QUQ: Quadruplet Uniform Quantization for Efficient Vision Transformer Inference

Xinkuang Geng[1], Siting Liu[2], Leibo Liu[3], Jie Han[4], Honglan Jiang[1]
[1] Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
[2] School of Information Science and Technology, ShanghaiTech University, Shanghai, China
[3] School of Integrated Circuits, Tsinghua University, Beijing, China
[4] Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada
xinkuang@sjtu.edu.cn, liust@shanghaitech.edu.cn, liulb@tsinghua.edu.cn, jhan8@ualberta.ca, honglan@sjtu.edu.cn

ABSTRACT

While exhibiting superior performance in many tasks, vision transformers (ViTs) face challenges in quantization. Some existing low-bit-width quantization techniques cannot effectively cover the whole inference process of ViTs, leading to an additional memory overhead (22.3%-172.6%) compared with corresponding fully quantized models. To address this issue, we propose quadruplet uniform quantization (QUQ) to deal with data of various distributions in ViT. QUQ divides the entire data range into at most four subranges that are uniformly quantized with different scale factors. To determine the partition scheme and quantization parameters, an efficient relaxation algorithm is proposed accordingly. Moreover, dedicated encoding and decoding strategies are devised to facilitate the design of an efficient accelerator. Experimental results show that QUQ surpasses state-of-the-art quantization techniques; it is the first viable scheme that can fully quantize ViTs to 6-bit with acceptable accuracy. Compared with conventional uniform quantization, QUQ leads to not only a higher accuracy but also an accelerator with lower area and power.

1 INTRODUCTION

With the development of deep learning techniques, neural networks (NNs) of an increasingly large scale have been applied to a wide range of domains. However, severe challenges arise when deploying these models onto edge devices with limited storage and computational capacity. Making on-device inference feasible for large models, quantization effectively compresses the models and reduces resource requirements without modifying the NN structure. Concerning data privacy and retraining costs, post-training quantization (PTQ) is receiving increasing attention.

Taking advantage of the attention mechanism [11], vision transformers (ViTs) achieve superior performance in various image processing tasks [3]. However, compared to convolutional neural networks (CNNs), the diverse computing modes in ViT introduce data with significantly different distribution characteristics, leading to great challenges in quantization.

When quantizing a ViT, many existing works [2, 12] focus only on the input of compute-intensive operations that can be implemented by general matrix multiplication (GEMM). However, other activations that are difficult to be uniformly quantized remain untouched, which results in a significant resource overhead for inference, including extra floating-point operations and increased memory. Additionally, some works adopt special encoding for specific data in ViT [4, 7, 12], which requires diverse computing hardware; thus, they are infeasible for hardware with a single type of arithmetic units or a common architecture. To support these quantization strategies, additional hardware is necessary.

To guarantee an efficient hardware implementation, ViT is expected to be fully quantized. Thus, a quantization technique that can be adaptable to diverse data distribution characteristics is needed. We observe that data in ViT shows certain common traits: ① Most elements cluster around zero, and outliers exhibit a wide range; ② positive and negative data show different distribution characteristics. Thus, we propose quadruplet uniform quantization (QUQ) to leverage these traits. Specifically, the entire data range is divided into at most four subranges based on the proposed progressive relaxation algorithm. The data belonging to each subrange are then uniformly quantized with a particular scale factor. Also, for certain data, QUQ allows the merging of encoding spaces between subranges with different signs, enabling dynamic adjustments of different data distribution characteristics.

Our major contributions are summarized as follows.

- We characterize the data distribution in ViT and propose QUQ accordingly to enable full quantization.
- A progressive relaxation algorithm is introduced to determine the quantization parameters of QUQ.
- The quadruplet uniform byte (QUB) is proposed to facilitate the encoding and decoding of QUQ, which is then utilized for the design of a QUQ-compatible accelerator.
- The performance of QUQ is evaluated in three ViT models for image classification. Experimental results show that QUQ results in higher accuracy than state-of-the-art quantization methods.
- Compared with the accelerator for uniform quantization, our design demonstrates superior efficiency in area and power while maintaining accuracy at a lower bit-width.

2 BACKGROUND

As shown in Figure 1, a typical ViT mainly consists of cascaded multi-head self-attention (MSA) modules and multi-layer perceptron (MLP) modules. In addition, layer normalization (LayerNorm) and residual connection (element-wise addition) are commonly performed before and after each module, respectively.

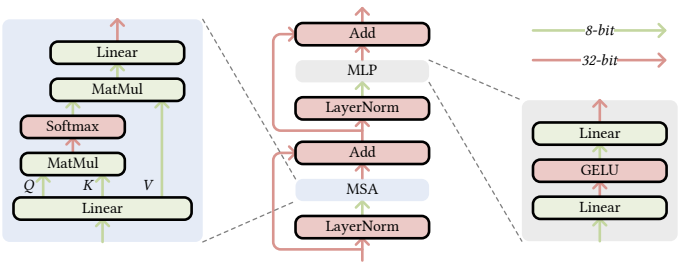

Figure 1: Data flow of a partially quantized ViT block.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3656516

In general, GEMM operations (Linear and MatMul) are quantized (shown in the green components in Figure 1), ensuring that most computations operate on low-bit-width integers [2, 12]. However, the inputs of residual connection, LayerNorm, Softmax, and GELU are not effectively quantized (shown in the red components); thus, high-bit-width activations exist in the data flow. This necessitates high-precision computation units and large intermediate storage for hardware deployment.

We simulate and count the required sizes of the on-chip memory for the ViT blocks (shown in Figure 1) of different scales during inference. In this simulation, we assume that only the weights required for the current operations are loaded during inference, as it is impractical to load the entire model into on-chip memory in edge devices. Additionally, considering the dynamic generation and usage of activations, it is assumed that they are always stored on-chip to avoid extra accesses to off-chip memory.

Figure 2 illustrates that, compared to the partially quantized ViT models (PQ), the fully quantized models (FQ) exhibit much lower peak memory demands. The predominance becomes more evident in small models, which is of particular concern for edge devices. Moreover, increasing the batch size, which enables an improved throughput, further enhances the superiority of the full quantization method. This occurs because a higher ratio of activations would occur when a larger batch size is utilized.

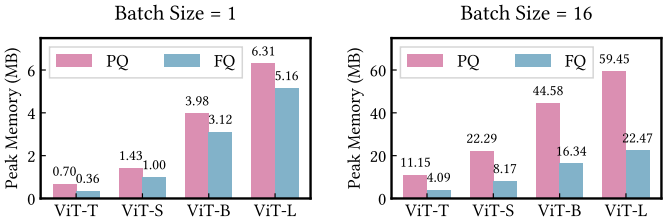

Figure 2: Peak memory usage in ViT blocks.

However, a low quantization bit-width leads to a significant accuracy drop in fully quantized ViT models. Therefore, to obtain the hardware benefits from full quantization, a technique that can effectively quantize data with various characteristics in ViT is urgently needed.

3 QUADRUPLET UNIFORM QUANTIZATION

3.1 Preliminaries

As the most commonly used data discretization method, symmetric uniform quantization [9] can be expressed as

$$\hat{x} = U_b(x;\Delta) = \text{clip}(\lfloor \frac{x}{\Delta} \rceil; -2^{b-1}, 2^{b-1}-1), \tag{1}$$

where b is the quantization bit-width, Δ represents the interval between adjacent discrete points and is also known as the scale factor. $\lfloor \cdot \rceil$ denotes the nearest rounding operation and clip$(\cdot)$ constraints the quantized data within the range of b-bit integer. A smaller Δ results in a higher quantization resolution but it causes more outliers to be clipped.

After quantization, a dot product $y_k = \sum x_i w_i$ that is commonly performed in GEMM can be approximated as

$$\hat{y}_k \approx \frac{1}{\Delta_y} \sum \Delta_x \hat{x}_i \Delta_w \hat{w}_i = \frac{\Delta_x \Delta_w}{\Delta_y} \sum \hat{x}_i \hat{w}_i \approx \frac{M}{2^N} \sum \hat{x}_i \hat{w}_i, \tag{2}$$

which indicates that all multiplications occur exclusively between the quantized values, as the scale factor Δ_x or Δ_w is shared by every element in the corresponding matrix. In some integer-only implementations [5, 6], the floating-point operations for the scale of an accumulation result are often replaced with multiplication and shift of integers, i.e., $\sum \hat{x}_i \hat{w}_i$ is multiplied by an integer M and then shifted right by N bits.

3.2 Quantization Scheme

Figure 3 shows the distributions of the weights for the query matrix in MSA, and the activations after Softmax, before element-wise addition, and after GELU, respectively. As per Figure 3, two observations are obtained. ① Data in ViT commonly follows a long-tailed distribution, i.e., most elements concentrate around zero and outliers lie within a broad range. In this case, determining an appropriate scale factor becomes challenging. A large Δ results in sparse resolution around zero, while a small Δ clips too many outliers to small values. ② Some data are not symmetrically distributed in the positive and negative parts, e.g., the output of GELU shows significant differences between the positive and negative parts, and the output of Softmax contains only non-negative values. Consequently, some discrete points of symmetric uniform quantization may not represent any elements, resulting in a waste of encoding space. To sum up, symmetric uniform quantization is not suitable for the quantization of ViT due to its various data distributions.

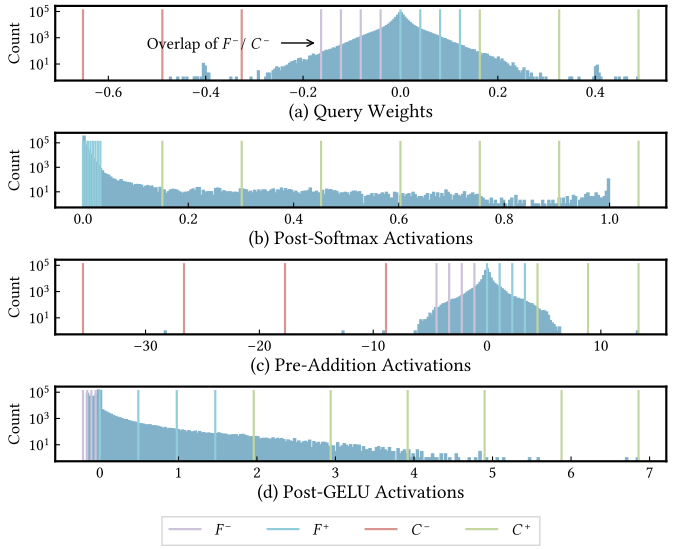

Figure 3: The distributions of weights and activations from different modules in ViT and the corresponding quantization points (vertical lines) generated by QUQ.

To effectively discretize these diverse data, we propose quadruplet uniform quantization (QUQ). For the cases where outliers appear on both sides of zero, as shown in Figures 3 (a) and (c), QUQ first divides the entire quantization range into the negative part R_{C^-} and the positive part R_{C^+} that are assigned coarse quantization intervals (scale factors), ensuring that outliers are not clipped. Subsequently, considering that most elements concentrate around zero, we further isolate two smaller ranges, R_{F^-} and R_{F^+}, also with zero as the boundary and being assigned fine intervals to reduce quantization error. Finally, uniform quantization is applied to each subrange based on its scale factor. Therefore, QUQ requires up to four different scale factors, namely Δ_{F^-}, Δ_{F^+}, Δ_{C^-}, and Δ_{C^+}, as shown in Mode A in Figure 4. b-bit QUQ can be implemented by four symmetric uniform quantizers as

$$\hat{x} = Q_b(x;\Delta_{F^-,F^+,C^-,C^+}) = \begin{cases} U_{b-1}(x;\Delta_{C^-}) & x \in R_{C^-} - R_{F^-} \\ U_{b-1}(x;\Delta_{F^-}) & x \in R_{F^-} \\ U_{b-1}(x;\Delta_{F^+}) & x \in R_{F^+} \\ U_{b-1}(x;\Delta_{C^+}) & x \in R_{C^+} - R_{F^+} \end{cases}. \tag{3}$$

Since each symmetric uniform quantizer U_{b-1} receives inputs from only one side of zero, it produces up to 2^{b-2} different quantization results. This means that we assign a quarter of the encoding space to each

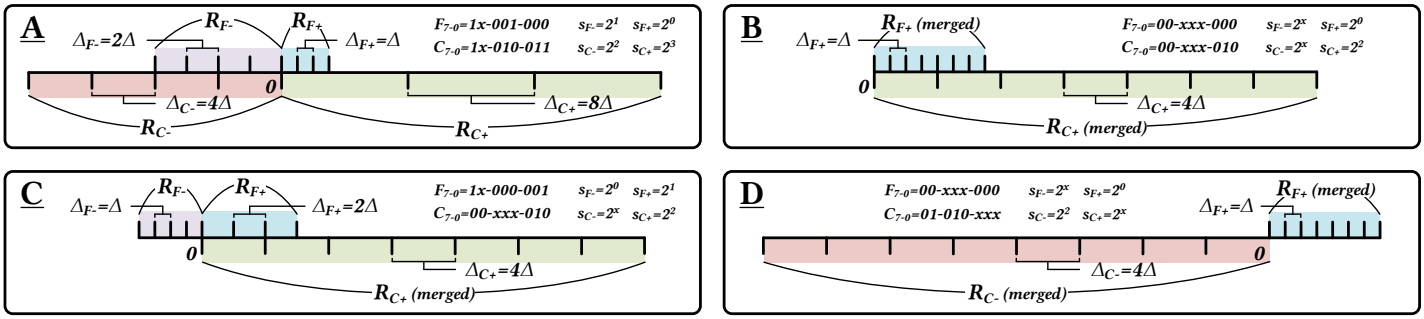

Figure 4: Quantization points of QUQ and the corresponding FC Registers in four different modes.

subrange, e.g., for $x \in R_{F^+}$, $\hat{x}$ is a $(b-2)$-bit unsigned integer. As shown in Figure 4, the overlap of different quantization points may occur because a coarse range may include a fine range. Although this results in potential encoding inefficiency, it is important that each subrange is bounded by zero. This ensures that the encoding of each quantization point is proportional to the original value, eliminating the need to store and process additional zero points.

So far, a dot product cannot be implemented solely through integer multiplications between quantized values as shown in (2), because the scale factor of each element can take four different values depending on the subrange it falls into. To reduce the hardware complexity, we enforce the relationship among the four scale factors as

$$\frac{\Delta_{F^-}}{s_{F^-}} = \frac{\Delta_{F^+}}{s_{F^+}} = \frac{\Delta_{C^-}}{s_{C^-}} = \frac{\Delta_{C^+}}{s_{C^+}} = \Delta, \quad s = 2^0, 2^1, 2^2, \cdots \tag{4}$$

Consequently, when performing a dot product, the shared Δ for the same vector can be extracted. In this way, the multiplication occurs only between quantized results scaled by integer powers of two that can be simplified into a shift operation on the product as

$$\begin{aligned} \hat{y}_k &= \frac{1}{s_{y_k}\Delta_y} \sum s_{x_i}\Delta_x\hat{x}_i\, s_{w_i}\Delta_w\hat{w}_i \\ &= \frac{1}{s_{y_k}} \frac{\Delta_x\Delta_w}{\Delta_y} \sum \hat{x}_i\hat{w}_i \ll (\log_2 s_{x_i} + \log_2 s_{w_i}). \end{aligned} \tag{5}$$

As discussed above, QUQ divides the entire range into four subranges, with each subrange being assigned a quarter of the encoding space. To better accommodate the diverse data distributions in ViT, QUQ enables the merging of two subranges with the same granularity.For instance, as shown in Figure 3 (d), as the negative part does not have outliers, R_{C^-} becomes unnecessary, allowing the corresponding encoding space to be utilized for finer quantization of R_{C^+}. Following this idea, the quantization points of QUQ can differentiate into several modes, as shown in Figure 4.

In Mode A, as the general form of QUQ, no merging occurs. In Mode B, merging occurs at both fine and coarse quantization subranges. Mode C is designed for the data without outliers on either side of zero, where the encoding spaces for the coarse subranges are merged. In Mode D, the fine and coarse subranges are merged separately, and their encoding spaces are assigned to the different sides of zero. As a result, both the positive and negative parts degenerate into uniform quantization.

Furthermore, setting $\Delta_{C^-} = \Delta_{F^+}$ in Mode D generates uniform quantization points throughout the entire range. Thus, symmetric uniform quantization can be considered as a special case of QUQ. This indicates that, with appropriate quantization settings, the performance of QUQ for any type of data will not be inferior to that of symmetric uniform quantization.

3.3 Progressive Relaxation Algorithm

As the quantization target of QUQ contains multiple modes, a customized strategy is needed to determine the quantization parameters based on the calibration data and ensure that the scale factors satisfy (4). To this end, we propose a progressive relaxation algorithm, which is formulated based on the following two guiding principles.

① The ratio between coarse and fine quantization subranges should be as large as possible, which reduces the encoding space wastage caused by the overlap of subranges.

② The fine quantization subrange should cover as many elements as possible, as it allows quantization with higher resolution.

First, we propose Algorithm 1 to relax two scale factors Δ_1 and Δ_2, one of which will be modified to satisfy (4), based on the rounding direction of L, which is the ratio of the two scale factors in the logarithmic domain. This ensures that the larger scale factor is not reduced and avoids clipping for the original data.

Algorithm 1 Relax Two Scale Factors.

Require: $\Delta_1 > 0$ and $\Delta_2 > 0$
Ensure: $\frac{\Delta_1}{\Delta_2} = 2^k, k = \cdots, -2, -1, 0, 1, 2, \cdots$

```
1 function RELAX(Δ1, Δ2)
2     L ← log2(Δ2/Δ1)
3     if ⌊L⌉ > L then   Δ1, Δ2 ← Δ1, 2^⌊L⌉ Δ1      ▷ make Δ2 larger
4     else              Δ1, Δ2 ← 2^-⌊L⌉ Δ2, Δ2     ▷ make Δ1 larger
5     return Δ1, Δ2
```

Based on Algorithm 1, the quantization parameters are determined by Algorithm 2. In this step, the coarse scale factors are obtained by using the maximum and minimum values of the calibration data as the boundaries for coarse uniform quantization. The boundaries for the fine subranges are set as the q-th quantile points, where the initial q is a hyperparameter. We apply three rounds of Algorithm 1 to ensure that these four scale factors satisfy (4), as shown in Lines 4 to 8.

Once the four scale factors are obtained under the assumption of Mode A, further relaxing or mode switching is performed by using four branches, as shown in Lines 10 to 17.

In the first branch, the ratios between the coarse and fine scale factors for both positive and negative parts fall below λ_A, indicating unacceptable wastage of encoding space. In this case, the current coarse-fine range partition scheme is considered unsuitable under the quantile point q; thus, Algorithm 2 is restarted with a smaller q, i.e., relaxing Principle ② to satisfy Principle ①. As the endpoint of the recursion, q_A also limits the minimum proportion of the data covered by fine subranges.

In the second and third branches, either the positive or negative part processes an unsuitable coarse-fine range partition scheme and has a sufficiently small boundary. In this case, uniform quantization is performed for the corresponding part with the initial coarse scale factor; the encoding space of the subrange is merged into that of the coarse subrange in the other side of zero, enhancing its quantization resolution. Note that, in the pseudo-code, setting the scale factor of

Algorithm 2 Progressive Relaxation Algorithm.

Input: Tensor $\mathbf{x}$, Quantization Bit-Width b, Acceptable Ratio λ_A of $\frac{\Delta_C}{\Delta_F}$, Initial Quantile q, Acceptable Quantile q_A

Output: Quantization Parameters $\Delta_{F^-}, \Delta_{F^+}, \Delta_{C^-}, \Delta_{C^+}$

function PRA($\mathbf{x}, b, \lambda_A, q, q_A$)
  ▷ *determine parameters for Mode A* ◁
  $\mathbf{x}^-, \mathbf{x}^+ \leftarrow -\mathbf{x}[\mathbf{x} < 0], \mathbf{x}[\mathbf{x} > 0]$
  $\Delta_{C^-}, \Delta_{C^+} \leftarrow \textsc{Relax}(\frac{\textsc{Max}(\mathbf{x}^-)}{2^{b-2}}, \frac{\textsc{Max}(\mathbf{x}^+)}{2^{b-2}-1})$ ▷ *relaxation round 1*
  $\Delta_{F^-}, \Delta_{F^+} \leftarrow \textsc{Relax}(\frac{\textsc{Quantile}(\mathbf{x}^-,q)}{2^{b-2}}, \frac{\textsc{Quantile}(\mathbf{x}^+,q)}{2^{b-2}-1})$ ▷ *relaxation round 2*
  $s_F, s_C \leftarrow \frac{\Delta_{F^-}}{\Delta_{F^+}}, \frac{\Delta_{C^-}}{\Delta_{C^+}}$ ▷ *record the ratios before relaxing F and C*
  $\Delta_{F^+}, \Delta_{C^+} \leftarrow \textsc{Relax}(\Delta_{F^+}, \Delta_{C^+})$ ▷ *relaxation round 3*
  $\Delta_{F^-}, \Delta_{C^-} \leftarrow s_F\Delta_{F^+}, s_C\Delta_{C^+}$ ▷ *Mode A*
  ▷ *further relax or switch the mode* ◁
  if $\frac{\Delta_{C^-}}{\Delta_{F^-}} < \lambda_A$ and $\frac{\Delta_{C^+}}{\Delta_{F^+}} < \lambda_A$ and $q > q_A$ **then**
    return PRA($\mathbf{x}, b, \lambda_A, q - 0.01, q_A$) ▷ *recursively relax*
  if $\frac{\Delta_{C^-}}{\Delta_{F^-}} < \lambda_A$ and $\Delta_{C^-} \leq \Delta_{F^+}$ **then**
    $\Delta_{F^-}, \Delta_{C^-}, \Delta_{C^+} \leftarrow \Delta_{C^-}, \varnothing, \frac{\Delta_{C^+}}{2}$ ▷ *Mode C*
  if $\frac{\Delta_{C^+}}{\Delta_{F^+}} < \lambda_A$ and $\Delta_{C^+} \leq \Delta_{F^-}$ **then**
    $\Delta_{F^+}, \Delta_{C^-}, \Delta_{C^+} \leftarrow \Delta_{C^+}, \frac{\Delta_{C^-}}{2}, \varnothing$ ▷ *Mode C*
  if $\frac{\Delta_{C^-}}{\Delta_{F^-}} < \lambda_A$ or $\frac{\Delta_{C^+}}{\Delta_{F^+}} < \lambda_A$ **then**
    $\Delta_{F^-}, \Delta_{F^+}, \Delta_{C^-}, \Delta_{C^+} \leftarrow \frac{\Delta_{C^-}}{2}, \varnothing, \varnothing, \frac{\Delta_{C^+}}{2}$ ▷ *Mode D*
  return $\Delta_{F^-}, \Delta_{F^+}, \Delta_{C^-}, \Delta_{C^+}$

the subrange to $\varnothing$ indicates that the subrange is merged. These two cases are mapped to Mode C, where the data on either side of zero lacks significant long-tailed distribution characteristics.

In the last branch, as a fallback, uniform quantization is applied to the positive and negative parts, respectively, aligning with Mode D.

Additionally, for a non-positive or non-negative tensor $\tilde{\mathbf{x}}$, it is first concatenated with $-\tilde{\mathbf{x}}$ to form a new tensor. Subsequently, the progressive relaxation algorithm is applied to obtain quantization parameters. Finally, the two scale factors corresponding to the parts of $-\tilde{\mathbf{x}}$ are set to $\varnothing$. This implements Mode B.

Figure 3 shows the 4-bit quantization points generated for data in different modules of ViT using the proposed algorithm. It can be seen that the obtained quantization points adequately match the corresponding data distributions. Additionally, we evaluate the quantization errors of uniform quantization (BaseQ) and QUQ for the four types of data in Figure 3, based on the mean squared error (MSE). The results for various quantization bit-widths are presented in Table 1. It shows that QUQ introduces smaller quantization errors than conventional uniform quantization.

Table 1: MSEs of Different Quantization Methods.

Method	Bit	Query W	Post-Softmax A	Pre-Addition A	Post-GELU A
BaseQ	4	2.14×10^{-4}	3.55×10^{-5}	3.19×10^{-1}	9.40×10^{-3}
QUQ	4	8.71×10^{-5}	2.34×10^{-5}	8.53×10^{-2}	1.78×10^{-3}
BaseQ	6	1.93×10^{-5}	8.80×10^{-6}	4.58×10^{-2}	2.97×10^{-3}
QUQ	6	5.59×10^{-6}	9.39×10^{-7}	5.31×10^{-3}	1.01×10^{-4}
BaseQ	8	1.23×10^{-6}	8.13×10^{-7}	4.11×10^{-3}	1.83×10^{-4}
QUQ	8	3.41×10^{-7}	6.13×10^{-8}	3.29×10^{-4}	6.00×10^{-6}

4 HARDWARE DESIGN

4.1 Encoding and Decoding Methods

To achieve efficient inference on hardware, we propose using quadruplet uniform bytes (QUBs) to encode the QUQ results. As shown in Figure 5, in addition to the base scale factor Δ, additional registers (denoted as FC Registers) are required for each tensor to store the QUQ mode and the ratios of the four scale factors to Δ.

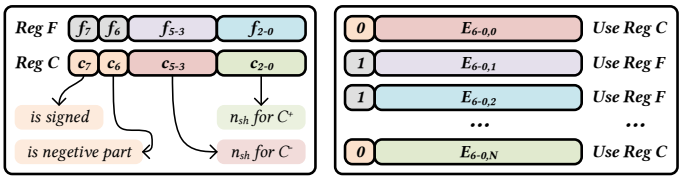

Figure 5: (Left) FC Registers. (Right) QUBs.

Two 8-bit registers are utilized to encode the parameters for coarse or fine subranges, denoted as FC Registers. For example, c_7 indicates whether the coarse subrange contains both positive and negative parts, to support the modes where the encoding spaces of different subranges are merged. If it contains only one part ($c_7 = 0$), then c_6 indicates whether the merged part is negative. c_{5-3} and c_{2-0} encode $\log_2 s_{C^-}$ and $\log_2 s_{C^+}$, respectively, representing the number of shifted bits n_{sh} when performing the multiplication as shown in (5).

For each QUB, taking 8-bit QUQ as an example, E_{7-0} is utilized to represent the encoded result, where E_7 indicates whether the QUB belongs to the fine or coarse subranges. If it belongs to coarse subranges ($E_7 = 0$), c_7 determines whether E_{6-0} represents a signed integer. ① If so, it means that R_{C^-} and R_{C^+} are not merged, and the sign bit E_6 determines which subrange the QUB falls into. ② If not, it means that R_{C^-} and R_{C^+} are merged, and c_6 indicates the reserved subrange, which only contains values on one side of zero. Therefore, in this case, the encoding for the sign bit is unnecessary, and E_{6-0} represents an 8-bit two's complement encoding without the sign bit. Based on the encoding rules, a straightforward decoding scheme is devised as

$$d = \begin{cases} \{1, E_{6-0}\} \ll f_{5-3} & \text{if } E_7\ (\overline{f_7}\, f_6 + f_7\, E_6) \\ \{0, E_{6-0}\} \ll f_{2-0} & \text{if } E_7\ (\overline{f_7}\, \overline{f_6} + f_7\, \overline{E_6}) \\ \{1, E_{6-0}\} \ll c_{5-3} & \text{if } \overline{E_7}\ (\overline{c_7}\, c_6 + c_7\, E_6) \\ \{0, E_{6-0}\} \ll c_{2-0} & \text{if } \overline{E_7}\ (\overline{c_7}\, \overline{c_6} + c_7\, \overline{E_6}) \end{cases}, \tag{6}$$

where the sign extension and the shift count n_{sh} are determined based on the subrange that the QUB falls into. After decoding, the output d can be represented by an 8-bit signed number D_{7-0} and a 3-bit n_{sh} as

$$d = \{sign, E_{6-0}\} \ll n_{sh} = D_{7-0} \ll n_{sh}. \tag{7}$$

It is noteworthy that the decoding result of an 8-bit unsigned QUB corresponding to Mode B is also expressed as an 8-bit signed number (with sign extension). This means that an 8-bit signed multiplier can accommodate QUB in any mode. In contrast, unsigned 8-bit integers cannot be processed by an 8-bit signed multiplier. Therefore, in uniform quantization, a signed multiplier of a higher bit-width is necessary to support unsigned integers.

4.2 Accelerator Design

We design a quadruplet uniform accelerator (QUA), as shown in Figure 6. As per the representation methods and arithmetic rules of QUB, the accelerator for QUQ can be supported by adding additional circuits to existing designs, shown as the red components in Figure 6.

Decoding unit (DU). DU is devised based on (6) and (7). It takes a QUB as the input and converts it into a signed integer D and an unsigned integer n_{sh}.

Processing element (PE) array. Each PE performs a multiply-accumulate operation and stores the intermediate results. Once the final result is generated, it is output to the quantization unit (QU) in turn. To support QUQ, the main change is that the inputs of the multiply-accumulate operation are no longer individual integers but

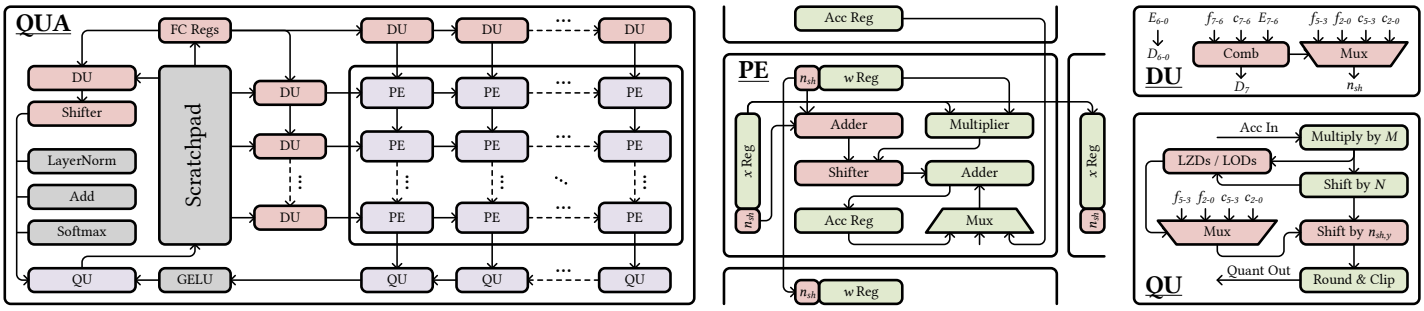

Figure 6: (Left) Quadruplet Uniform Accelerator. (Middle) Processing Element. (Right) Decoding Unit and Quantization Unit.

rather D and n_{sh} generated by DUs. As per (5), the hardware overhead of the arithmetic circuit is limited to a low-bit-width addition and a shift operation.

Quantization unit (QU). A traditional QU for uniform quantization scales the accumulated result followed by clipping and rounding operations [9]. Comparing (2) and (5), an additional right-shift operation is necessary to perform the scaling by s_{y_k}. Since the scale factor for y_k is dynamically determined based on the dot product result and FC Registers, it is necessary to compare it with the boundaries of the four subranges. These operations can be optimized by detecting the number of leading zeros or ones, as the boundary for comparison is either -2^b or $2^b - 1$, where $b \in \mathbb{N}$.

Special function unit (SFU). The inference process of a ViT requires more than just GEMM. Special functions are also needed to implement LayerNorm, element-wise addition, Softmax and GELU. We introduce a QUB decoder and a shifter in the data loading path of SFUs to convert the encoded QUB into an integer $d = D \ll n_{sh}$. Consequently, we can streamline the SFUs to perform the same functions as the accelerator designed for uniform quantization in [5, 6].

It can be seen that we only insert some conversion circuits without altering the original data flow. QUA can be considered more as an integration method rather than an architecture itself. Therefore, existing techniques for NN accelerators can also be employed to enhance hardware efficiency.

5 RELATED WORKS

PTQ4ViT [12] and APQ-ViT [2] focus on partially quantized ViT models, i.e., they do not consider input activations of some operations such as residual connection and LayerNorm. Notably, PTQ4ViT uses twin uniform quantization for specific activation values, which can be considered as a subset of QUQ.

BiScaled-FxP [4] records an index table to identify outliers for each tensor and applies an additional scale factor to quantize them. While BiScaled-FxP is effective in handling non-negative activations and symmetrically distributed weights in CNNs, it is unsatisfactory when dealing with data exhibiting diverse distribution patterns in ViT. Furthermore, the index table introduces unpredictable overhead when there are numerous outliers to be indexed.

FQ-ViT [7] enables a full quantization for ViT. However, it employs row-wise quantization for weights and specific activations, leading to distinct quantization parameters for different row vectors within a matrix. Row-wise scheme incurs additional memory overhead and complexity to the computation and quantization, and may not be supported by existing architectures [9].

I-Bert [5] and I-ViT [6] investigate integer-only inference for transformers. It is noteworthy that, although they avoid floating-point operations at all stages in Figure 1, 32-bit integer activations are still required to maintain accuracy. Consequently, there is no actual reduction in memory overhead.

6 EXPERIMENTS

6.1 Accuracy Evaluation

To evaluate the performance of the proposed QUQ, PTQ experiments are conducted for image classification on ImageNet [1]. Three models are considered, including ViT [3], DeiT [10], and Swin [8].

Experimental details. We randomly select 32 images from the training dataset of ImageNet as the input for calibration to obtain quantization parameters. After obtaining the four scale factors following the steps of our proposed algorithm, we employ a grid search similar to [12] to conduct a layer-wise Hessian-based optimization. For the hyperparameters in Algorithm 2, in all experiments, the acceptable ratio s_A, initial quantile q, and acceptable quantile q_A are set to 4, 0.99, and 0.95, respectively.

For partial quantization, our method is compared with PTQ4ViT [12] and APQ-ViT [2] under 6-bit quantization. For a fair comparison, only operations that can be implemented by GEMM are quantized to 6-bit, while the remaining parts are retained in floating-point format. Additionally, to further evaluate the effectiveness of QUQ, we substitute QUQ with uniform quantization while maintaining the rest of the PTQ process unchanged, denoted as BaseQ. Table 2 shows the Top-1 accuracy of different quantization methods across various models.

Table 2: Accuracy Comparison of Partially Quantized ViTs.

Method	W/A	ViT-S	ViT-L	DeiT-S	DeiT-B	Swin-T	Swin-S
Original	32/32	81.39	85.84	79.80	81.80	81.39	83.23
BaseQ	6/6	69.73	80.96	72.55	78.94	78.44	82.04
PTQ4ViT*	6/6	78.63	85.05	76.28	80.25	80.47	82.38
APQ-ViT†	6/6	79.10	-	77.76	80.42	-	82.76
QUQ	6/6	**79.65**	**85.57**	**78.73**	**81.60**	**80.95**	**83.06**

* Some activations are not uniformly quantized.
† Block-wise Hessian information is considered to optimize the parameters instead of the layer-wise counterpart in others.

It can be seen that QUQ surpasses state-of-the-art quantization methods for partial quantization of ViTs. Compared to the full precision models, 6-bit QUQ results in less than 0.3% accuracy drop for ViT-L, DeiT-B, and Swin-S, yet a larger drop for models with the same architecture of a smaller scale (ViT-S, DeiT-S, and Swin-T).

For full quantization, our method is compared with FQ-ViT [7] and BiScaled-FxP [4] under 6-bit and 8-bit quantization, as shown in Table 3. Since BiScaled-FxP conducts experiments only on CNNs, the relevant experimental results are reproduced based on the method described in [4]. Note that the optimization techniques used in QUQ are also applied to BiScaled-FxP.

Table 3 shows that QUQ exhibits a more pronounced advantage than state-of-the-art works for fully quantized models. Although a more significant accuracy drop can be observed in 6-bit quantization, QUQ is, to the best of our knowledge, the first method that can produce usable results in this case.

Table 3: Accuracy Comparison of Fully Quantized ViTs.

Method	W/A	ViT-S	ViT-L	DeiT-S	DeiT-B	Swin-T	Swin-S
Original	32/32	81.39	85.84	79.80	81.80	81.39	83.23
BaseQ	6/6	0.10	0.10	0.09	0.17	0.10	00.41
BiScaled-FxP	6/6	0.30	5.94	0.64	7.89	0.16	00.39
FQ-ViT†	6/6	9.92	6.86	60.14	68.84	36.25	70.17
QUQ	6/6	**69.43**	**78.51**	**69.96**	**76.17**	**76.05**	**79.14**
BaseQ	8/8	1.00	2.44	66.26	31.37	55.90	40.06
BiScaled-FxP	8/8	72.37	70.70	78.40	73.73	80.04	77.41
FQ-ViT†	8/8	79.49	85.03	79.17	81.20	80.51	82.71
QUQ	8/8	**80.49**	**85.73**	**79.40**	**81.40**	**81.24**	**83.25**

† Weights and certain activations are quantized row-wise.

To evaluate the impact of QUQ on the attention mechanism in fully quantized ViT, we select some images from the validation dataset of ImageNet and visualize the attention maps, as shown in Figure 7. For the 8-bit case, the attention of uniform quantization in crucial regions begins to decrease, while the attention of QUQ remains relatively constant, compared to the original. For the 6-bit case, the attention of uniform quantization is no longer activated, while QUQ still effectively maintains attention in crucial regions.

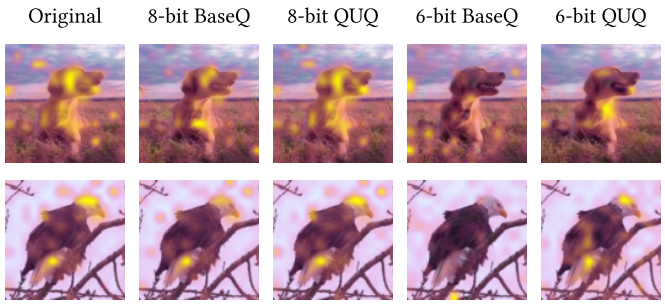

Figure 7: Attention map visualization for ViT-S.

6.2 Hardware Evaluation

We evaluate the hardware overhead of QUQ by comparing the proposed QUA with the one for uniform quantization. The evaluation model is implemented based on the architecture depicted in Figure 6. Since QUQ can utilize the same SFUs and scratchpad memory as uniform quantization, they are not taken into consideration. All designs are synthesized under a consistent constraint, leveraging Synopsys Design Compiler on a 28 nm CMOS technology, with power reported through PrimeTime PX under a 500 MHz clock. The evaluation results are shown in Table 4.

Table 4: Area and Power of Various NN Accelerators.

Method	W/A	16 × 16 PE Array		64 × 64 PE Array	
		Area(mm^2)	Power(mW)	Area(mm^2)	Power(mW)
BaseQ	6/6	0.148	52.4	2.205	701.3
QUQ	6/6	0.153	57.2	2.247	767.5
BaseQ	8/8	0.175	60.6	2.702	796.7
QUQ	8/8	0.182	65.1	2.714	851.6

It shows that QUQ incurs marginal hardware overhead compared to uniform quantization, exhibiting less than 5% and 10% overheads in area and power, respectively, for the considered cases. Increasing the size of the PE array reduces the relative area overhead of the accelerator. This phenomenon can be attributed to the decreasing proportion of additional circuits required to support DUs and QUs, compared to the quadratic growth of PEs. Additionally, the increase in power mainly stems from the additional registers required to pipeline n_{sh}, which further increases the clock load.

It is noteworthy that 6-bit QUQ achieves significantly higher accuracy than 8-bit BaseQ across all models (see Table 3), yet results in 12.6%-16.8% and 3.7%-5.6% reductions in area and power consumption, respectively. Moreover, reducing the quantization bit-width further decreases the memory overhead.

7 CONCLUSION

To support efficient quantization for data with various distribution characteristics, we propose quadruplet uniform quantization (QUQ). A progressive relaxation algorithm is devised for QUQ to select suitable quantization parameters. Furthermore, we encode the quantization results as quadruplet uniform bytes (QUBs) and design a quadruplet uniform accelerator (QUA). The experimental results show that QUQ results in higher accuracy than state-of-the-art PTQ methods for ViT, especially for fully quantized models. While achieving higher accuracy, QUQ requires lower area, power, and memory than conventional uniform quantization.

It is noteworthy that, besides ViTs, QUQ is inherently capable of effectively quantizing the other NN models. Given its compatibility with uniform quantization and ease of deployment, QUQ can serve as a versatile extension for any NN accelerator, which offers an additional option for software and hardware co-optimization.

8 ACKNOWLEDGEMENTS

This work was supported in part by the National Key Research and Development Program of China under grant 2022YFB4500200 and in part by the National Natural Science Foundation of China under grant number 62374108.

REFERENCES

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[2] Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. 2022. Towards Accurate Post-Training Quantization for Vision Transformer. In *Proceedings of the 30th ACM International Conference on Multimedia*. 5380–5388.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[4] Shubham Jain, Swagath Venkataramani, Vijayalakshmi Srinivasan, Jungwook Choi, Kailash Gopalakrishnan, and Leland Chang. 2019. BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.

[5] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *International conference on machine learning*. PMLR, 5506–5518.

[6] Zhikai Li and Qingyi Gu. 2023. I-vit: Integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17065–17075.

[7] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. 2021. Fq-vit: Post-training quantization for fully quantized vision transformer. *arXiv preprint arXiv:2111.13824* (2021).

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[9] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295* (2021).

[10] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.

[11] 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 (2017).

[12] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. 2022. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *European Conference on Computer Vision*. Springer, 191–207.