AdaLog: Post-Training Quantization for Vision Transformers with Adaptive Logarithm Quantizer

Zhuguanyu Wu[1,2], Jiaxin Chen[1,2](✉), Hanwen Zhong[1,2], Di Huang[2], and Yunhong Wang[1,2]

[1] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
[2] School of Computer Science and Engineering, Beihang University, Beijing, China
{goatwu,jiaxinchen,hanwenzhong,dhuang,yhwang}@buaa.edu.cn

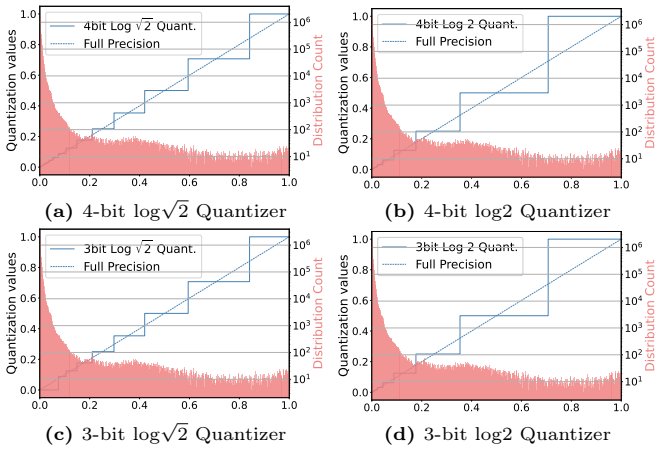
Abstract. Vision Transformer (ViT) has become one of the most prevailing fundamental backbone networks in the computer vision community. Despite the high accuracy, deploying it in real applications raises critical challenges including the high computational cost and inference latency. Recently, the post-training quantization (PTQ) technique has emerged as a promising way to enhance ViT's efficiency. Nevertheless, existing PTQ approaches for ViT suffer from the inflexible quantization on the post-Softmax and post-GELU activations that obey the power-law-like distributions. To address these issues, we propose a novel non-uniform quantizer, dubbed the Adaptive Logarithm AdaLog (AdaLog) quantizer. It optimizes the logarithmic base to accommodate the power-law-like distribution of activations, while simultaneously allowing for hardware-friendly quantization and de-quantization. By employing the bias reparameterization, the AdaLog quantizer is applicable to both the post-Softmax and post-GELU activations. Moreover, we develop an efficient Fast Progressive Combining Search (FPCS) strategy to determine the optimal logarithm base for AdaLog, as well as the scaling factors and zero points for the uniform quantizers. Extensive experimental results on public benchmarks demonstrate the effectiveness of our approach for various ViT-based architectures and vision tasks including classification, object detection, and instance segmentation. Code is available at https://github.com/GoatWu/AdaLog.

Keywords: Post-training quantization · Vision Transformer · Adaptive logarithm quantizer · Progressive searching

1 Introduction

Along with the success of Transformers in natural language processing [29], Vision Transformer (ViT) has become a prevailing deep neural network architecture in the computer vision community, achieving promising performance for a variety of vision tasks such as image classification [3, 7, 21], object detection [2,5,36,38,39], semantic segmentation [26,33,37], and action recognition [13].

✉ Corresponding author.



(a) 4-bit $\log\sqrt{2}$ Quantizer **(b)** 4-bit log2 Quantizer

(c) 3-bit $\log\sqrt{2}$ Quantizer **(d)** 3-bit log2 Quantizer

Fig. 1: Histogram of post-Softmax activations. (a)-(b): In 4-bit quantization, the $\log\sqrt{2}$ quantizer allocates more bits to the relatively important large values compared to the log2 quantizer, thus reaching higher accuracy. (c)-(d): In 3-bit quantization, the $\log\sqrt{2}$ quantizer quantizes the majority of values to 0, leading to significant degradation.

Nonetheless, the advancement of vision transformers in accuracy comes at the cost of increased model size and slow inference speed, substantially hindering its applicability in practice, especially when deploying on resource-constrained mobile or edge devices [12].

Recently, model quantization has emerged as an effective way to compress and accelerate deep models by mapping their weights or activations with full precision into integers of lower bit-width. Existing approaches for model quantization mainly fall into the following two categories: the Quantization Aware Training (QAT) [4, 8] and the Post-Train Quantization (PTQ) [10, 24]. Despite the advantage in accuracy, QAT usually needs to retrain the model on the entire training dataset, taking a considerable amount of time and computational cost. By contrast, PTQ merely requires a small-scale validation set to obtain a quantized model with even comparable accuracy, thus being much more efficient.

In this paper, we mainly focus on the PTQ-based approaches. Most of these methods achieve promising performance with sufficiently large bit-width, but their accuracy drops sharply when quantizing with extremely low bit-width (*e.g.* 4 bits and below). Actually, they suffer from the following two limitations. 1) *Inflexible Logarithm Base.* The representative logarithm-based non-uniform quantizers [18, 20] adopt a fixed base, *i.e.* 2 or $\sqrt{2}$, to deal with the power-law-like activation distributions. As shown in Fig. 1, the $\log 2$ quantizer incurs substantial rounding errors for large activations under 4-bits, while the $\log\sqrt{2}$ quantizer suffers from truncation errors for small activations under 3-bits. Moreover, the value range of post-GELU activations differs significantly in distinct layers as displayed in Fig. 4. This implies that the current logarithm quantizers with fixed bases cannot adaptively search for an optimal partitioning on the truncation in-

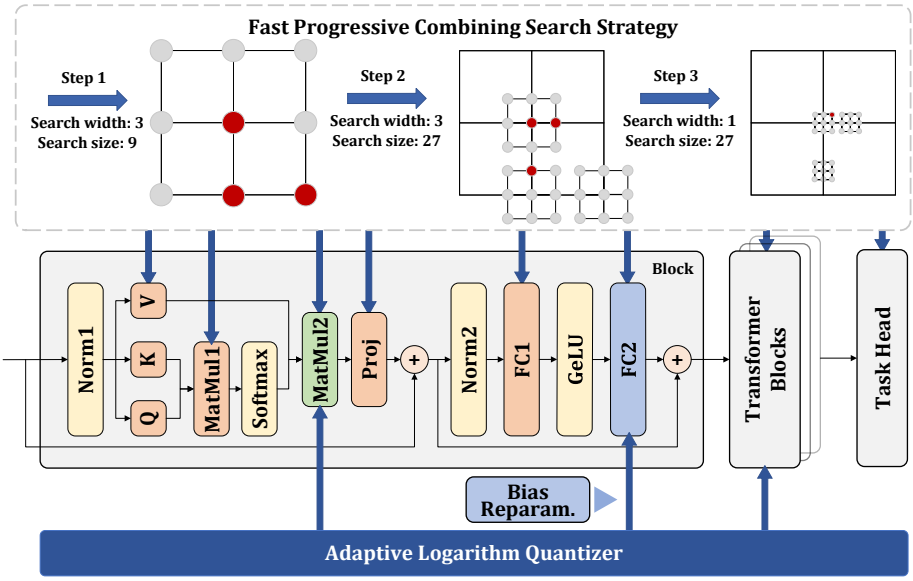


Fig. 2: Illustration on the framework of our method. The AdaLog quantizer is employed for quantizing the post-Softmax and post-GELU activations, where the bias reparameterization is specifically integrated to extend AdaLog to the post-GELU layers. The Fast Progressive Combining Search (FPCS) strategy facilitates AdaLog to search for the optimal scaling factors and logarithm base, as well as the scaling factors and zero points of the uniform quantizers.

terval as the data or bit-width varies, thus deteriorating the ultimate accuracy. In the meantime, the $\log\sqrt{2}$ quantizer fails to avoid the floating-point multiplication as shown in Fig. 3(b), making it not hardware-friendly. 2) *Excessively sparse partition of hyperparameter search space.* Given the wide distribution range of ViT activations, the potential value range for the corresponding scaling factor also becomes broad. The conventional grid search usually adopts a uniform sparse partitioning of the entire search space by considering the search efficiency, which however is prone to fall into a local optimum.

To address the above issues, as illustrated in Fig. 2, we propose a novel quantizer, namely **Ada**ptive **Log**arithm (**AdaLog**) Quantizer, for post-training quantization of vision transformers. To deal with the *inflexible logarithm base*, AdaLog firstly establishes the quantization and the de-quantization process with an arbitrary base, which allows for efficient computation in the integer form, thus being hardware-friendly. The optimal logarithm base is subsequently determined via hyperparameter searching. By further employing the bias reparameterization, AdaLog is applicable to quantize both the post-Softmax and post-GELU activations. To tackle the *sparse hyperparameter search space*, we develop a **F**ast **P**rogressive **C**ombining **S**earch (**FPCS**) strategy that divides the search space more finely without increasing the searching complexity, compared to the previ-

ous grid search methods. Besides, this strategy can be used not only for general uniform quantizers but also for the base search of AdaLog quantizers.

The main contributions of our work are summarized in three-fold:

1. We propose a novel quantizer, dubbed AdaLog, for post-training quantization of vision transformers. This non-uniform quantizer adapts the logarithmic base to accommodate the power-law distribution of activations and simultaneously allows for hardware-friendly quantization and de-quantization. It is applicable to both the post-Softmax and post-GELU activations, by employing the bias reparameterization.
2. We develop an efficient hyperparameter search strategy, namely the Fast Progressive Combining Search strategy. Compared to the conventional uniform grid search, it is able to locate the optimal hyperparameter more precisely without sacrificing the efficiency, thus being more suitable for quantizers with multiple hyperparameters.
3. We extensively evaluate the performance of the proposed method on various computer vision tasks including classification, object detection, and instance segmentation. The experimental results demonstrate that our method significantly outperforms the state-of-the-art approaches with distinct vision transformer architectures, especially in the case of low-bit quantization.

2 Related Work

2.1 Vision Transformer

Vision Transformer (ViT) and its variants have emerged as pivotal backbone networks in the computer vision community. ViT [7] firstly introduces the self-attention mechanism to the image classification task, eschewing convolutional layers in favor of Transformer blocks, thereby unveiling the potential of Transformers for visual representation learning. To enhance the efficiency, DeiT [28] integrates the knowledge distillation technique to optimize the training in the data-restricted scenario. Meanwhile, the Swin Transformer [21] adopts the hierarchical design and localized windowed self-attention to reduce the computational overhead while strengthening its capability of mining the long-range dependency.

Due to the intensive matrix multiplication operations involved, the vision transformers usually take a considerable amount of time and memory cost, hindering their deployment in practical applications. To overcome these drawbacks, MobileViT [22] and LeViT [11] attempt to design lightweight vision transformer structures. Alternatively, Token Merging [1] and X-pruner [34] endeavor to expedite the inference speed of ViTs through the pruning technique.

2.2 Model Quantization

Model quantization is a critical technique for model compression, aiming at mapping the floating-point weights and activations to integers or values of even lower

bit-width. Most of the existing approaches can be categorized into Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). The QAT approaches [8,15,30] usually achieve high accuracy, but are computationally intensive, as they retrain the model on the entire training dataset.

By contrast, the PTQ method only needs to calibrate on small-scale data, thus being suitable for rapid deployment. AdaRound [23], BRECQ [17], and QDrop [31] are pioneering works on PTQ, but they focus on the convolutional neural networks. Several recent works have explored PTQ for vision transformers. FQ-ViT [20] designs a Power-of-Two Factor for LayerNorm quantization and a $\log 2$ quantizer for softmax quantization. Based on FQ-ViT, Evol-Q [9] introduces small perturbations in quantization and utilizes an InfoNCE loss to enhance the performance. EasyQuant [32] employs an alternating optimization strategy for weights and activations to mitigate the quantization loss. By following EasyQuant, PTQ4ViT [35] introduces the Twin-Uniform quantizer to address power-law distributions and advances a Hessian-guided search strategy for optimization. APQ-ViT [6] boosts the Hessian-guided approach with Blockwise the Bottom-elimination Calibration and incorporates a scale parameter in the quantization of attention maps to preserve the Matthew effect. RepQ-ViT [18], focuses on quantizing the Post-LayerNorm layer by employing the reparameterization technique to balance the large activation quantization errors with small weight quantization inaccuracies. It further introduces the $\log \sqrt{2}$ quantizer to promote the accuracy. However, current methods still severely suffer from the substantial quantization loss at low bit-width.

3 The Proposed Approach

Overview As displayed in Fig. 2, there are four linear layers in a standard ViT block, including *QKV*, *Proj*, *FC1* and *FC2*, as well as two matrix multiplications denoted by *MatMul1* and *MatMul2*. Existing approaches [18] have extensively studied quantizing the *QKV* and *FC1* layers. However, the post-Softmax and the post-GELU layers, including *MatMul2* and *FC2*, have not been properly handled yet, thus still suffering from non-negligible quantization loss. In this paper, we first introduce some preliminaries, and employ an adaptive logarithm quantizer (AdaLog) for post-Softmax and post-GELU layers, as detailed in Sec. 3.2 and Sec. 3.3 respectively. Moreover, to address the issue of sparse partition of hyper-parameter search space in low-bit quantization, the Fast Progressive Combining Search (FPCS) strategy is employed in all the quantized layers, which is elaborated in Sec. 3.4.

3.1 Preliminaries

Power-Law Distribution of Activations. As displayed in Fig. 1, the post-Softmax activations exhibit a power-law probability distribution, making it challenging for model quantization. The $\log 2$ quantizer [20] and $\log \sqrt{2}$ quantizer [18] have attempted to deal with the above problem by non-uniformly partitioning

the truncation intervals with fixed logarithm bases, which are briefly described as below.

Log2 Quantizer. The $\log 2$ quantizer [20] is a common choice to deal with the power-law activation distributions, which can be formulated as:

$$\textit{Quantization} : \boldsymbol{A}^{(\mathbb{Z})} = \text{clamp}\left(\left\lfloor -\log_2 \frac{\boldsymbol{A}}{s} \right\rceil, 0, 2^{bit}-1\right), \tag{1}$$

$$\textit{De-quantization} : \widehat{\boldsymbol{A}} = s \cdot 2^{-\boldsymbol{A}^{(\mathbb{Z})}}, \tag{2}$$

where $\lfloor \cdot \rceil$ denotes the rounding function, $s \in \mathbb{R}^+$ is the scaling factor and *bit* is the quantization bit-width. By leveraging the efficient bit-shift operation, the $\log_2(\cdot)$ function and the power-of-2 function can be implemented even faster than integer multiplication, thus being hardware-friendly.

Log$\sqrt{2}$ Quantizer. The $\log \sqrt{2}$ quantizer [18] adopts the scale reparameterization technique, formulated as below:

$$\textit{Quantization} : \boldsymbol{A}^{(\mathbb{Z})} = \text{clamp}\left(\left\lfloor -2\log_2 \frac{\boldsymbol{A}}{s} \right\rceil, 0, 2^{bit}-1\right), \tag{3}$$

$$\textit{De-quantization} : \widehat{\boldsymbol{A}} = \widetilde{S} \cdot 2^{\lfloor -\frac{\boldsymbol{A}^{(\mathbb{Z})}}{2} \rfloor}, \tag{4}$$

where $\widetilde{S} = s \cdot \left(\mathbb{1}[x^{(\mathbb{Z})}] \cdot (\sqrt{2}-1)+1\right)$ is the reparameterized scale and $\mathbb{1}[\cdot]$ is a parity indicator function.

However, due to the differing parity of $x^{(\mathbb{Z})}$ at various positions, $\widetilde{S}$ becomes an element-wise floating-point scaling matrix. As shown in Fig. 3(b), when quantizing the multiplication between two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the $\log \sqrt{2}$ quantizer needs to conduct the Hadamard product between the reparameterized scale $\widetilde{S}$ and $\boldsymbol{A}^{(\mathbb{Z})}$, and then multiplying with $\boldsymbol{B}^{(\mathbb{Z})}$, where $\boldsymbol{A}^{(\mathbb{Z})}$ and $\boldsymbol{B}^{(\mathbb{Z})}$ denote the quantized form of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. As it is unable to infer in the integer form, the $\log \sqrt{2}$ quantizer is therefore not hardware-friendly.

Both the $\log 2$ and $\log \sqrt{2}$ quantizers are inflexible in searching for an optimal partition as they adopt fixed logarithm bases, thus leaving much room for improvement, especially when performing with extremely low bits (*e.g.* 4 bits and below).

3.2 Adaptive Logarithm Base Quantizer

To overcome the limitations of the log2 quantizer and the $\log\sqrt{2}$ quantizer, we propose the AdaLog quantizer by adaptively searching for an optimal logarithmic base rather than adopting a fixed one.

Specifically, given the activation $\boldsymbol{A}$, a logarithm base $b \in \mathbb{R}^+$, the bit-width *bit* and the scaling factor $s \in \mathbb{R}$, the quantization process of $\boldsymbol{A}$ is formulated as:

$$\begin{aligned} \boldsymbol{A}^{(\mathbb{Z})} &= \text{clamp}\left(\left\lfloor -\log_b \frac{\boldsymbol{A}}{s} \right\rceil, 0, 2^{bit}-1\right) \\ &= \text{clamp}\left(\left\lfloor -\frac{\log_2 \frac{\boldsymbol{A}}{s}}{\log_2 b} \right\rceil, 0, 2^{bit}-1\right), \end{aligned} \tag{5}$$

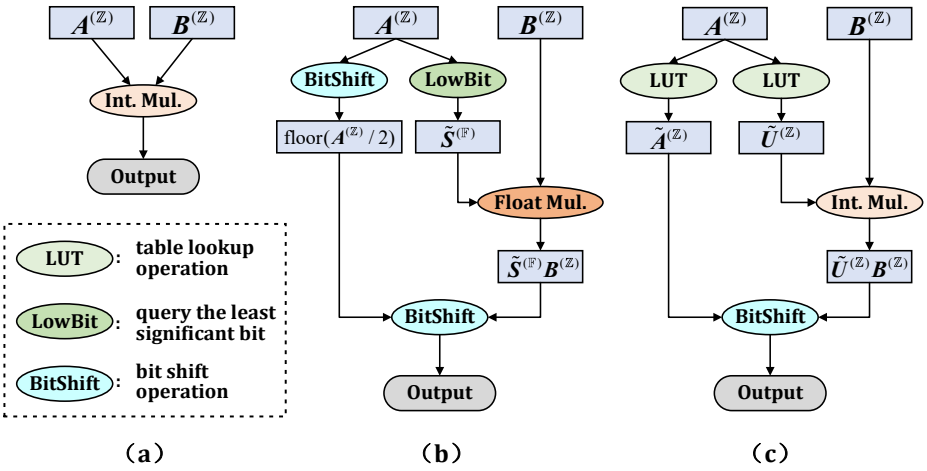


Fig. 3: (a) is the flowchart for linear quantized data. (b) shows the flowchart of the $\log\sqrt{2}$ quantizer [18] that fails to avoid the floating-point multiplication operation, which is not hardware-friendly. (c) displays the flowchart of the proposed AdaLog method, which only takes two extra table lookup operations and one bit-shift operation compared to the standard linear integer multiplication, making it efficient and hardware-friendly.

and the de-quantization of $\boldsymbol{A}^{(\mathbb{Z})}$ is written as the following:

$$\widehat{\boldsymbol{A}} = s \cdot b^{-\boldsymbol{A}^{(\mathbb{Z})}}. \tag{6}$$

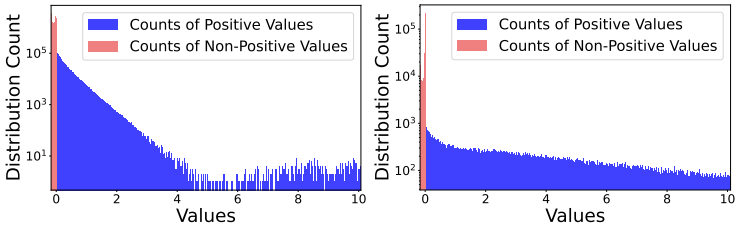
However, similar to the $\log\sqrt{2}$ quantizer, for a general base b rather than the base 2, the computation of $b^{-\boldsymbol{A}^{(\mathbb{Z})}}$ in Eq. (6) cannot be expedited through bit shift operations. To overcome this problem, we first approximate $\log_2 b$ using a rational number, *i.e.*, $\log_2 b \approx q/r$, where $q, r \in \mathbb{Z}^+$. By applying it to Eq. (6), the de-quantization can be reformulated as the following by employing the change-of-base formula:

$$\widehat{\boldsymbol{A}} = s \cdot b^{-\boldsymbol{A}^{(\mathbb{Z})}} = s \cdot \left(2^{-\widetilde{\boldsymbol{A}}^{(\mathbb{Z})}} \circ 2^{-\widetilde{\boldsymbol{U}}}\right), \tag{7}$$

where

$$\widetilde{\boldsymbol{A}}^{(\mathbb{Z})} = \left\lfloor \frac{q \cdot \boldsymbol{A}^{(\mathbb{Z})}}{r} \right\rfloor, \quad \widetilde{\boldsymbol{U}} = \frac{\left(q \cdot \boldsymbol{A}^{(\mathbb{Z})}\right) \bmod r}{r}. \tag{8}$$

Since the elements of $\boldsymbol{A}^{(\mathbb{Z})}$ belong to $\{0, 1, \cdots, 2^{bit}-1\}$, we can observe from Eq. (8) that the elements of both $\widetilde{\boldsymbol{A}}^{(\mathbb{Z})}$ and $\widetilde{\boldsymbol{U}}$ also distribute in a finite discrete field. This implies that we can record the value range of $\widetilde{\boldsymbol{A}}^{(\mathbb{Z})}$ and $2^{-\widetilde{\boldsymbol{U}}}$ via two separate tables, once we determine the hyperparameters q and r for each layer. By this means, we only need to perform two table lookup operations to obtain $\widetilde{\boldsymbol{A}}^{(\mathbb{Z})}$ and $2^{-\widetilde{\boldsymbol{U}}}$ in Eq. (7) instead of directly calculating Eq. (8) by floating-points during the inference process.



(a) blocks.6.mlp.fc2 **(b)** blocks.10.mlp.fc2

Fig. 4: Illustration on the distribution of post-GeLU activations. (a) and (b) are the distributions of post-GeLU activation values from different layers of ViT-Base. It can be observed that although they both follow power-law-like distributions, their value ranges substantially differ, showing the necessity of adaptive logarithm bases.

As shown in Fig. 3(c), the de-quantization process for the multiplication between $\boldsymbol{A}$ and $\boldsymbol{B}$ involves the operation $(2^{-\widetilde{\boldsymbol{U}}} \circ 2^{-\widetilde{\boldsymbol{A}}^{(\mathbb{Z})}})\boldsymbol{B}^{(\mathbb{Z})}$, which is not hardware-friendly as $2^{-\widetilde{\boldsymbol{U}}}$ is a floating-point matrix. To overcome this drawback, we quantize the recorded table of $2^{-\widetilde{\boldsymbol{U}}}$ by a uniform quantizer. Since its value falls in the range of $(0.5, 1]$ as in Eq. (9), we adopt the scaling factor $s_{\text{table}} = 1/(2 \cdot (2^{bit} - 1))$ and quantize $\widetilde{\boldsymbol{U}}$ into

$$\widetilde{\boldsymbol{U}}^{(\mathbb{Z})} = \left\lfloor \frac{2^{-\widetilde{\boldsymbol{U}}}}{s_{\text{table}}} \right\rceil . \tag{9}$$

By virtue of the above steps, we can quickly obtain the reparameterized matrix $\widetilde{\boldsymbol{U}}^{(\mathbb{Z})}$ in the integer form via the table lookup operation, and conduct the de-quantization of the multiplication between $\boldsymbol{A}$ and $\boldsymbol{B}$ in a hardware-friendly way:

$$\widehat{\boldsymbol{A}} \cdot \widehat{\boldsymbol{B}} = s \cdot s' \cdot s_{\text{table}} \cdot \left[\left(\widetilde{\boldsymbol{U}}^{(\mathbb{Z})} \boldsymbol{B}^{(\mathbb{Z})} \right) >> \widetilde{\boldsymbol{A}}^{(\mathbb{Z})} \right], \tag{10}$$

where $>>$ denotes the right shift operation, s and s' indicates the scaling factors for $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. The overall computational flowchart of AdaLog is illustrated in Fig. 3(c).

3.3 Extending AdaLog for Post-GELU Layers

As shown in Fig. 4, similar to the post-Softmax layer, the activations of the post-GELU layers also obey the power-law-like distributions. Besides, the post-GELU activations suffer from the following two issues: 1) the data distributions exhibit large variations across distinct layers; 2) the majority of the values are concentrated in the range of $(-0.17, 0]$. As a consequence, AdaLog is inapplicable for the post-GELU layers as it requires non-negative inputs. To deal with the issues above, we leverage bias reparameterization to make AdaLog feasible for the post-GELU layers.

Specifically, the post-GELU linear layer *FC2* in Fig. 2 has the following form:

$$\boldsymbol{Y} = \boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b}, \tag{11}$$

where $\boldsymbol{W} \in \mathbb{R}^{p \times m}$, $\boldsymbol{b} \in \mathbb{R}^{p}$, $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$ denote the weight matrix, the bias, the post-GELU activation and the output, respectively.

Since the majority values in $\boldsymbol{X}$ locate in the range of $(-0.17, 0]$, we reformulate Eq. (11) as below:

$$\boldsymbol{Y} = \boldsymbol{W} \cdot (\boldsymbol{X} + 0.17 \cdot \mathbf{1}_{m \times n}) + (\boldsymbol{b} - 0.17 \cdot \boldsymbol{W} \cdot \mathbf{1}_{m}), \tag{12}$$

where $\mathbf{1}_m$ and $\mathbf{1}_{m \times n}$ refer to the m-dimensional vector and $m \times n$ matrix with all ones, respectively.

As $\boldsymbol{X}' = \boldsymbol{X} + 0.17 \cdot \mathbf{1}_{m \times n}$ is non-negative, the proposed AdaLog is thus applicable for quantizing the first term of the linear layer in Eq. (12). Concretely, the quantization and the de-quantization are conducted as below:

$$\boldsymbol{X}'^{(\mathbb{Z})} = \text{clamp}\left(\left\lfloor -\log_b \frac{\boldsymbol{X}'}{s} \right\rceil, 0, 2^{bit} - 1\right), \tag{13}$$

$$\widehat{\boldsymbol{X}}' = s \cdot b^{-\boldsymbol{X}'^{(\mathbb{Z})}} \approx \boldsymbol{X} + 0.17 \cdot \mathbf{1}_{m \times n}. \tag{14}$$

In regards to the second term in Eq. (12), in order to keep consistent with the de-quantization on the weight $\widehat{\boldsymbol{W}}$, we employ the following bias reparameterization w.r.t. $\boldsymbol{b}$:

$$\boldsymbol{b}_{rep} = \boldsymbol{b} - 0.17 \cdot \widehat{\boldsymbol{W}} \cdot \mathbf{1}_{m}, \tag{15}$$

where $\widehat{\boldsymbol{W}}$ denotes the de-quantized weight and $\boldsymbol{b}_{rep}$ denotes the reparameterized bias of *FC2*.

3.4 Fast Progressive Combining Search

Both the asymmetrically uniform quantizer and the proposed AdaLog quantizer have two types of hyperparameters. To determine these hyperparameters, existing works adopt the calibration step by using the brute-force search [17, 31] or the alternating search [6, 35]. They require discretizing the continuous hyperparameter space through the uniform grid division. After discretization, the brute-force search traverses all possible combinations of the hyperparameters, while the alternating search iteratively fixes one hyperparameter and searches for the other. However, the complexity of brute-force search is $O(nm)$, where n and m are the number of candidates for the two hyperparameters. By contrast, the alternating search achieves a complexity of $O(n+m)$, but is prone to falling into a local minimum, resulting in a degradation of accuracy.

In this paper, we aim to leverage both the advantages of the above methods, by developing a hyperparameter search algorithm with a linear complexity that can finely partition the search space. Concretely, motivated by the beam search in NLP [27], we develop the Fast Progressive Combining Search (FPCS) strategy. Without loss of generality, we describe FPCS based on the asymmetrically uniform quantizer in the rest part.

Algorithm 1 Fast Progressive Combing Searching.

Input: Coefficients x, y, z_1, z_2, k, p; a pretrained full-precision model; a set of calibration data $\mathcal{D}_{calib}$; and the l-th layer to be quantized ϕ_l.
Output: Quantization hyperparameters a^*, b^*.
The initialization step:
Generate the raw input $\boldsymbol{X}_l$ and output $\boldsymbol{O}_l$ by ϕ_l based on $\mathcal{D}_{calib}$, and compute the percentiles pct_0, $pct_{0.1}$, $pct_{0.9}$ and pct_1 by [14].
Compute the uniform partition of the first and second hyperparameters as $\mathcal{A} = \{pct_{0.1} + i \cdot \tau_A | i = 0, \cdots, x\}$ and $\mathcal{B} = \{pct_{0.9} + j \cdot \tau_B | j = 0, \cdots, y\}$ with the intervals $\tau_A = (pct_0 - pct_{0.1})/x$ and $\tau_B = (pct_1 - pct_{0.9})/y$.
Generate the candidate set $\mathcal{C}_0$ as the Cartesian product of $\mathcal{A}$ and $\mathcal{B}$: $\mathcal{C}_0 = \mathcal{A} \times \mathcal{B}$.
# The progressive searching step:
for $i = 0, \cdots, p$ **do**
# The coarse searching step:
    Construct the subset $\mathcal{C}' \subset \mathcal{C}_i$ by selecting the partitions that have the top-k smallest quantization loss.
The expanding step:
    Update the intervals for fine partitions: $\tau_A := \tau_A/(2 \cdot z_1)$, $\tau_B := \tau_B/(2 \cdot z_2)$.
    Update the candidate set with fine partitions: $\mathcal{C}_{i+1} = \{(a + i \cdot \tau_A, b + j \cdot \tau_B) | (a, b) \in \mathcal{C}'; i = -z_1, \cdots, z_1; j = -z_2, \cdots, z_2\}$.
end for
The optimal hyperparameter $(a^*, b^*) \in \mathcal{C}_p$ is the one that has the smallest quantization loss.

(1) Initialization. Assuming that the desired search complexity is $O(n)$, we generate x candidates for the first hyperparameter and y candidates for the second one by ensuring that $xy = n$. When quantizing the l-th layer ϕ_l, we first utilize the calibration set $\mathcal{D}_{calib}$ to obtain the raw input $\boldsymbol{X}_l$ and the output $\boldsymbol{O}_l$ by ϕ_l. We then calculate the 0.1'th and 0.9'th percentiles denoted by $pct_{0.1}$ and $pct_{0.9}$ via the Percentile method [14]. We employ a uniform partitioning scheme to derive the initial candidate set $\mathcal{C}_0$ for the hyperparameter search.

(2) Coarse searching. Given the candidates $\mathcal{C}_0$, we compute the quantization loss for each candidate $(a, b) \in \mathcal{C}_0$, *i.e.* the MSE loss as between the quantized output and the full-precision output denoted as $\text{MSE}(\phi(l)(\boldsymbol{X}_l, a, b), \boldsymbol{O}_l)$, and build the subset $\mathcal{C}' \subset \mathcal{C}_0$ by selecting the ones with the top-k smallest losses.

(3) Expanding. For each candidate $(a, b) \in \mathcal{C}'$, we expand it to z candidates with fine-grained partitions, via extending a to z_1 candidates and b to z_2 candidates, by ensuring that $z_1 z_2 = z$ and $kz = n$. The expanded candidates form the updated candidate set for searching.

(4) Progressive searching. We iteratively repeat the above two steps until reaching the maximal step p. In the last iteration, we choose the one that has the smallest quantization loss as the optimal hyperparameter (a^*, b^*).

The overall pipeline of the fast progressive combining search strategy is summarized in Algorithm. 1.

4 Experimental Results and Analysis

In this section, we evaluate the effectiveness of our method by comparing to the state-of-the-art post-training quantization approaches for vision transformers on the image classification task, as well as extensively conducting ablation studies and efficiency analysis. For more results on the object detection and instance segmentation tasks on COCO [19], please refer to the *Supplementary Material*.

4.1 Experimental Setup

Datasets and Models. By following [6, 18, 35], for the classification task, we evaluate our method on ImageNet [25] with representative vision transformer architectures, including ViT [7], DeiT [28] and Swin [21].
Implementation Details. In order to make fair comparisons, we adopt the same calibration strategy as depicted in [6,18,35]. Concretely, we randomly select 32 unlabeled images from ImageNet for the classification task. As for weights, we employ the channel-wise quantization. As for activations, we utilize layer-wise quantization in conjunction with the scale reparameterization technique. The AdaLog quantizer is used in all the post-Softmax and post-GELU activations. We set $r = 37$ and search for the best q in Eq. (8) by Algorithm. 1. It is worth noting that we suggest r to be a prime number such that q and r are coprime, since the value of U is desired to vary when $\boldsymbol{A}^{(\mathbb{Z})}$ takes different values. The hyperparameter n that controls the searching complexity and the searching step p in FPCS are fixed to 128 and 4, respectively.

4.2 Comparison with the State-of-the-Art Approaches

We firstly compare our method with the state-of-the-art post-training quantization approaches for vision transformers on ImageNet, including PTQ4ViT [35], APQ-ViT [6] and RepQ-ViT [18]. We report the results under the 6, 4, and 3 bit-widths with distinct vision transformer architectures.

As summarized in Table 1, for 6-bit quantization, many approaches such as PTQ4ViT and APQ-ViT exhibit a clear decrease in accuracy. In contrast, our method still delivers a promising performance, reaching the highest accuracy among the compared approaches with various backbone models. In regards to 4-bit quantization, all the compared methods suffer a remarkable degradation in accuracy due to severe quantization loss of weights and activations. However, the performance of the proposed AdaLog remains competitive in comparison with the full-precision models. Meanwhile, AdaLog significantly outperforms the compared approaches, achieving an average improvement of 5.13% over the second best method, *i.e.* RepQ-ViT. In addition, we evaluate on the more challenging 3-bit quantization. As displayed in Table 1, RepQ-ViT and PTQ4VIT fail to properly deal with the quantization on the post-GELU and post-Softmax activations, thus yielding extremely low performance (*e.g.* 0.1%) in most scenarios. By contrast, AdaLog reaches more reasonable accuracies when reducing the bit-width from 32 to 3.

Table 1: Comparison of the top-1 accuracy (%) on the ImageNet dataset with different quantization bit-width. '-' implies that the result is not reported or not available.

Model	Full Prec.	Method	W3/A3	W4/A4	W6/A6
ViT-S/224	81.39	PTQ4ViT	0.10	42.57	78.63
		APQ-ViT	-	47.95	79.10
		RepQ-ViT	0.10	65.05	80.43
		AdaLog (Ours)	**13.88**	**72.75**	**80.91**
ViT-B/224	84.54	PTQ4ViT	0.10	30.69	81.65
		APQ-ViT	-	41.41	82.21
		RepQ-ViT	0.10	68.48	83.62
		AdaLog (Ours)	**37.91**	**79.68**	**84.80**
DeiT-T/224	72.21	PTQ4ViT	3.50	36.96	69.68
		APQ-ViT	-	47.94	70.49
		RepQ-ViT	0.10	57.43	70.76
		AdaLog (Ours)	**31.56**	**63.52**	**71.38**
DeiT-S/224	79.85	PTQ4ViT	0.10	34.08	76.28
		APQ-ViT	-	43.55	77.76
		RepQ-ViT	0.10	69.03	78.90
		AdaLog (Ours)	**24.47**	**72.06**	**79.39**
DeiT-B/224	81.80	PTQ4ViT	31.06	64.39	80.25
		APQ-ViT	-	67.48	80.42
		RepQ-ViT	0.10	75.61	81.27
		AdaLog (Ours)	**57.45**	**78.03**	**81.55**
Swin-S/224	83.23	PTQ4ViT	28.69	76.09	82.38
		APQ-ViT	-	77.15	82.67
		RepQ-ViT	0.10	79.45	82.79
		AdaLog (Ours)	**64.41**	**80.77**	**83.19**
Swin-B/224	85.27	PTQ4ViT	20.13	74.02	84.01
		APQ-ViT	-	76.48	84.18
		RepQ-ViT	0.10	78.32	84.57
		AdaLog (Ours)	**69.75**	**82.47**	**85.09**

4.3 Ablation Study

Effect of the Main Components. We first evaluate the effectiveness of the proposed AdaLog quantizer and the FPCS strategy. As summarized in Table 2, by applying AdaLog to the post-GELU and post-Softmax activation quantization, the top-1 accuracy is significantly promoted for distinct vision transformer architectures and different bit-widths. For instance, AdaLog improves the accuracy by 9.81%, and 4.86% when quantizing ViT-S and DeiT-T on W4/A4, respectively. The proposed FPCS also consistently boosts the accuracy, obtaining 5.82% and 5.02% performance gains when quantizing ViT-B and Swin-B on W3/A3, respectively. A combination of them further promotes the performance.

Table 2: Ablation results w.r.t the top-1 accuracy (%) of the proposed main components on ImageNet with the W4/A4 and W3/A3 settings.

AdaLog	FPCS	**ViT-S** (81.39)		**DeiT-T** (72.21)		**Swin-S** (81.80)	
		W3/A3	W4/A4	W3/A3	W4/A4	W3/A3	W4/A4
		3.51	62.20	22.73	58.01	44.65	78.40
✓		11.40	72.01	28.41	62.87	61.50	80.46
	✓	3.77	63.14	24.80	59.93	44.61	78.79
✓	✓	**13.88**	**72.75**	**31.56**	**63.52**	**64.41**	**80.77**
AdaLog	**FPCS**	**ViT-B** (84.54)		**DeiT-S** (79.85)		**Swin-B** (85.27)	
		W3/A3	W4/A4	W3/A3	W4/A4	W3/A3	W4/A4
		9.68	76.49	22.49	69.04	47.18	80.33
✓		28.80	79.19	22.81	71.64	68.97	82.10
	✓	15.50	78.08	23.55	69.23	52.20	80.67
✓	✓	**37.91**	**79.68**	**24.47**	**72.06**	**69.75**	**82.47**

Table 3: Comparison of FixOPs/Model size under different bits.

Model	Bits	Method	Prec.	FixOPs	Model Size
DeiT-T FixOPs: 20.1B Size: 21.9MB	4/4	RepQ-ViT	57.43	0.613B	3.4MB
	4/4	AdaLog	**63.52**	**0.539B**	3.4MB
	3/3	RepQ-ViT	0.10	0.444B	2.7MB
	3/3	AdaLog	**31.56**	**0.391B**	2.7MB

On the Efficiency and Effectiveness of AdaLog. To display the efficiency of AdaLog, we compare to RepQ-ViT in terms of the overall model size. Additionally, we report the number of FixOP [16], *i.e.* one operation between an 8-bit weight and an 8-bit activation, as the evaluation metric for the inference cost. Since AdaLog completely avoids floating-point operations by quantizing the lookup table, it is more efficient than the $\log\sqrt{2}$ quantizer which requires floating-point operations during inference. Table 3 clearly shows that AdaLog is more efficient than RepQ-ViT with almost the same model size. It is worth noting that AdaLog utilizes four lookup operations in each layer, and the length of each table is 2^{bit}, thus taking negligible memory cost. For instance, in 4-bit quantization on DeiT-T with 12 layers, the lookup tables only take about 3KB memory, which is less than 0.2% of the overall quantized model size.

To further demonstrate the effectiveness of AdaLog, we implement BRECQ [17] on ViT-Small by using 1024 calibration images. The results show that when using the AdaLog quantizer, BRECQ significantly benefits from training activation parameters with LSQ [8]. However, without the AdaLog quantizer, training activation parameters may incur a collapse of accuracy. This indicates that the

Table 4: Quantization results on ImageNet. 'Optim.' refers to using the BRECQ [17] optimizing strategy. 'Train Act.' indicates training the scaling factor of activations by applying LSQ [8], besides the defaulted optimization on the rounding parameters in AdaRound [23].

Model	AdaLog	Imgs	Optim.	Train Act.	W3/A3	W4/A4
ViT-S/224 81.39	×	32	×	-	3.77	63.14
	×	1024	✓	×	28.19	69.21
	×	1024	✓	✓	0.93	1.93
	✓	32	×	-	13.88	72.75
	✓	1024	✓	×	37.18	76.48
	✓	1024	✓	✓	**62.50**	**77.25**

Table 5: Comparison of the top-1 accuracy and time consumption on a single NVIDIA RTX 4090 GPU during the hyperparameter search process in quantization.

Model	**Method**	**Top-1 Acc. (%)**	**Complexity**	**GPU Min.**
DeiT-T/224 (W3A3)	Alternating [35]	28.41	$O(n)$	3.3
	Brute Force [31]	32.04	$O(n^2)$	183
	FPCS (Ours)	31.56	$O(pn)$	4.1
DeiT-S/224 (W3A3)	Alternating [35]	22.17	$O(n)$	5.7
	Brute Force [31]	29.38	$O(n^2)$	312
	FPCS (Ours)	28.51	$O(pn)$	6.5

AdaLog quantizer can be integrated into existing PTQ frameworks, facilitating stabilizing the activation training process under extremely low bit-width.

On the Efficiency of FPCS. We further validate the efficiency of FPCS by comparing to the Alternating search strategy [35] and the Brute Force search strategy [31]. As shown in Tab. 5, due to the progressive search space partitioning with linear complexity, FPCS reaches a high accuracy as the Brute Force search, while taking extremely less time cost as the Alternating search.

5 Conclusion

In this paper, we propose a novel approach for post-training quantization of vision transformers. We first develop a non-uniform quantizer dubbed AdaLog that is capable of adaptively selecting the logarithm base, and is simultaneously hardware-friendly during inference. By employing the bias reparameterization, AdaLog is applicable to quantize both the post-Softmax and the post-GELU activations, and significantly promote the performance. Moreover, we propose a Fast Progressive Combining Search strategy to improve the successive hyperparameter searching. Extensive experimental results show the efficiency and effectiveness of our approach for distinct ViT-based architectures.

Acknowledgments

This work was partly supported by the National Key R&D Program of China (2021ZD0110503), the National Natural Science Foundation of China (Nos. 62202 034, 62176012, 62022011), the Beijing Natural Science Foundation (No. 4242044), the Beijing Municipal Science and Technology Project (No. Z231100010323002), the Research Program of State Key Laboratory of Virtual Reality Technology and Systems, and the Fundamental Research Funds for the Central Universities.

References

1. Bolya, D., Fu, C., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. In: ICLR (2023)
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. pp. 213–229 (2020)
3. Chen, C.R., Fan, Q., Panda, R.: Crossvit: Cross-attention multi-scale vision transformer for image classification. In: ICCV. pp. 347–356 (2021)
4. Choi, J., Wang, Z., Venkataramani, S., Pierce, I., Chuang, J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085 (2018)
5. Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., Zhang, L.: Dynamic DETR: end-to-end object detection with dynamic attention. In: ICCV. pp. 2968–2977 (2021)
6. Ding, Y., Qin, H., Yan, Q., Chai, Z., Liu, J., Wei, X., Liu, X.: Towards accurate post-training quantization for vision transformer. In: ACM MM. pp. 5380–5388 (2022)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
8. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. In: ICLR (2020)
9. Frumkin, N., Gope, D., Marculescu, D.: Jumping through local minima: Quantization in the loss landscape of vision transformers. In: ICCV. pp. 16978–16988 (2023)
10. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision. pp. 291–326 (2022)
11. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M.: Levit: a vision transformer in convnet's clothing for faster inference. In: ICCV. pp. 12239–12249 (2021)
12. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342 (2018)
13. Li, B., Chen, J., Bao, X., Huang, D.: Compressed video prompt tuning. In: NeurIPS (2023)
14. Li, R., Wang, Y., Liang, F., Qin, H., Yan, J., Fan, R.: Fully quantized network for object detection. In: CVPR. pp. 2810–2819 (2019)
15. Li, Y., Xu, S., Zhang, B., Cao, X., Gao, P., Guo, G.: Q-vit: Accurate and fully quantized low-bit vision transformer. In: NeurIPS (2022)

16. Li, Y., Dong, X., Wang, W.: Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In: ICLR (2020)
17. Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., Gu, S.: BRECQ: pushing the limit of post-training quantization by block reconstruction. In: ICLR (2021)
18. Li, Z., Xiao, J., Yang, L., Gu, Q.: Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In: ICCV. pp. 17227–17236 (2023)
19. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV. pp. 740–755 (2014)
20. Lin, Y., Zhang, T., Sun, P., Li, Z., Zhou, S.: Fq-vit: Post-training quantization for fully quantized vision transformer. In: IJCAI. pp. 1173–1179 (2022)
21. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 9992–10002 (2021)
22. Mehta, S., Rastegari, M.: Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In: ICLR (2022)
23. Nagel, M., Amjad, R.A., van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. In: ICML. pp. 7197–7206 (2020)
24. Rokh, B., Azarpeyvand, A., Khanteymoori, A.: A comprehensive survey on model quantization for deep neural networks. ACM Transactions on Intelligent Systems and Technology **14**(6), 1–50 (2023)
25. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
26. Strudel, R., Pinel, R.G., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV. pp. 7242–7252 (2021)
27. Tillmann, C., Ney, H.: Word reordering and a dynamic programming beam search algorithm for statistical machine translation. Computational Linguistics **29**(1), 97–133 (2003)
28. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. vol. 139, pp. 10347–10357 (2021)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS. pp. 5998–6008 (2017)
30. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: HAQ: hardware-aware automated quantization with mixed precision. In: CVPR. pp. 8612–8620 (2019)
31. Wei, X., Gong, R., Li, Y., Liu, X., Yu, F.: Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. In: ICLR (2022)
32. Wu, D., Tang, Q., Zhao, Y., Zhang, M., Fu, Y., Zhang, D.: Easyquant: Post-training quantization via scale optimization. arXiv preprint arXiv:2006.16669 (2020)
33. Xu, Z., Zhang, W., Zhang, T., Yang, Z., Li, J.: Efficient transformer for remote sensing image segmentation. Remote. Sens. **13**(18), 3585 (2021)
34. Yu, L., Xiang, W.: X-pruner: explainable pruning for vision transformers. In: CVPR. pp. 24355–24363 (2023)
35. Yuan, Z., Xue, C., Chen, Y., Wu, Q., Sun, G.: Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In: ECCV. pp. 191–207 (2022)
36. Zhang, Y., Chen, J., Huang, D.: Cat-det: Contrastively augmented transformer for multi-modal 3d object detection. In: CVPR. pp. 908–917 (2022)

37. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H.S., Zhang, L.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR. pp. 6881–6890 (2021)
38. Zhou, C., Zhang, Y., Chen, J., Huang, D.: Octr: Octree-based transformer for 3d object detection. In: CVPR. pp. 5166–5175 (2023)
39. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: deformable transformers for end-to-end object detection. In: ICLR (2021)