accurately does UniQuan$_F$ quantize LLMs on general knowledge benchmarks?
- **Q2. Task-specific knowledge evaluation.** How accurately does UniQuan$_F$ quantize LLMs on task-specific knowledge benchmarks?
- **Q3. Ablation study.** Do all ideas in UniQuan$_F$ improve the accuracy of quantized LLMs?
- **Q4. Case study.** Does UniQuan$_F$ effectively utilize BCQ's expressiveness and UQ's optimizability as we intended?

4.1 Experimental Setup

We briefly introduce the experimental setup. Refer to Appendix C for further details.

Setup. We evaluate the quantized performance of Llama-3 8B, Llama-3 70B (Dubey et al., 2024), and Mistral 7B (Jiang et al., 2023) models on MMLU (Hendrycks et al., 2021) and WikiText2 (Merity et al., 2017) benchmarks. For task-specific experiment, we quantize Llama-3 8B Instruct (Dubey et al., 2024) model and evaluate on

Table 4: Accuracies (%) of quantized Llama-3 Instruct 8B models on GSM8K benchmark. UniQuan$_F$ outperforms the competitors in all cases.

Scheme	Method	4bit	3bit
Full precision		76.12	
UQ	RTN	49.43	0.08
	OmniQuant	59.16	2.78
	FlexRound	<u>70.66</u>	<u>54.13</u>
BCQ	ALTERNATING	0.00	0.00
	UniQuan$_F$	**72.38**	**58.73**

GSM8K (Cobbe et al., 2021). We sample 128 token sequences of length 2048 from C4 (Raffel et al., 2020) and GSM8K (Cobbe et al., 2021) for general and task-specific benchmarks, respectively. All experiments are done with a single A100 GPU.

Baselines. We compare UniQuan$_F$ with the four UQ and BCQ methods: RTN (Gupta et al., 2015), OmniQuant (Shao et al., 2024), FlexRound (Lee et al., 2023), and ALTERNATING (Xu et al., 2018).

Hyperparameters. We set each row of weight matrices as a single weight group except for Llama-3 70B where we divide each row into small weight groups of size 128 to prevent severe accuracy loss. Note that each row has at least 1024 weights. Hyperparameters of UniQuan$_F$ is in Appendix C.1. We provide a thorough and detailed analysis of UniQuan$_F$ by altering its hyperparameters to guide hyperparameter search processes in Section F.

4.2 General Knowledge Evaluation

We evaluate the amount of general knowledge retained in quantized models. Table 3 summarizes the average accuracies of 5-shot and 0-shot MMLU, and perplexity on WikiText2 (Wiki). UniQuan$_F$ achieves the highest performance in almost all cases, outperforming the second-best method by up to 3.24% in average MMLU accuracy. In Llama-3 70B, a small group size of 128 makes a small number of weights sharing their quantization parameters, reducing the difficulty of quantization; competitors show comparative performance to UniQuan$_F$ in both 3- and 4-bit cases.

4.3 Task-specific Knowledge Evaluation

Table 4 summarizes the accuracies of a quantized Llama-3 8B Instruct model evaluated on GSM8K benchmark, which focuses exclusively on mathematical problems. Note that GSM8K is a difficult task in which models should generate the exact answer, while general knowledge benchmarks require only selecting correct answers among multiple choices, or evaluating given texts. UniQuan$_F$ achieves the highest performance, outperforming the second-best competitor by 4.60% and 1.72% in the 3-bit and 4-bit experiments, respectively. Notably, UniQuan$_F$ significantly improves the performance of FlexRound by utilizing ALTERNATING even when ALTERNATING exhibits zero accuracy. This demonstrates that UniQuan$_F$ effectively unifies UQ and BCQ methods, leading to achieving previously unattainable performance.

Table 5: Ablation study results on MMLU benchmark. All ideas contribute to improving accuracies.

Method	4bit	3bit
UniQuan$_F$	**61.43**	**53.46**
- remapping	56.89	47.83
- unified init. (Θ_F)	24.70	22.87
- unified init. (Θ_B)	<u>60.84</u>	<u>52.14</u>

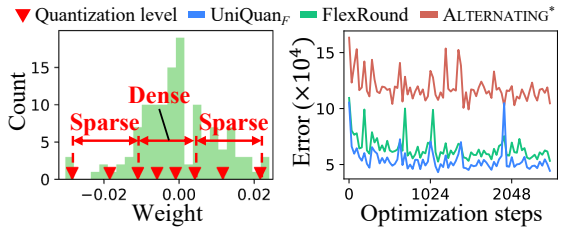
4.4 Ablation Study

Table 5 summarizes the result of an ablation study using a Llama-3 8B model to show the effectiveness of our main ideas. "-remapping" refers to the case where we do not apply local and periodic mapping, thus mappings are not updated during optimization. "-unified init. (Θ_F)" and "-unified init. (Θ_B)" denote the cases when we do not use unified initialization. In "-unified init. (Θ_F)", we do not perform a grid search process to initialize quantization parameters in Θ_F, i.e. we initialize Δ, z_U, s, and s_r in Θ_F as 1, 0, $\mathbf{1}_g$, and 1, respectively. In "-unified init. (Θ_B)", we do not perform an alternating update process to initialize quantization parameters in Θ_B, i.e. we initialize $\boldsymbol{\alpha}$ and z_B in Θ_B to have uniformly spaced quantization levels in the space transformed by $\mathcal{T}_F$. As a result, UniQuan$_F$ outperforms all of its variants, showing the effectiveness of our main ideas. Specifically, precisely initializing Θ_F is crucial to exploit the strong optimizability of Θ_F.

4.5 Case Study

We conduct a case study on a Llama-3 8B model quantized with UniQuan$_F$ to verify whether the intended mechanisms function properly.

Expressiveness. Figure 4(a) visualizes the distribution of weights (green bars) within a weight group and the assigned quantization levels (red triangles) following UniQuan$_F$. As shown in the



(a) Quantization levels (b) Error decrements

Figure 4: (a) Quantization levels learned by UniQuan$_F$ and (b) changes of reconstruction error during optimization. UniQuan$_F$ assigns quantization levels that align closely with the weight distribution, and effectively reduces the reconstruction error during optimization.

figure, UniQuan$_F$ assigns dense quantization levels near zero, where most weights are concentrated, and sparse levels in regions with fewer weights. This demonstrates that UniQuan$_F$ effectively learns non-linear quantization levels aligned to the weight distribution, harnessing the strong expressiveness of BCQ scheme.

Optimizability. Figure 4(b) shows the changes of reconstruction error during optimization for a block when using FlexRound (green), ALTERNATING* (brown), and UniQuan$_F$ (blue); ALTERNATING* is a modification of ALTERNATING that optimizes $\boldsymbol{\alpha}$ using a block-wise output reconstruction process in UniQuan$_F$ devised for comparing optimizability. Compared to the baselines, UniQuan$_F$ achieves the least error at the end with a steep decrease at the early stage of optimization. This demonstrates that UniQuan$_F$ effectively minimizes the reconstruction error, harnessing the strong optimizability of UQ scheme.

In summary, UniQuan$_F$ successfully unifies UQ and BCQ schemes, harnessing their strong expressiveness and optimizability, as we intended. We further verify that UniQuan$_F$ effectively leverages flexible mapping, the main optimization technique of FlexRoud, in Appendix F.6.

5 Related Work

Quantization. Quantization (Lin et al., 2024; Ashkboos et al., 2024; Piao et al., 2022; Liu et al., 2024b) has gained great attention since it effectively speeds up LLM inference by minimizing the cost of memory operations. The majority of quantization approaches rely on uniform quantization (UQ) (Shao et al., 2024; Lee et al., 2023) due to its hardware-friendly nature, benefiting from pre-existing acceleration kernels. Some works (Ashkboos et al., 2024; Liu et al., 2024b) even quantize activation by mitigating outliers through weight rotation, which is orthogonal to our approach. Other approaches, such as vector (Kim et al., 2024; Tseng et al., 2024; Liu et al., 2024a) and additive quantization (Egiazarian et al., 2024) suffer from slow inference since they lack of acceleration kernels. Binary-coding quantization (BCQ) (Xu et al., 2018; Kwon et al., 2022) also has been underexplored due to its missing kernels even with broader expressiveness over UQ (see Section 2.3). However, recent studies (Park et al., 2024a; You et al., 2024) have proposed kernels leveraging look-up tables, enabling BCQ to achieve the same speed as UQ. Taking advantage of them, UniQuan unlocks the superior expressiveness of BCQ by unifying it with the UQ scheme.

Other compression methods. Pruning (Men et al., 2024; Frantar and Alistarh, 2023; Ma et al., 2023; Park et al., 2025, 2024b; Zhong et al., 2024) and knowledge distillation (Gu et al., 2024; Wang et al., 2020; Jeon et al., 2023; Jang et al., 2023; Lee et al., 2021; Cho and Kang, 2022; Yoo et al., 2019; Kim et al., 2021) are compatible techniques that improve the accuracy of compressed models when used with quantization (Park et al., 2024c). Pruning improves the accuracy of quantized models by allowing us to utilize more bit-widths for the necessary weights under memory constraints through explicitly removing unnecessary weights. Knowledge distillation boosts the accuracy of quantized models by transferring the useful knowledge of uncompressed models to quantized models during quantization.

6 Conclusion

We propose UniQuan$_F$, an accurate quantization method for LLMs. We unify the best-performing uniform quantization (UQ) and binary-coding quantization (BCQ) methods to leverage their strong optimizability and expressiveness concurrently. We propose local and periodic mapping, unified initialization, and unification theorem to improve the accuracies of quantized models without introducing additional memory and computational costs at deployment. As a result, UniQuan$_F$ achieves the best performance, demonstrating the effectiveness of the unification. Incorporating diverse UQ methods into BCQ, and integrating UQ methods with other non-uniform quantization schemes beyond BCQ are our promising future works.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No.RS-2020-II200894, Flexible and Efficient Model Compression Method for Various Applications and Environments], [No.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], and [No.RS-2021-II212068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)]. This work was supported by Youlchon Foundation. The Institute of Engineering Research at Seoul National University provided research facilities for this work. The ICT at Seoul National University provides research facilities for this study. U Kang is the corresponding author.

Limitations

In this paper, we define UniQuan which is a quantization scheme that unifies uniform quantization (UQ) and binary-coding quantization (BCQ) schemes. We verify the effectiveness of UniQuan with only UniQuan$_F$ which unifies the best-performing UQ and BCQ methods. We cover the unification of extended quantization methods in Appendix E.6 in theoretical aspects. The experimental validation of diverse pairs of quantization methods in extended quantization schemes is one of our promising future works.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *CoRR*, abs/2404.00456.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ikhyun Cho and U Kang. 2022. Pea-kd: Parameter-efficient and accurate knowledge distillation on bert. *PLOS ONE*, 17(2).

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jun-Gi Jang, Chun Quan, Hyun Dong Lee, and U Kang. 2023. Falcon: lightweight and accurate convolution based on depthwise separable convolution. *Knowl. Inf. Syst.*, 65(5):2225–2249.

Hyojin Jeon, Seungcheol Park, Jin-Gee Kim, and U. Kang. 2023. Pet: Parameter-efficient knowledge distillation on transformer. *PLOS ONE*, 18(7).

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Junghun Kim, Jinhong Jung, and U. Kang. 2021. Compressing deep graph convolution network with multi-staged knowledge distillation. *PLOS ONE*, 16.

Minjun Kim, Jaehyeon Choi, Jongkeun Lee, Wonjin Cho, and U Kang. 2025a. Zero-shot quantization: A comprehensive survey. In *IJCAI*.

Minjun Kim, Jongjin Kim, and U Kang. 2025b. Synq: Accurate zero-shot quantization by synthesis-aware fine-tuning. In *ICLR*.

Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2024. Squeezellm: Dense-and-sparse quantization. In *Forty-first International Conference on Machine Learning*.

Se Jung Kwon, Jeonghoon Kim, Jeongin Bae, Kang Min Yoo, Jin-Hwa Kim, Baeseong Park, Byeongwook Kim, Jung-Woo Ha, Nako Sung, and Dongsoo Lee. 2022. Alphatuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3288–3305. Association for Computational Linguistics.

Hyun Dong Lee, Seongmin Lee, and U. Kang. 2021. Auber: Automated bert regularization. *PLOS ONE*, 16(6).

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. 2023. Flexround: Learnable rounding based on element-wise division for post-training quantization. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 18913–18939. PMLR.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. 2024a. VPTQ: extreme low-bit vector post-training quantization for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8181–8196. Association for Computational Linguistics.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024b. Spinquant: LLM quantization with learned rotations. *CoRR*, abs/2405.16406.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2024a. LUT-GEMM: quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Seungcheol Park, Hojun Choi, and U Kang. 2024b. Accurate retraining-free pruning for pretrained encoder-based language models. In *ICLR*.

Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. 2024c. A comprehensive survey of compression algorithms for language models. *arXiv preprint arXiv:2401.15347*.

Seungcheol Park, Sojin Lee, Jongjin Kim, Jinsik Lee, Hyunjik Jo, and U Kang. 2025. Accurate sublayer pruning for large language models by exploiting latency and tunability information. In *IJCAI*.

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

Tairen Piao, Ikhyun Cho, and U Kang. 2022. Sensimix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for bert compression. *PloS one*, 17(4):e0265621.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. 2018. Alternating multi-bit quantization for recurrent neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. 2019. Knowledge extraction with no observable data. In *NeurIPS*.

Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, and Yingyan Celine Lin. 2024. Shiftaddllm: Accelerating pretrained llms via post-training multiplication-less reparameterization. *arXiv preprint arXiv:2405.04532*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2024. Blockpruner: Fine-grained pruning for large language models. *arXiv*.

A Terminology

We summarize the definitions of terminologies frequently used in this paper to promote clarity.

A.1 Units in LLMs

We summarize the definitions of the units in Large Language Models (LLMs) from a weight to a model. Figure 5 illustrates an example of a Transformer-based LLM with N blocks.

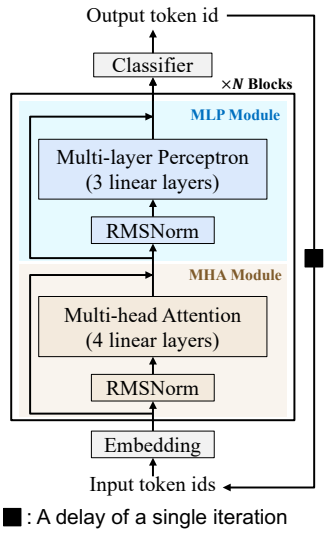


Figure 5: An illustration of a Transformer architecture with L blocks.

- **Weight:** the smallest unit, representing an individual numerical weight value. The quantization bit width represents the number of bits to represent each weight.
- **Weight group:** a collection of weights grouped by a specified group size, all of which share the same quantization parameters. It is easy to quantize models when we have small group sizes since we have plenty of quantization parameters.
- **Weight matrix:** a two-dimensional matrix composed of weights, containing multiple weight groups.
- **Layer:** a component that performs affine transformations with a weight matrix and a bias vector.
- **Module:** a collection of layers that performs a specific functionality. In Transformers, modules include Multi-Head Attention (MHA) and Multi-Layer Perceptron (MLP).
- **Block:** a fundamental unit of a Transformer, consisting of one MHA module and one MLP module. FlexRound (Lee et al., 2023) and UniQuan$_F$ sequentially quantize each block from the bottom to top to reduce the cost of quantization.
- **Model:** a complete language model consisting of multiple blocks. An LLM refers to a model.

Table 6: Symbols and their definitions.

Symbol	Definition
w	Weight
$\boldsymbol{w}$	Group of weights
$\bar{\boldsymbol{w}}$	Group of transformed weights
$\widetilde{\boldsymbol{w}}$	Group of mapped weights
$\widehat{\boldsymbol{w}}$	Group of quantized weights
Φ	Set of weight groups
g	The size of a weight group
k	Quantization bit-width
f	Transformer block
$\lvert\mathbb{D}\rvert$	Number of data points
$\mathbf{1}_g$	Vector of size g filled with ones
$\lfloor\cdot\rceil$	Rounding function
$Clip(\cdot, m, M)$	Clipping function with range $[m, M]$
$\oslash$	Element-wise division
Δ	Scale factor of UQ
z_U	Zero-point of UQ
$\boldsymbol{C}$	Binary-code matrix of BCQ
$\boldsymbol{\alpha}$	Scale factors of BCQ
z_B	Shifting factor of BCQ
$\boldsymbol{s}$	Element-wise scale factor of FlexRound
s_r	Row-wise scale factor of FlexRound
G	Grid search iterations
T	Alternating update iterations
p	Remapping period

A.2 Error Types

In this paper, we mention two types of errors: quantization error and output reconstruction error. Quantization error represents the error before and after quantization at the weight level without considering the model's input. If we quantize a weight group $\boldsymbol{w}$ into $\widehat{\boldsymbol{w}}$, then the quantization error is $||\boldsymbol{w} - \widehat{\boldsymbol{w}}||_2^2$. On the other hand, output reconstruction error measures how the quantized model's outputs are close to those before quantization. We measure the reconstruction error at block level. Assume that we have a block f with parameters Φ, quantized into $\widehat{f}$ with parameters $\widehat{\Phi}$. The reconstruction error is $||f(\boldsymbol{X}; \Phi) - \widehat{f}(\widehat{\boldsymbol{X}}; \widehat{\Phi})||_F^2$, where $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ are the inputs of the block f and the quantized block $\widehat{f}$, respectively.

We first initialize the quantization parameters to minimize quantization errors at the weight level. Then, we optimize the quantization parameters to minimize the output reconstruction errors at the block level to take account of the input distribution.

Table 7: A summary of terminologies regarding different quantization methods. $\Rightarrow$ represents the application of the unification theorem (see Theorem 1). UniQuan's quantization parameters Θ_U, transformation function $\mathcal{T}_U$ and detransformation function $\mathcal{D}_U$ are determined according to the UQ method unified into the BCQ scheme.

Method	**Quantizer**	**Quantization Parameters**	**Transformation Function**	**Mapping Function**	**Detransformation Function**	**Reconstruction Function**
General form	Q	Θ	$\mathcal{T}(\boldsymbol{w};\Theta)$	$\mathcal{M}(\boldsymbol{w};\Theta)$	$\mathcal{D}(\boldsymbol{w};\Theta)$	$\mathcal{R}_B(\boldsymbol{C};\Theta)$
RTN	Q_R	$\Theta_R=\{\Delta,z_U\}$	$\mathcal{T}_R(\boldsymbol{w};\Theta_R)$	$\mathcal{M}_U(\boldsymbol{w},k)$	$\mathcal{D}_R(\boldsymbol{w};\Theta_R)$	-
FlexRound	Q_F	$\Theta_F=\{\Delta,z_U,\boldsymbol{s},s_r\}$	$\mathcal{T}_F(\boldsymbol{w};\Theta_F)$			
ALTERNATING	Q_B	$\Theta_B=\{\boldsymbol{\alpha},z_B\}$	$\mathcal{T}_B(\boldsymbol{w})$	$\mathcal{M}_B(\boldsymbol{w};\Theta_B)$	$\mathcal{D}_B(\boldsymbol{w})$	$\mathcal{R}_B(\boldsymbol{C};\Theta_B)$
UniQuan	Q_I	$\Theta_I=\Theta_U\cup\Theta_B\Rightarrow\Theta_B^*$	$\mathcal{T}_U(\boldsymbol{w};\Theta_U)$	$\mathcal{M}_B^*(\boldsymbol{w};\Theta_B)$	$\mathcal{D}_U(\boldsymbol{w};\Theta_U)$	$\mathcal{R}_B(\boldsymbol{C};\Theta_B^*)$
UniQuan$_F$	Q_{I_F}	$\Theta_{I_F}=\Theta_F\cup\Theta_B\Rightarrow\Theta_B^*$	$\mathcal{T}_F(\boldsymbol{w};\Theta_F)$		$\mathcal{D}_F(\boldsymbol{w};\Theta_F)$	

A.3 Symbols and Definitions

We summarize the frequently used symbols and their definitions in Tables 6 and 7.

B Inference Speed of BCQ Kernels

In this paper, we propose UniQuan which unifies UQ (Lee et al., 2023; Lin et al., 2024; Shao et al., 2024; Kim et al., 2025a,b) and BCQ (Xu et al., 2018; Kwon et al., 2022) schemes, and the resulting models are represented in a BCQ scheme. The effectiveness of UniQuan depends on the inference speed of BCQ kernels (Park et al., 2024a; You et al., 2024). Thus, we summarize the inference speed of LUT-GEMM (Park et al., 2024a), the state-of-the-art BCQ kernel, for completeness.

Table 8: Latency comparison of the first FFN layer on OPT-175B model with various precision and kernels on A100-80GB-GPU.

Kernel	Schemes	# Bits	Latency (ms)
cuBLAS	-	16	0.7256
GPTQ	UQ	3	0.3599
AWQ	UQ	4	0.3238
LUT-GEMM	UQ, BCQ	4	0.2688
LUT-GEMM	UQ, BCQ	3	0.2250

Table 8 summarizes the main results of Table 1 in LUT-GEMM (Park et al., 2024a). In this table, latency represents the time for inferencing the first FFN layer in OPT-175B (Zhang et al., 2022) with various precision, and LUT-GEMM shows faster inference speeds than GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) kernels, which support only a UQ scheme.

Table 9 summarizes the main results of Table 6 in LUT-GEMM (Park et al., 2024a), reporting the end-to-end latency per token for OPT family models. k and g represent the bit width and group size, respectively. As summarized in the table, LUT-GEMM provides end-to-end inference speedup with diverse bit-widths and group sizes.

Table 9: Comparison of end-to-end latency per token for OPT-30B models on a A100-80GB-GPU.

Model	Kernel-k-g	Latency (ms)
OPT -30B	cuBLAS-16-N/A	40.5
	LUT-GEMM-4-32	18.5
	LUT-GEMM-4-64	17.8
	LUT-GEMM-3-32	16.7
	LUT-GEMM-3-64	15.7

In summary, LUT-GEMM provides faster inference speed for quantized models than existing UQ kernels and speeds up end-to-end inference. Therefore, UniQuan$_F$, which is supported by LUT-GEMM, is an essential method for quantizing LLMs. Note that we need to convert quantized models using a UQ scheme into a BCQ scheme to use LUT-GEMM which provides a faster inference speed than existing UQ kernels as shown in Table 8. Therefore, quantized models in both schemes require the same memory and computational costs when they have the same bit width.

C Implementation Details

We use PyTorch (Paszke et al., 2019) and Hugging Face (Wolf et al., 2019) libraries for implementation. We use the pretrained weights of Llama-3 (Dubey et al., 2024) and Mistral (Jiang et al., 2023) models from the Hugging Face library. We discuss the implementation details of UniQuan$_F$ and competitors to reproduce the performance reported in this paper.

C.1 Implementation Details of UniQuan$_F$

Hyperparameter settings. Our objective is to demonstrate that the outstanding performance of

Table 10: Hyperparameter settings of UniQuan$_F$

Hyperparameter	Setting
Learning rate for Θ_F	0.005
Learning rate for Θ_B	0.0005
Grid search iterations (G)	1, 30
Alternating update iterations (T)	15
Remapping Period (p)	2
Epochs	20
Batch size	1
Clipping strategy	Fixed-minimum

UniQuan$_F$ is achieved without expensive hyperparameter tuning although UniQuan$_F$ employs a block-wise output reconstruction process that requires many hyperparameters. To this end, we fix all hyperparameters except for the grid size G for unified initialization and utilize only two combinations of hyperparameters across all cases. We outline these combinations in Table 10.

Gradient filtering. We utilize a straight-through estimator (STE) (Bengio et al., 2013) to update UniQuan$_F$'s quantization parameters since its mapping function $\mathcal{M}_B^*$ is not differentiable. We filter the gradient of the weights that have large mapping errors $|\bar{w} - \widetilde{w}|$ to stabilize the optimization process of UniQuan$_F$, where $\bar{w}$ and $\widetilde{w}$ represent transformed and mapped weights, respectively. STE hypothesizes that the gradient of a transformed weight $\bar{w}$ and mapped weight $\widetilde{w}$ have the same gradients. However, if the difference between transformed and mapped weight is significant, the hypothesis does not hold which degrades the accuracy of quantized models. Therefore, we set the hyperparameter τ as a gradient filtering threshold, and zero out the gradients of weights whose mapping error is larger than τ.

We mimic the quantization process in the UQ scheme to determine τ. In UQ, weights within the clipping range $[w_{m,c}, w_{M,c}]$ are transformed to the values in the range $[0, 2^{k-1}]$, while weights outside the clipping range, such as w_m and w_M, are transformed into values outside the range $[0, 2^{k-1}]$. These out-of-range weights cause significant mapping errors. In the transformed space, the interval between quantization levels is 1, and each level has a range of 0.5 on either side. Thus, it is reasonable to allow a margin of 0.5 even for the smallest and largest quantization levels 0 and 2^{k-1}.

For UniQuan$_F$, every quantization level has a potential risk of significant mapping errors, and the value of 0.5 is replaced with the smallest scale factor $min(\boldsymbol{\alpha})$ which is the smallest width in the binary tree constructed by BCQ's quantization levels (see Figure 2 (b)). Therefore, we set the gradient filtering threshold τ as $min(\boldsymbol{\alpha})$.

C.2 Implementation Details of Competitors

RTN (Gupta et al., 2015). We search the clipping range with 100 iterations, which is larger than the number of iterations used in unified initialization. We execute the same source code for clipping range search in UniQuan$_F$.

ALTERNATING (Xu et al., 2018). We implement ALTERNATING based on the original paper (Xu et al., 2018). We use an alternating update with 15 iterations which is the same as the number of iterations used in unified initialization. We use the same source code for the alternating update in UniQuan$_F$.

FlexRound (Lee et al., 2023). We implement the FlexRound following the original paper (Lee et al., 2023).

OmniQuant (Shao et al., 2024). We refer to the official implementation[2] of OmniQuant and report the results following the best hyperparameter settings reported in the paper.

D Evaluation Protocol

We report the average performance using random seeds 0, 1, and 2 except for Llama-3 70B which uses only 0 because of its long quantization time. We sample 128 token sequences of length 2048 from C4 (Raffel et al., 2020) and GSM8K (Cobbe et al., 2021) for general and task-specific knowledge evaluations, respectively. We detail the evaluation protocols for general and task-specific knowledge evaluation as follows.

General knowledge evaluation. We use MMLU (Hendrycks et al., 2021) and WikiText2 (Merity et al., 2017) benchmarks for general knowledge evaluation. MMLU consists of multiple-choice problems across 57 subjects. We evaluate the 0-shot and 5-shot accuracies on MMLU to evaluate the amount of general knowledge in the quantized models. 0-shot and 5-shot refer to settings with 0 and 5 examples provided in the prompt, respectively. We follow the evaluation protocol in the official code repository[1] for the MMLU benchmark. WikiText2 consists of tokens within a set of

[1] https://github.com/hendrycks/test

verified articles from Wikipedia. We report the perplexity of quantized models on WikiText2 benchmark to evaluate the language modeling capabilities of quantized models. We follow the evaluation protocol used in OmniQuant (Shao et al., 2024) according to its official implementation[2].

Task-specific knowledge evaluation. We use GSM8K (Cobbe et al., 2021) benchmark for task-specific knowledge evaluation. GSM8K consists of high-quality grade school math problems that require multi-step reasoning, and we use GSM8K to evaluate the amount of Mathematical knowledge in the quantized models. We implement our evaluation code using the language model evaluation harness (Gao et al., 2023) package[3].

We summarize the properties of benchmarks in Table 11.

Table 11: Properties of benchmarks

Benchmark	Subject	Instance	Metric
MMLU	General	14,042	Accuracy
WikiText2	General	141	Perplexity
GSM8K	Math	1,319	Accuracy

E Theoretical Details

We explain the theoretical details on clipping strategy, general alternating update, local and periodic mapping, unification theorem, and extensibility of UniQuan.

E.1 Algorithm of Unified Initialization

Algorithm 2 outlines the overall process of unified initialization which initializes the quantization parameters in the set $\Theta_{I_F} = \{\Delta, z_U, \boldsymbol{s}, s_r, \boldsymbol{\alpha}, z_B\}$ for each weight group. Following FlexRound (Lee et al., 2023), we initialize $\boldsymbol{s}$ and s_r as $\mathbf{1}_g$ and 1, respectively. We initialize the center z_B of BCQ's quantization levels to $(2^k-1)/2$ since $\mathcal{T}_F$ transforms weights $\boldsymbol{w}$ into the range $[0, 2^k-1]$ (line 2). Then, we conduct a grid search by adjusting the scale ratio γ to find the optimal quantization parameters (lines 3-13). We explore diverse values for the center of BCQ's quantization levels in the original weight space, by adjusting candidate scale factor Δ' and candidate zero-point z'_U according to γ; the weight transformed into z_B changes as the adjusted

[2] https://github.com/OpenGVLab/OmniQuant/tree/main

[3] https://github.com/EleutherAI/lm-evaluation-harness/tree/main/lm_eval/tasks/gsm8k

Algorithm 2 Unified Initialization

Input: Weights $\boldsymbol{w}$, grid search iterations G, alternating update iterations T, and a bit-width k
Output: Initialized quantization parameters Θ_{I_F}

1: $w_m, w_M \leftarrow min(\boldsymbol{w}), max(\boldsymbol{w})$
2: $z_B, \boldsymbol{s}, s_r, e \leftarrow (2^k-1)/2, \mathbf{1}_g, 1,$ MAX_NUM
3: **for** γ in $1/G, 2/G, ..., 1$ **do**
4: $\quad\Delta', z'_U \leftarrow$ adjust-clipping(w_m,w_M,γ,k) ▷ Appendix E.2
5: $\quad\bar{\boldsymbol{w}} \leftarrow \mathcal{T}_F(\boldsymbol{w}; \Delta', z'_U, \boldsymbol{s}, s_r)$ ▷ Equation 1
6: $\quad\boldsymbol{\alpha}', z'_B \leftarrow$general-alternating($\bar{\boldsymbol{w}}$, z_B, G, T) ▷ Algorithm 3 in Appendix E.3
7: $\quad\Theta'_{I_F} \leftarrow \{\Delta', z'_U, \boldsymbol{s}, s_r, \boldsymbol{\alpha}', z'_B\}$
8: $\quad\widehat{\boldsymbol{w}} \leftarrow Q^*_{I_F}(\boldsymbol{w}; \Theta'_{I_F})$ ▷ Equation 2
9: $\quad e' \leftarrow ||\boldsymbol{w} - \widehat{\boldsymbol{w}}||_2^2$ ▷ Quantization error
10: $\quad$**if** $e' < e$ **then**
11: $\quad\quad\Delta, z_U, \boldsymbol{\alpha}, z_B, e \leftarrow \Delta', z'_U, \boldsymbol{\alpha}', z'_B, e'$
12: $\quad$**end if**
13: **end for**
14: $\Theta_{I_F} \leftarrow \{\Delta, z_U, \boldsymbol{s}, s_r, \boldsymbol{\alpha}, z_B\}$

Δ' and z'_U modify the transformation function $\mathcal{T}_F$. There are Fixed-minimum, Fixed-maximum, and Balanced strategies for adjusting Δ' and z'_U. The selection of a clipping strategy is a hyperparameter and we detail those strategies in Appendix E.2 (line 4). Then, we transform $\boldsymbol{w}$ into $\bar{\boldsymbol{w}}$ using Equation 1 (line 5). We find α' and z'_B through a general alternating update in Algorithm 3 which is the improved version of alternating update (Xu et al., 2018) to find z_B (line 6). After obtaining the candidate quantization parameters in Θ'_{I_F}, we quantize $\boldsymbol{w}$ into $\widehat{\boldsymbol{w}}$ and evaluate the quantization error e' (lines 7-9). We update the quantization parameters using the candidate ones only if the evaluated error e' is lower than the previous minimum error e (lines 10-12). After the grid search process over G iterations, we obtain the initialized quantization parameters (line 14).

E.2 Details of Clipping Strategies

In unified initialization, we find the optimal quantization parameters by adjusting the clipping range $[w_{m,c},\ w_{M,c}]$ from altering a hyperparameter γ. The length of the clipping range is adjusted as $\gamma(w_M - w_m)$ by setting scale factor $\Delta' = \gamma(w_M - w_m)/(2^k-1)$, and the clipping range is shifted according to the definition of the zero-point z'_U. There are three strategies to define z'_U as follows:

- **Fixed-minimum.** w_m is fixed at $w_{m,c}$ so that the w_m is always included in the clipping range. In this case, we find $z'_U = -w_m/\Delta'$ by solving

Algorithm 3 General alternating update

Input: A weight group $\boldsymbol{w}$ of size g. Numbers G and T of iterations for grid search and alternating update, respectively

Output: Updated $\boldsymbol{\alpha}$ and z_B

1: Greedily initialize $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ ▷ Equation 4 in Xu et al. (2018)
2: **for** t in 1, 2, ..., T **do**
3: $\quad\boldsymbol{\alpha} \leftarrow (\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T(\boldsymbol{w} - z_B\mathbf{1}_g)$
4: $\quad\boldsymbol{C} \leftarrow \arg\min_{\boldsymbol{C}'} ||\boldsymbol{w} - (\boldsymbol{C}'\boldsymbol{\alpha} + z_B\mathbf{1}_g)||_2^2$
5: $\quad$**if** $G = 1$ **then** ▷ Update z_B only if not grid search
6: $\quad\quad z_B \leftarrow \sum_{i=1}^{g}(w_i - \boldsymbol{C}_{i,:}\boldsymbol{\alpha})/g$
7: $\quad$**end if**
8: **end for**

$\mathcal{T}_F(w_m) = 0$.

- **Fixed-maximum.** w_M is fixed at $w_{M,c}$ so that the w_M is always included in the clipping range. In this case, we find $z'_U = 2^k - 1 - w_M/\Delta'$ by solving $\mathcal{T}_F(w_M) = 2^k - 1$.
- **Balanced.** $w_{m,c}$ and $w_{M,c}$ are adjusted as γw_m and γw_M, respectively, so that the minimum and maximum clipped values are adjusted in balance. In this case, we find $z'_U = -\gamma w_m/\Delta'$ by solving $\mathcal{T}_F(\gamma w_m) = 0$.

The center of BCQ's quantization levels explores more diverse values during grid search when we use Fixed-maximum or Fixed-minimum strategies than Balanced strategy; the clipping range is asymmetrically adjusted in Fixed-maximum and Fixed-minimum strategies.

E.3 Algorithm of General Alternating Update

Algorithm 3 details the general alternating update process employed in line 6 of Algorithm 2 which is an improved version of alternating update in Xu et al. (2018) to find proper z_B. We initialize $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ using a greedy initialization strategy (Equation (4) in Xu et al. (2018)), where we determine $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ for each bit-width one by one (line 1). Then, we iteratively minimize the quantization error $e = ||\boldsymbol{w} - (\boldsymbol{C}'\boldsymbol{\alpha} + z_B\mathbf{1}_g)||_2^2$ for T iterations to find the optimal quantization parameters (lines 2-8). In each iteration, we optimize for $\boldsymbol{\alpha}$, $\boldsymbol{C}$, and z_B sequentially while keeping the others fixed. For $\boldsymbol{\alpha}$ and z_B, we find the values where the derivatives $\frac{\partial e}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial e}{\partial z_B}$ equal to zero, respectively, where $\boldsymbol{C}_{i,:}$ represents the ith row of $\boldsymbol{C}$ (lines 3 and 6). For $\boldsymbol{C}$, we find the optimal binary codes for each weight by comparing errors obtained using all possible binary codes (line 4). After completing these iterations, we obtain the optimized values for $\boldsymbol{\alpha}$ and z_B. We do not update z_B when G is greater than 1 since diverse values for z_B are explored during the iterative grid search process (line 5).

Algorithm 4 Local and periodic mapping ($\mathcal{M}_B^*$)

Input: A transformed weight group $\bar{\boldsymbol{w}}$ of size g, iteration step s, a remapping period p, BCQ's quantization parameter $\Theta_B = \{\boldsymbol{\alpha}, z_B\}$, and mapping indices $\boldsymbol{d}$ and mapped weights $\widetilde{\boldsymbol{w}}$ in the previous step

Output: Updated mapping indices $\boldsymbol{d}$ and mapped weights $\widetilde{\boldsymbol{w}}$

1: **if** $s \equiv 0 \pmod p$ **then** ▷ Periodic mapping
2: $\quad\boldsymbol{q} \leftarrow$ compute-quantization-levels($\boldsymbol{\alpha}$, z_B)
3: $\quad$**for** i in $\{1, 2, ..., g\}$ **do**
4: $\quad\quad l, r \leftarrow max(d_i - 1, 1), min(d_i + 1, 2^k)$
5: $\quad\quad\mathcal{N} \leftarrow \{l, d_i, r\}$ ▷ Neighbor quantization levels' indices
6: $\quad\quad d_i \leftarrow \arg\min_{d' \in \mathcal{N}} |\bar{w}_i - q_{d'}|$ ▷ Local mapping
7: $\quad\quad\widetilde{w}_i \leftarrow q_{d_i}$
8: $\quad$**end for**
9: **end if**

E.4 Algorithm of Local and Periodic Mapping

Algorithm 4 describes the overall process of local and periodic mapping which swiftly updates the mapping between weights and quantization levels during optimization. In the beginning, we initialize the mapping by calculating distances to all quantization levels. Then, we begin local and periodic mapping using the resulting mapping $\boldsymbol{d}$ and mapped weights $\widetilde{\boldsymbol{w}}$ where $\boldsymbol{d}$ contains the indices of quantization levels that the weights are mapped. We periodically update the mappings between weights and quantization levels in every p optimization step (line 1). We compute quantization levels using $\boldsymbol{\alpha}$ and z_B, and store them in $\boldsymbol{q} \in \mathbb{R}^{2^k}$ (line 2). Then, we find the closest quantization level among its previously mapped level and its neighboring levels (lines 4-6). We also update the mapped weight value to use them for the successive steps before the next update (line 7). Local and periodic mapping is efficient, updating $1/p$ times less often than non-periodic algorithms and using only 3 among 2^k quantization levels for computing the distance between weights.

E.5 Proof of Theorem 1

Proof. By the definitions of $\mathcal{R}_B(\boldsymbol{C};\Theta_B)$ and $\mathcal{D}_R(\widetilde{\boldsymbol{w}};\Theta_R)$, we reduce $\mathcal{D}_R(R_B(\boldsymbol{C};\Theta_B);\Theta_R)$ as follows.

$$\begin{aligned}\mathcal{D}_R(\mathcal{R}_B(\boldsymbol{C};\Theta_B);\Theta_R)&\\ &= \Delta((\boldsymbol{C}\boldsymbol{\alpha} + z_B\mathbf{1}_g) - z_U\mathbf{1}_g)\\ &= \boldsymbol{C}(\Delta\boldsymbol{\alpha}) + \Delta(z_B - z_U)\mathbf{1}_g.\end{aligned}$$

Thus, $\boldsymbol{\alpha}^* = \Delta\boldsymbol{\alpha}$, $z_B^* = \Delta(z_B - z_U)$, and $R_B(\boldsymbol{C};\Theta_B^*) = \boldsymbol{C}\boldsymbol{\alpha}^* + z_B^*\mathbf{1}_g$ where $\Theta_B^* = \{\boldsymbol{\alpha}^*, z_B^*\}$. □

E.6 Extensibility of UniQuan

In the main text, we employ UniQuan$_F$ which unifies FlexRound (Lee et al., 2023) and ALTERNATING, the best-performing UQ and BCQ methods, respectively. We cover the unification between other quantization methods to explain the extensibility of UniQuan. The main idea of UniQuan is to utilize the unified quantization process with quantization parameters $\Theta_I = \Theta_U \cup \Theta_B$ as follows:

$$Q_I(\boldsymbol{w};\Theta_I) = \mathcal{D}_U(\mathcal{M}_B(\mathcal{T}_U(\boldsymbol{w};\Theta_U);\Theta_B);\Theta_U).$$

Considering the lack of accurate BCQ methods, we focus on replacing $\mathcal{T}_U$ and $\mathcal{D}_U$ with method-specific ones to unify UQ methods into the BCQ scheme.

AWQ (Lin et al., 2024) reduces output reconstruction error by scaling weights based on their importance before quantization. This is accomplished by using $\mathcal{T}_A$ and $\mathcal{D}_A$ defined as follows:

$$\begin{aligned}\mathcal{T}_A(\boldsymbol{w};\Theta_A) &= (\boldsymbol{w}\odot\boldsymbol{s}_A)/\Delta_A + z_A\mathbf{1}_g,\\ \mathcal{D}_A(\widetilde{\boldsymbol{w}};\Theta_A) &= \Delta_A(\widetilde{\boldsymbol{w}} - z_O\mathbf{1}_g)\oslash\boldsymbol{s}_A,\end{aligned}$$

where $\boldsymbol{s}_A \in \mathbb{R}^g$ is a column-wise scale factor to reflect the importance of each column of the weight matrix where the weight group $\boldsymbol{w}$ is located. $\Delta_A \in \mathbb{R}$ and $z_A \in \mathbb{R}$ represent the new scale factor and zero-point after applying $\boldsymbol{s}_A$, respectively. $\Theta_A = \{\Delta_A, z_A, \boldsymbol{s}_A\}$ is a set of quantization parameters for AWQ. $\odot$ and $\oslash$ represent element-wise multiplication and division, respectively.

OmniQuant (Shao et al., 2024) parameterizes the clipping range with parameters β and γ to find the best one through optimization. This is accomplished by using $\mathcal{T}_O$ and $\mathcal{D}_O$ as follows:

$$\begin{aligned}\mathcal{T}_O(\boldsymbol{w};\Theta_O) &= \boldsymbol{w}/\Delta_O + z_O\mathbf{1}_g,\\ \mathcal{D}_O(\widetilde{\boldsymbol{w}};\Theta_O) &= \Delta_O(\widetilde{\boldsymbol{w}} - z_O\mathbf{1}_g),\end{aligned}$$

where $\Delta_O = (\gamma w_M - \beta w_m)/(2^k - 1)$ and $z_O = -\lfloor \beta w_m/\Delta_O \rceil$ are the parameterized scale factor and zero-point, respectively. w_m and w_M are the minimum and maximum weights in a weight group $\boldsymbol{w}$. $\Theta_O = \{\beta, \gamma\}$ is the set of quantization parameters of OmniQuant.

GPTQ (Frantar et al., 2023) is widely used in a UQ scheme, but it is a general error compensation technique which covers even pruning (Frantar and Alistarh, 2023). Thus, GPTQ is able to be unified into a BCQ scheme by applying GPTQ's error compensation strategy to BCQ methods.

In summary, it is able to unify diverse UQ methods into a BCQ scheme following UniQuan's framework in theory. Extending the coverage of unification to other quantization schemes beyond UQ and BCQ, and validating with experiments is one of our promising future works.

F Additional Analyses

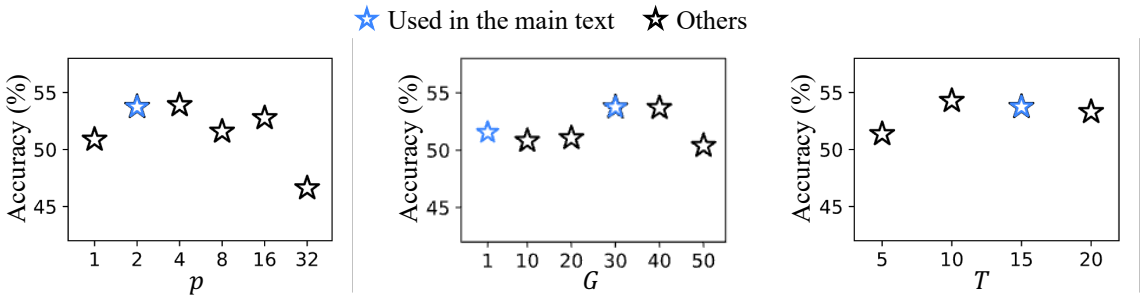
We analyze the memory usage, quantization time, hyperparameter sensitivity, and the effect of sample dataset size on UniQuan$_F$. We also validate that UniQuan$_F$ effectively leverages the optimization technique in FlexRound.

F.1 Memory Usage of UniQuan$_F$

We analyze the memory usage of quantized models using UniQuan$_F$. As summarized in Table 1, UniQuan$_F$ requires the same amount of memory at inference time as existing BCQ methods (Xu et al., 2018; Kwon et al., 2022) since we convert the inference process of the unified quantization process into a single BCQ's inference process using the unification theorem. When we quantize a weight group $\boldsymbol{w} \in \mathbb{R}^g$ of size g into k bits, we need to store a binary code matrix $\boldsymbol{C} \in \{-1, +1\}^{g\times k}$, BCQ's scale factors $\boldsymbol{\alpha} \in \mathbb{R}^k$, and BCQ's shifting factor $z_B \in \mathbb{R}$. Each binary code requires 1 bit and each real number requires 16 bits for saving, thus we require $gk + 16(k+1)$ bits in total. The memory overhead due to the quantization parameters is $16(k+1)/g$ bits per weight, and if we assume channel-wise quantization where g is larger than 4000 for LLMs (Jiang et al., 2023; Dubey et al., 2024), the memory overhead is negligible.

F.2 Quantization Time of UniQuan$_F$

We compare the running time for quantizing Llama-3 8B into 3 bits and the accuracy of the quantized model, using UniQuan$_F$ and FlexRound to evaluate the efficiency of UniQuan$_F$. We compare them



(a) Remapping period (p) (b) Grid search iterations (G) (c) Alternating update iterations (T)

Figure 6: Change of average accuracy on 0-shot and 5-shot MMLU benchmarks with regard to the change of remapping period p, grid search iterations G, and alternating update iterations T. Blue stars represent the hyperparameters used in the main text and black stars represent the others.

across different epoch settings since the quantization time varies depending on the number of epochs. We do not include the time for unified initialization in UniQuan$_F$ since it is performed as a preprocessing before quantization; the initialized quantization parameters are reused in different hyperparameter settings when the alternating update iteration T and the grid search iteration G are not changed.

Table 12: Comparison of average accuracies (%) on 0-shot and 5-shot MMLU benchmarks, and quantization time (s) when quantizing Llama-3 8B into 3bit using UniQuan$_F$ and FlexRound, respectively. Bold and underlined texts represent the best and second-best results in each method, respectively.

Epochs	UniQuan$_F$		FlexRound	
	Accuracy	Time	Accuracy	Time
5	47.97	5,089	46.71	4,239
10	51.04	10,022	47.97	8,350
15	52.08	15,905	51.08	12,405
20	53.46	20,403	**51.19**	16,868
25	**54.23**	25,884	48.19	21,970
30	<u>53.86</u>	28,521	<u>51.09</u>	25,139

Table 12 shows that UniQuan$_F$ requires about 20% longer time for quantization than FlexRound, but it achieves higher accuracy than FlexRound across all epoch settings. Especially, UniQuan$_F$ requires a shorter time of 15,905 seconds to outperform the highest accuracy of FlexRound which requires 16,868 seconds, demonstrating its efficiency.

The running time for the unified initialization process depends on G and T. When $G = 30$ and $T = 15$, as used in this experiment, it takes approximately 6,700 seconds and UniQuan$_F$ takes longer than FlexRound if we include this. However, considering that the quantization time only needs to be performed once for deployment, the initialization results are reusable, and FlexRound cannot reach the high performance of UniQuan$_F$ even though we invest more time.

F.3 Sensitivity Analysis on p, G, and T

We report the change in the performance of quantized models according to the variation of UniQuan$_F$'s hyperparameters G, T, and p to promote reproducibility. We report the average accuracy of 0-shot and 5-shot MMLU benchmarks of the 3-bit quantized Llama-3 8B models.

Remapping period (p). Figure 6(a) shows the change in the quantized models' accuracy with regard to the change of remapping period p. We use $p \in \{1, 2, 4, 8, 16, 32\}$ where a higher p represents the sparse update of mapping between weights and quantization levels. As a result, we find that the quantized model achieves the highest accuracy when $p \in \{2, 4\}$, outperforming the case of $p = 1$ where we update the mapping in every step. This result indicates that periodic mapping not only improves the efficiency of UniQuan$_F$ but also improves the accuracy of the quantized models. We recommend using $p = \{2, 4\}$ and we use only $p = 2$ in this paper since they exhibit similar accuracies.

Grid search iterations (G). Figure 6(b) shows the change in the quantized models' accuracy with regard to the change of grid search iterations G. We use $G \in \{1, 10, 20, 30, 40, 50\}$ where a higher G represents the exhaustive search for unified initialization. We also include $G = 1$ which indicates the case that we find the center of BCQ's quantization levels using the general alternating update in Algorithm 3, unlike the other cases that use grid search. As a result, we find that $G \in \{1, 30, 40\}$ shows the highest accuracy and we recommend using $G = \{1, 30\}$ to efficiently explore two different

Table 13: The average accuracies on 0-shot and 5-shot MMLU benchmarks of 3-bit Llama-3 8B models with various clipping strategies.

Strategy	Accuracy
Fixed-minimum	53.46
Fixed-maximum	54.34
Balanced	53.21

Table 14: The average accuracies on 0-shot and 5-shot MMLU benchmarks of 3-bit Llama-3 8B models on various sizes of sample datasets.

Sample Size	32	64	128	256
Accuracy	47.72	50.71	53.46	53.86

strategies for searching the BCQ's center.

Alternating update iterations (T). Figure 6(c) shows the change in the quantized models' accuracy with regard to the change of alternating update iterations G. We use $T \in \{5, 10, 15, 20\}$ where a higher T represents the more iterations for alternating update. As a result, we find that UniQuan$_F$ achieves high accuracy when T is equal to or higher than 10. Thus, we recommend using $T = \{10, 15\}$ which shows the highest accuracy.

F.4 Sensitivity on the Clipping Strategies

We compare the performance of three clipping strategies in unified initialization. Table 13 summarizes the comparison of the performance of Llama-3 8B models quantized using the three clipping strategies.

Fixed-maximum and Fixed-minimum strategies outperform the Balanced strategy. This is because the center of BCQ's quantization level explores more diverse values in Fixed minimum and Fixed maximum strategies than Balanced strategy, resulting in better quantization levels. Therefore, we recommend evaluating Fixed minimum and Fixed maximum strategies and using the better one according to the experimental settings.

F.5 Sensitivity on the Size of Sample Dataset

To illustrate the effect of sample dataset size on the performance of UniQuan$_F$, we quantize the Llama-3 8B model to 3 bits using sample datasets of varying sizes. We evaluate the performance of quantized models on 0-shot and 5-shot MMLU benchmarks. We use sample datasets ranging in size from 32 to 256, where 128 is the size used in the main text. We summarize the result in Table 14.

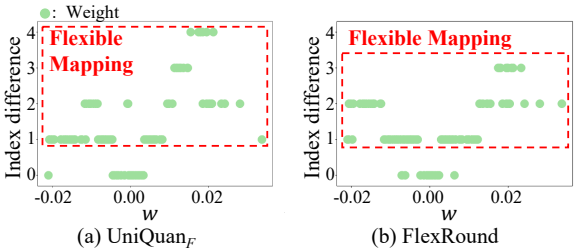


Figure 7: Index difference of mapped quantization levels per weight before and after applying $\boldsymbol{s}$ and s_r. Flexible mappings occur sufficiently in UniQuan$_F$ compared to FlexRound.

Experimental results show that the accuracy of the quantized models increases as the sample size grows. This is attributed to the fact a large number of data points promote the distillation of the general knowledge in the unquantized model to the quantized model. Therefore, UniQuan$_F$ achieves higher performance than that reported in the main text by providing enough data points.

F.6 Flexible Mappings in UniQuan$_F$

UniQuan$_F$ unifies FlexRound (Lee et al., 2023) and ALTERNATING. We perform an in-depth analysis to verify that UniQuan$_F$ effectively leverages the main optimization technique of FlexRound. The main idea of FlexRound is "flexible mapping" which introduces additional scale factors $\boldsymbol{s}$ and s_r to make weights explore diverse quantization levels, and selects the best one. We compare the amount of flexible mapping occurred in UniQuan$_F$ and FlexRound for validation. We use a 3-bit quantized Llama-3 8B model for analysis.

Case study. We analyze the amount of flexible mappings in a weight group of quantized models and visualize the results. Figure 7 represents the index difference of mapped quantization levels for each weight before and after applying scale factors $\boldsymbol{s}$ and s_r when quantizing Llama-3 8B models into 3-bit using UniQuan$_F$ and FlexRound. An index difference equal to or larger than 1 represents that the weight experiences the flexible mapping through $\boldsymbol{s}$ and s_r. Note that plenty of flexible mappings occur in the quantization process of UniQuan$_F$ similarly in FlexRound.

Global pattern. We analyze the proportion of flexible mapping applied across all model weights to examine that flexible mappings occur globally. Table 15 illustrates the proportion of the flexibly mapped weights in the entire model. The column names in the tables indicate the amount of changes

Table 15: Proportion of weights exhibiting flexible mapping across the entire model. Each column indicates the index difference of the mapped quantization level resulting from flexible mapping.

Method	Index Difference			
	0	**1**	**2**	**>2**
FlexRound	95.4132	4.5862	5.00e-04	1.30e-05
UniQuan$_F$	96.0395	3.8619	9.85e-02	7.62e-05

in indices of weights' mapped quantization levels resulting from flexible mapping. Across the entire model, UniQuan$_F$ demonstrates a similar level of flexible mapping as FlexRound. Therefore, UniQuan$_F$ effectively utilizes flexible mapping across the entire model as we intended.

G Use of AI Assistant

We use ChatGPT[4] and Gemini[5] only for grammar checking and sentence re-wording purposes. We do not use them for research purposes such as developing our main ideas or analyzing our experimental results.

H Potential Risks

In this paper, we propose UniQuan$_F$, a quantization method for large language models (LLMs), and there is a potential risk of losing the models' knowledge during quantization. In experiments, we rigorously validate the amount of knowledge loss in both general and task-specific aspects to verify whether the risk occurs. As a result, we demonstrate that UniQuan$_F$ results in significantly less knowledge loss than other methods, confirming its low risk.

[4] https://chatgpt.com/
[5] https://gemini.google.com/