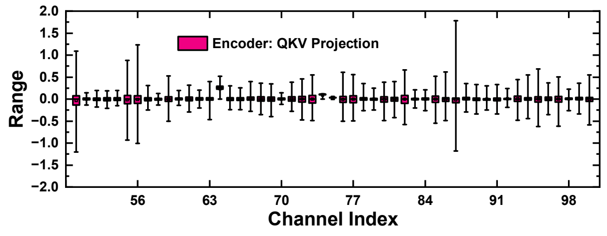
AHCPTQ: Accurate and Hardware-Compatible Post-Training Quantization for Segment Anything Model

Supplementary Material

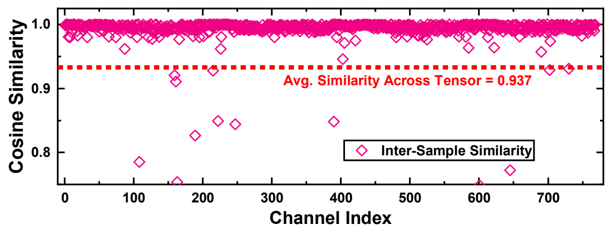
A. Analysis of Inter-Channel Variation and Inter-Sample Similarity in SAM Model

In this section, we provide an in-depth analysis of inter-channel variation and inter-sample similarity using the SAM-B model with YOLOX as the prompt detector. We extract the quantization ranges for channel indices 50 to 100 to examine channel-wise variation across multiple layers where CAG is applied. Additionally, we determine the optimal scale and zero point for each channel using different samples to assess parameter consistency across 100 samples.

In the image encoder, Figs. 7a and 8a show that QKV projection and Linear-1 activations in the MLP block exhibit channel-wise variation due to LayerNorm operations.

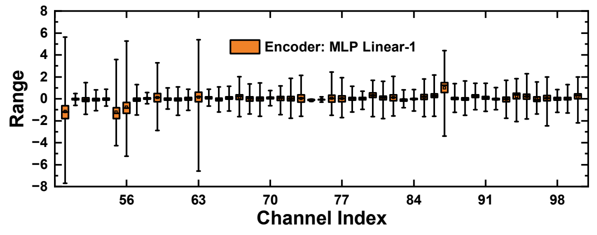


(a) Range distribution for channel indices 50 to 100.

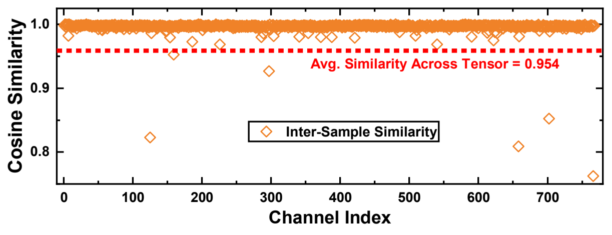


(b) Cosine similarity across 100 samples for each channel.

Figure 7. Range distribution and cosine similarity of QKV projection activations in the image encoder.



(a) Range distribution for channel indices 50 to 100.

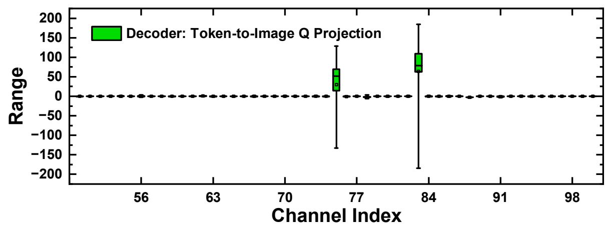


(b) Cosine similarity across 100 samples for each channel.

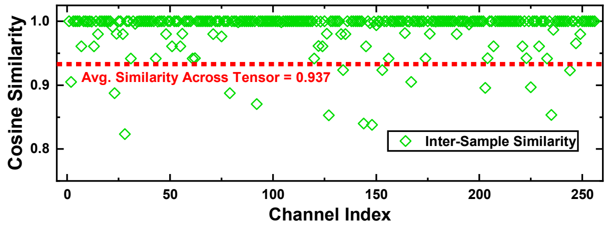
Figure 8. Range distribution and cosine similarity of Linear-1 activations in the MLP block of the image encoder.

In the mask decoder, channel-wise variation in certain layers poses a major challenge for low-bit quantization. In Token-to-Image cross-attention, linear projection activations in the Query and Value projections serve as the primary challenges, as shown in Fig. Figs. 1b and 9a. Notably, activation outliers in the Query projection extend the quantization range to approximately 400, severely increasing quantization error under per-tensor quantization.

In Image-to-Token cross-attention, linear projection activations in the Key and Value projections exhibit distribution variations that degrade SAM's performance. As shown in Figs. 10a and 11a, Key activations remain relatively stable, whereas Value activations demonstrate higher variance.



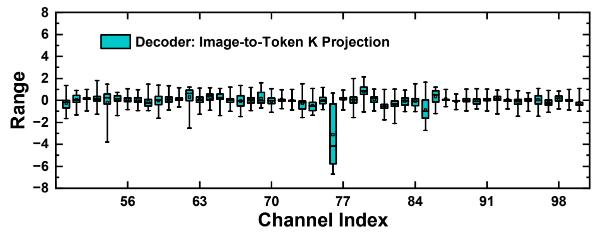
(a) Range distribution for channel indices 50 to 100.



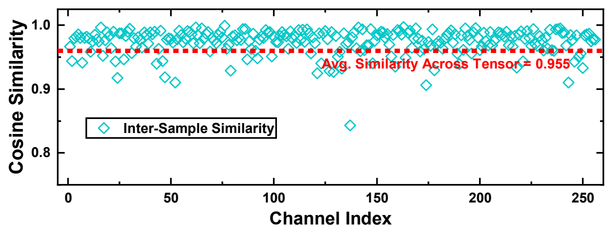
(b) Cosine similarity across 100 samples for each channel.

Figure 9. Range distribution and cosine similarity of pre-projection activations for Token-to-Image Query in the mask decoder.

Lastly, the range distribution of Linear-1 activations in the MLP block of the mask decoder is shown in Fig. 12a.

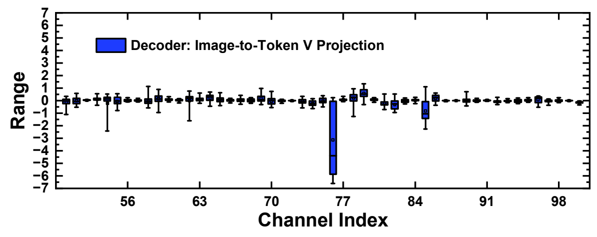


(a) Range distribution for channel indices 50 to 100.

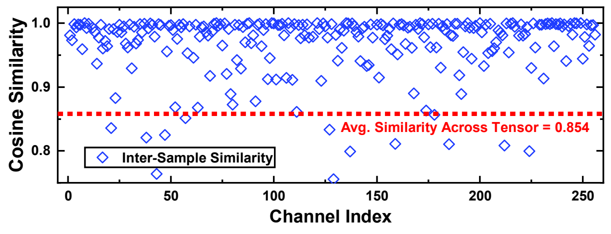


(b) Cosine similarity across 100 samples for each channel.

Figure 10. Range distribution and cosine similarity of pre-projection activations for Image-to-Token Key in the mask decoder.



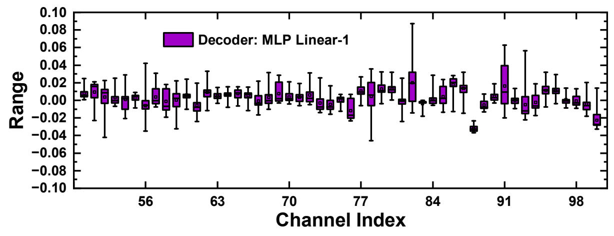
(a) Range distribution for channel indices 50 to 100.



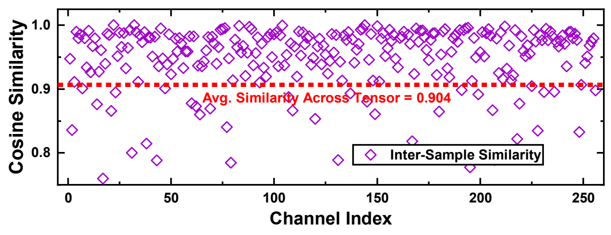
(b) Cosine similarity across 100 samples for each channel.

Figure 11. Range distribution and cosine similarity of pre-projection activations for Image-to-Token Value in the mask decoder.

Fortunately, we verified that in all layers exhibiting severe channel-wise variation, the optimal quantization parameters remain stable across different samples. As shown in Figs. 3, 7b, 8b, 9b, 10b, 11b and 12b, the normalized cosine similarity scores remain consistently high across most channels. Therefore, the CAG-based grouping strategy proves to be a highly reliable approach for enhancing SAM's quantization performance.



(a) Range distribution for channel indices 50 to 100.



(b) Cosine similarity across 100 samples for each channel.

Figure 12. Range distribution and cosine similarity of Linear-1 activations in the MLP block of the mask decoder.

B. Hardware Cost Benchmark Across Different Quantization Granularity

To benchmark the hardware cost associated with different quantization granularities, we analyze the implementation overhead of the SAM-H model under three configurations: per-tensor quantization, per-channel quantization, and CAG in AHCPTQ. Deep learning accelerators typically handle quantization parameters in two ways:

(1) Storing scales and zero points in on-chip registers. This approach completely eliminates data transfer between the off-chip DRAM and the accelerator, ensuring zero transmission latency and cost. However, it increases area and resource overhead due to the large number of registers required.

(2) Storing scales and zero points in off-chip DRAM. This method reduces the on-chip resource overhead but incurs substantial energy and latency costs due to frequent DRAM accesses [10].

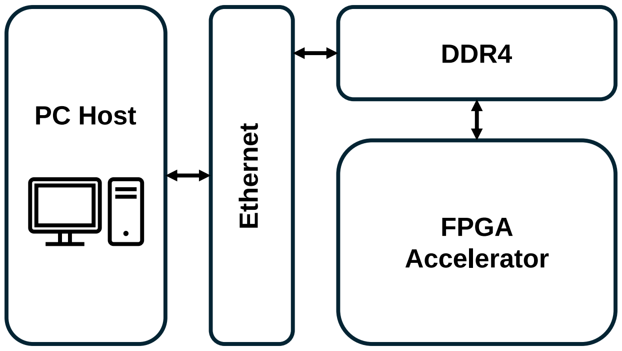
As illustrated in Fig. 4, any combination between these two approaches follows a trade-off curve—reducing one overhead inevitably increases the other. To mitigate this trade-off, we propose CAG, which clusters quantization parameters to significantly reduce their count. By grouping scale and zero-point values into just four clusters, CAG reduces the required registers in **(1)** by 99.7% or the data transfer cost in **(2)** by 99.7%. This approach enables hardware efficiency close to per-tensor quantization while maintaining accuracy comparable to per-channel quantization. Since only 144 registers are needed to store four groups of scale and zero points in 4-bit quantization, we opt to store them on-chip, ensuring a more efficient and simplified hardware design.

C. Hardware Implementation Details

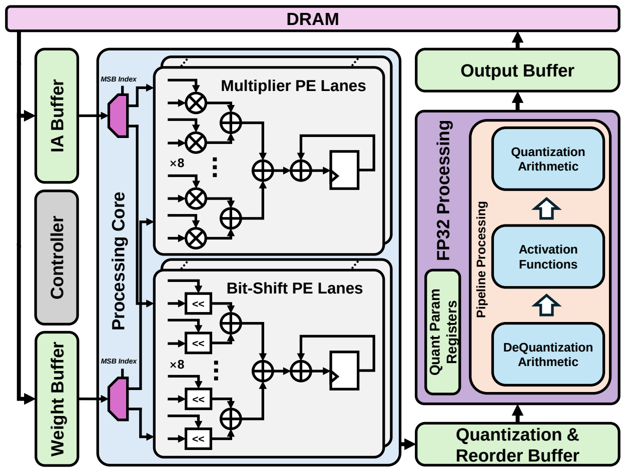
Our AHCPTQ approach is implemented on an FPGA, with a custom accelerator designed to validate the quantization strategy. Fig. 13a depicts the validation system, which includes a host PC for data transfer via Ethernet, DDR4 DRAM for data buffering, and a Xilinx Artix UltraScale+ FPGA configured with the accelerator. The general architecture of the accelerator, shown in Fig. 13b, incorporates four groups of buffers that leverage on-chip BRAM resources. These buffers temporarily store input activations, model weights, and output activations before and after quantization. The accelerator supports two configurations of PEs: **(1)** an 8-input integer multiplier-and-adder PE for uniform quantization, where both inputs and outputs are integers and partial sums are updated using an accumulator, and **(2)** an 8-input decimal bit-shift-and-adder PE for log2 quantization, with integer inputs, decimal outputs, and a decimal accumulator for partial sums. In our 4-bit AHCPTQ implementation, the accelerator uses 64 lanes for multiplier and bit-shift PEs. To enable HLUQ, weights and activations must be dynamically allocated to their corresponding PEs. This allocation logic is embedded in the controller, which indexes the MSBs of activations. When data is read from the IA buffer, weights and activations are automatically distributed to the corresponding PE types.

After completing the integer and decimal inner product operations, the output values are transferred to the quantization processor in the FP32 domain. A small set of quantization parameter registers loads the scales required for the current and next layers, switching addresses to provide these scales to the quantization processor. The output activations are then fed into the dequantization unit, where, if HLUQ is applied, the uniform and log2 branches are merged. Subsequently, the activations pass through the activation functions and quantization units, ensuring the resulting integer values can be directly used by the next layer. Finally, the activations are sent to the output buffer. The dequantization, activation function, and quantization processes are designed in a pipeline to minimize latency, which is critical for HLUQ deployment in practical applications.

Weights are grouped offline and reordered based on group indices, while activations are reordered on-the-fly. At the start of layer computation, the accelerator accesses the DRAM to transfer weights and activations to the weight and IA buffers. Simultaneously, the quantization processor loads the quantization parameters required for quantization and dequantization. The controller then dispatches paired weights and activations to the appropriate PE lanes based on the MSB of the activations. For instance, with $\beta = \frac{1}{2}$ as described in Sec. 3.3.1, activations with an MSB of '1' are routed to multiplier PE lanes, while those with an MSB of '0' are sent to bit-shift PE lanes. Once computation is completed, the results stored in the accumulators are clustered into two categories and passed to their respective uniform or log2 dequantization units. The dequantized FP32 activations are then written back to the quantization buffer using a tailored addressing logic to restore their default sequence. Following this, the activations undergo the activation function and quantization for the next layer before being sent to the output buffer and ultimately recycled back to DRAM. To streamline the process, the reorder logic is fused into the DRAM controller, which writes activations to grouped addresses in DRAM when CAG is applied to the next layer.



(a) Evaluation system.



(b) Accelerator architecture.

Figure 13. Overview of the evaluation system and the accelerator architecture for the AHCPTQ configuration.

We implemented the RTL of the accelerator, Ethernet interface, and DRAM controller in Verilog, synthesized the design using Vivado Design Suite, and deployed it on the ALINX AXAU15 development board. As illustrated in Fig. 14, weights and activations are stored in on-board DDR4 DRAM and transferred via Ethernet from the host PC. The accelerator operates at 200 MHz to assess speedup and energy efficiency. For comparison, we implemented two baseline accelerators alongside our AHCPTQ config-

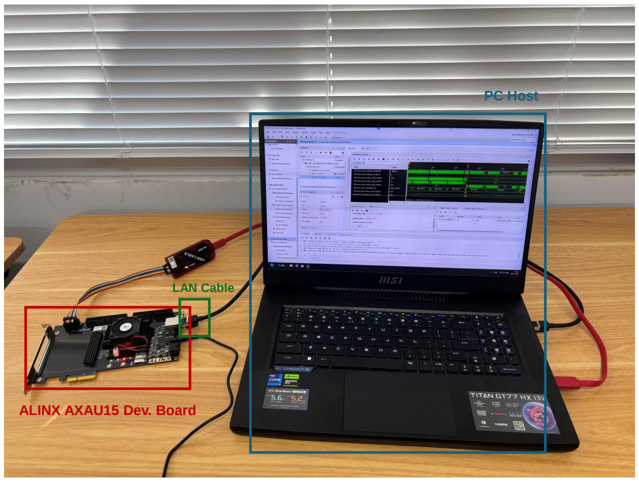


Figure 14. FPGA validation environment.

uration: **(1)** a standard FP32 implementation, **(2)** a default INT8 implementation. In all designs, floating-point operations such as quantization and dequantization are handled by an IP generator utilizing on-chip DSP resources. The detailed experimental results are presented in Sec. 4.4.

D. Experiment on Vision Transformers

To ensure that AHCPTQ generalizes to other vision models and tasks, we evaluate its effectiveness on ImageNet for image classification using DeiT. We integrate AHCPTQ into I&S-ViT [47], the latest state-of-the-art PTQ framework, by replacing its original post-GELU quantizer with HLUQ.

Table 4. Comparison of W4A4 PTQ methods on DeiT based on image classification accuracy on the ImageNet dataset.

Method	Opti.	DeiT-T	DeiT-S	DeiT-B
FQ-ViT [23]	×	0.10	0.10	0.10
PTQ4ViT [41]	×	36.96	34.08	64.39
APQ-ViT [4]	×	47.94	43.55	67.48
BRECQ [17]	✓	55.63	63.73	72.31
QDrop [39]	✓	61.93	68.27	72.60
PD-Quant [24]	✓	62.46	71.21	73.76
RepQ-ViT [21]	×	57.43	69.03	75.61
I&S-ViT [47]	✓	65.21	75.81	79.97
AHCPTQ	✓	**66.11**	**76.12**	**80.07**

In vision transformers (ViTs), inter-channel variation can be effectively addressed by reparameterizing LayerNorm's weight and bias, as LayerNorm consistently precedes the QKV linear projection. Consequently, we follow the settings of RepQ-ViT [21] and I&S-ViT [47], applying reparameterization to reduce inter-channel variance, making CAG unnecessary in this case. However, in SAM's decoder, the LayerNorm placement differs significantly from ViT's image encoder, making efficient reparameterization infeasible. This fundamental difference motivated the introduction of CAG as a dedicated PTQ solution for SAM with high hardware efficiency.

We perform PTQ on DeiT-T, DeiT-S, and DeiT-B, comparing AHCPTQ against static PTQ methods (FQ-ViT [23], PTQ4ViT [41], APQ-ViT [4], RepQ-ViT [21]) and optimization-based PTQ methods (BRECQ [17], QDrop [39], PD-Quant [24], I&S-ViT [47]). As shown in Table 4, AHCPTQ achieves the highest classification accuracy in W4A4 quantization, surpassing all competing methods.

E. Parameter Sensitivity Analysis

To initialize $\hat{b}$, s_1, and s_2 for subsequent optimization in HLUQ, we perform a grid search over two parameters, α and β. We extend the search to $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$ and $\beta \in \{\frac{1}{8}, \frac{1}{4}, \ldots, \frac{7}{8}\}$ on SAM-B with YOLOX. The resulting W4A4/W5A5/W6A6 mAP scores of 13.1/32.0/35.3 align with the reported values as shown in Table 5, indicating that quantization performance is empirically insensitive to these parameters and a limited subset suffices.

Table 5. Parameter sensitivity analysis on SAM-B with YOLOX.

Method	W4A4	W5A6	W6A6
Default	13.4	31.8	35.4
Extended Search	13.1	32.0	35.3

F. Generalizing CAG to Weight Grouping

To ensure a fair comparison with existing baselines, we apply per-channel quantization to model weights. However, CAG can also be extended to weight quantization, potentially broadening its applicability. To evaluate this, we apply CAG with 32 groups to all model weights on SAM-B with YOLOX. The resulting quantization performance is comparable to the per-channel baseline as shown in Table 6, demonstrating CAG's generalizability to model weights.

Table 6. Performance analysis of applying CAG with 32 groups to the model weights of SAM-B using YOLOX.

Method	W4A4	W5A6	W6A6
Default	13.4	31.8	35.4
+ Weight Grouping	12.3	30.8	34.3

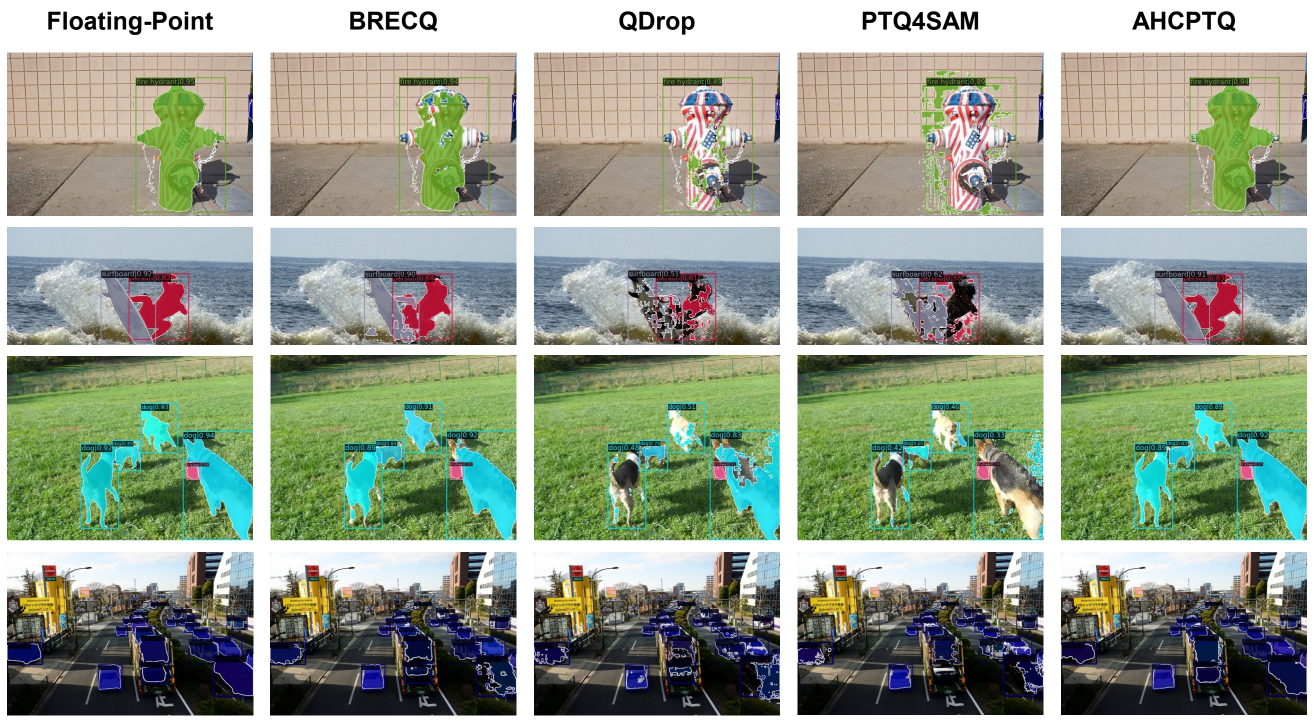


Figure 15. Qualitative comparison of segmentation masks generated by different quantization methods on SAM-B with YOLOX. Our AHCPTQ closely matches the floating-point reference, significantly outperforming other baselines.

G. Comparison of Visualization Results

We further provide visualization results on W4A4 quantization of SAM-H using YOLOX, comparing AHCPTQ with existing quantization methods, including BRECQ [17], QDrop [39], and PTQ4SAM [29], as illustrated in Fig. 15. Qualitatively, our AHCPTQ consistently generates segmentation masks closely resembling those obtained from the original floating-point model, preserving finer structural details and accurate object boundaries. In contrast, the masks produced by other PTQ methods exhibit notable degradations. For instance, QDrop and PTQ4SAM often fail to capture intricate object contours, resulting in coarse segmentation masks and inaccurate boundary delineations. BRECQ, while generally performing better than QDrop and PTQ4SAM, still experiences noticeable detail loss and fragmented segmentation in complex regions. Overall, the visualization outcomes clearly demonstrate that AHCPTQ effectively addresses the quantization challenges inherent in SAM, providing superior segmentation quality comparable to the floating-point baseline, thus confirming its effectiveness and robustness in practical low-bit deployment scenarios.