

# IterQuant: Iterative Quantization Framework for Mixed-Precision LLM Compression

Hyungyo Jeong<sup>1</sup>, Jiwoo Kim<sup>1</sup>, Hyeokjun Kwon<sup>2</sup>, Jaeho Lee<sup>1</sup>, and Youngjoo Lee<sup>3</sup>

<sup>1</sup>Department of Electrical Engineering, POSTECH, Pohang, Republic of Korea

<sup>2</sup>ETRI, Daejeon, Republic of Korea

<sup>3</sup>School of Electrical Engineering, KAIST, Daejeon, Republic of Korea

{hgjung, jiu.0210, kwon36hj, jaeho.lee}@postech.ac.kr, youngjoo@kaist.ac.kr

**Abstract**—Mixed-precision quantization is a promising approach for compressing large language models (LLMs) while maintaining output quality. However, the design space for selecting resolutions of different layers makes exhaustive search intractable. Existing methods either rely on rigid bit-width allocation schemes or require extensive hyperparameter tuning, often based on inaccurate layer-wise sensitivity metrics. In this work, we propose **IterQuant**, an iterative quantization framework that efficiently explores the mixed-precision space without requiring exhaustive enumeration. By incorporating momentum-based scoring to reflect historical performance trends and parameter grouping to balance quantization granularity, **IterQuant** achieves favorable trade-offs between compression and accuracy. Unlike prior approaches that assume bit allocation sensitivity from full-precision models directly transfers to quantized models, **IterQuant** dynamically updates its quantization decisions as the model evolves, better capturing inter-layer dependencies. Experimental results demonstrate that **IterQuant** significantly outperforms state-of-the-art mixed-precision quantization approaches by 2.8% near 4 bits in preserving token-level output quality across various LLM benchmarks.

**Index Terms**—Large language model (LLM), Mixed precision, Post-training quantization (PTQ), Weight-only quantization

## I. INTRODUCTION

Large language models (LLMs) have rapidly increasing parameter counts that substantially raise the memory and computational requirements for inference. The growth has far outpaced hardware improvements: while the parameter size of large transformer models has been increasing at a rate of about  $410\times$  every two years, GPU memory capacity has only grown about  $2\times$  over the same period [1]. The challenge is further exacerbated by the emergence of reasoning-oriented LLMs, such as ChatGPT o1 [2], Claude 3.7 [3], and Qwen3 [4], which rely on chain-of-thought (CoT) reasoning and often generate significantly longer outputs. The combination of expanding model size and longer output sequences substantially amplifies the memory, computation, and latency costs of LLM deployment.

To address these issues, model compression techniques such as pruning [5], distillation [6], and quantization [7]–[10] have been actively explored. Among them, quantization is particularly attractive due to its minimal structural impact and broad

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. RS-2025-09322969, No. RS-2025-02264052, and No. RS-2023-00213710), and the Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00457882).

compatibility with existing inference frameworks [11]. Post-training quantization (PTQ) [12]–[14] has become the default approach as it avoids additional training or datasets, making it more feasible than quantization-aware training (QAT) for large-scale models [15].

Within PTQ, mixed-precision quantization, which assigns different bit-widths to different layers based on their relative sensitivity, has proven effective in balancing compression efficiency and model accuracy [16]–[21].

However, existing mixed-precision PTQ methods rely on coarse layer-wise sensitivity metrics that overlook capture error propagation [18]–[21], restrict precision choices (e.g., binary allocations) [19], or require manual ratio tuning [18], [21]. These issues often result in suboptimal bit-width allocation and limited compression-accuracy trade-offs, motivating the need for more adaptive and output-aware frameworks.

Hence, to fully realize the potential of mixed-precision PTQ, it is crucial to develop an automated, accurate, and broadly applicable framework. To overcome the existing limitations, we propose **IterQuant**, a practical and general mixed-precision PTQ framework that automatically determines effective bit-width assignments without exhaustive search or manual tuning. **IterQuant** consists of three key components:

- 1) A **token-level sensitivity metric** that directly evaluates the effect of quantization on outputs, providing a more reliable basis for bit allocations than layer-wise proxies.
- 2) **Momentum-based scoring** that incorporates results from recent iterations, smoothing sample-level noise and stabilizing the allocation process.
- 3) **Parameter grouping** that balances allocation across layers with imbalanced sizes, avoiding biased selection toward smaller layers while preserving quality.

Our evaluation across multiple LLMs and PTQ backends shows that **IterQuant** consistently improves accuracy while reducing the average bit-width. For instance, on the LLaMA3.2-1B model, **IterQuant** achieves 1.53% higher accuracy than 5-bit uniform quantization, outperforming approaches such as APTQ, PMP, and ShiftAddLLM. In addition, compared to the recent ShiftAddLLM, our solution delivers better energy efficiency on a bit-serial hardware accelerator supporting weight-only quantization [22]. **IterQuant** integrates seamlessly with existing methods such as APTQ, GPTQ and RTN, enabling efficient LLM deployment across diverse real-world scenarios.

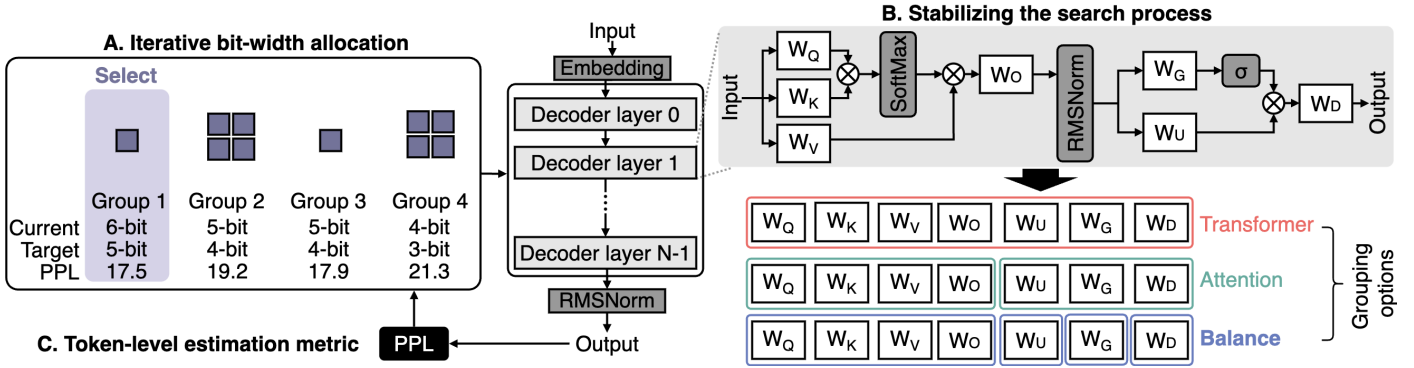


Fig. 1. Overall structure of IterQuant, which selects quantization bit-widths iteratively while preserving output quality.

## II. BACKGROUND

### A. Weight-Only Quantization in LLMs

Quantization refers to representing data at low precision, instead of high-resolution formats such as 32-bit or 16-bit floating points. It is well known that quantizing activations in LLMs is significantly more challenging than quantizing weights, due to the outlier features [23]. Unlike the fixed parameters of pretrained weights, activation values vary with the input, making them less predictable. Several approaches have been proposed to mitigate outlier effects [10], [24], but they typically require modifications to the model architecture or extra computation, incurring additional overhead.

In this context, weight-only schemes have become the de facto choice in many LLM applications [12]–[14], [18], [19]. For example, HAWQ [18] demonstrated that precision allocation guided by the eigenvalue spectrum of the Hessian matrix yields improved accuracy. GPTQ [13] extended this idea by using Hessian information from a calibration set to compute block-wise quantization errors and apply weight corrections. APTQ [19] further adapted the Hessian-based approach to transformers by exploiting information from self-attention blocks, though its scope is limited to specific modules. Moreover, methods such as OWQ [14] and AWQ [7] have targeted low-bit weight-only PTQ, aiming to preserve accuracy while improving robustness under aggressive compression. However, most weight-only quantization methods still rely on nearly uniform bit-width assignments across layers, which limits compression efficiency, naturally motivating mixed-precision quantization approaches.

### B. Previous Mixed-Precision Quantization Methods

Numerous studies have been attempted to assign different bit-widths to each layer based on sensitivity, as the layers contribute unequally to the final output quality. The key challenge is the combinatorial search space: with  $N$  layers and  $K$  bit-width options, the number of possible configurations grows as  $K^N$ , making exhaustive evaluation by perplexity or zero-shot tasks computationally infeasible. Therefore, the success of mixed-precision quantization depends on designing efficient sensitivity metrics that can guide the search toward effective bit-width allocations without testing all possible combinations.

Several approaches rely on Hessian-based sensitivity estimation [12], [13], [18], [19], [21]. HAWQ sorts layers according to Hessian eigenvalues and parameter counts, but it requires manual specification of weight and activation precisions, which makes performance highly dependent on heuristic choices [18]. APTQ [19] and ShiftAddLLM [21] also exploit Hessian trace information, allocating low precision to layers in a predefined order based on a target ratio. APTQ is restricted to binary options and attention blocks, whereas ShiftAddLLM supports multiple precisions but requires advanced ratio tuning. A fundamental limitation of such Hessian-trace approaches is that they assume layer sensitivities are static and independent of previously assigned precisions, leading to inaccurate performance estimation when quantization progresses across the network.

Other methods adopt alternative sensitivity metrics, e.g., PMP from [20] uses the signal-to-quantization-noise ratio (SQNR) by comparing the outputs of partially quantized models against a full-precision baseline. This approach supports multiple bit-width options and accounts for some interactions between layers. However, PMP evaluates sensitivity by quantizing one layer at a time while keeping others at full precision, and thus fails to capture error propagation when multiple layers are quantized together. As compression increases, the resulting allocation diverges from the true behavior of a heavily quantized network, causing the performance degradation.

In summary, existing mixed-precision methods have advanced the field but remain constrained by limited precision choices, reliance on manual ratio tuning, or inaccurate sensitivity estimation. These limitations highlight the need for a more adaptive framework that can iteratively refine bit-width assignments and directly capture their impact on output quality, which is the central motivation for our IterQuant approach.

## III. PROPOSED ITERQUANT FRAMEWORK

### A. Iterative Bit-Width Allocation

As depicted in Fig. 1, the central idea of IterQuant is to perform progressive and adaptive bit-width assignment using an iterative greedy algorithm. Instead of predetermining sensitivity values in the full-precision model, IterQuant evaluates candidate quantization choices directly on the partially quantized model at each step. Specifically, the evolution process of the

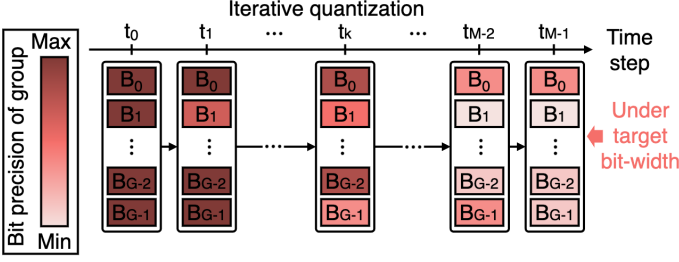


Fig. 2. Concept of iterative quantization, where group precisions are progressively refined across iterations.

### Algorithm 1 Proposed IterQuant Framework

- 1: **Input:** Model weight groups  $\{W_i\}_{i=0}^{G-1}$ , maximum bit-width  $b_{\max}$ , minimum bit-width  $b_{\min}$
- 2: **Output:** Estimation history  $H$  about bit-widths of layers
- 3: Initialize bit-widths:  $B_i \leftarrow b_{\max}$  for  $i = 0, 1, \dots, G - 1$
- 4:  $H \leftarrow \emptyset$
- 5:  $\{W_i\}_{i=0}^{G-1} \leftarrow \text{QUANTIZE}(\{W_i\}_{i=0}^{G-1}, \{B_i\}_{i=0}^{G-1})$
- 6: **while**  $\exists i : B_i > b_{\min}$  **do**
- 7:    $\mathcal{A} \leftarrow \{i : B_i > b_{\min}\}$
- 8:    $i^* \leftarrow -1, \text{ppl}^* \leftarrow \infty$
- 9:   **for**  $i \in \mathcal{A}$  **do**
- 10:      $W_i^{\text{temp}} \leftarrow \text{QUANTIZE}(W_i, B_i - 1)$
- 11:      $\text{ppl} \leftarrow \text{ESTIMATE}(\{W_j\}_{j \neq i} \cup \{W_i^{\text{temp}}\})$
- 12:     **if**  $\text{ppl} < \text{ppl}^*$  **then**
- 13:        $\text{ppl}^* \leftarrow \text{ppl}$
- 14:        $i^* \leftarrow i$
- 15:     **end if**
- 16:   **end for**
- 17:    $B_{i^*} \leftarrow B_{i^*} - 1$
- 18:    $W_{i^*} \leftarrow \text{QUANTIZE}(W_{i^*}, B_{i^*})$
- 19:    $H \leftarrow H \cup \{(B_0, B_1, \dots, B_{G-1})\}$
- 20: **end while**
- 21: **return**  $H$

model is illustrated in Fig. 2. IterQuant progressively increases degree of quantization while reflecting the evolving state of the network, ensuring that quantization decisions are always based on the most up-to-date feedback.

The overall procedure of IterQuant is outlined in Algorithm 1. The algorithm begins with all weight groups initialized at the maximum bit-width. At each step, it tentatively reduces the precision of candidate weight groups ( $W_i$ ) and evaluates the resulting change in model quality. The actual quantization in each iteration is performed through a generic Quantize() function, meaning that IterQuant itself is agnostic to the specific quantization method. In practice, any post-training quantization scheme (e.g., RTN, GPTQ, APTQ) can be plugged into this procedure. The weight group incurring the smallest degradation is then finalized, and the process continues until the minimum bit-width constraint is reached. This iterative strategy relaxes the unrealistic assumption in previous Hessian- or loss-based approaches [18], [19], [21] that layer sensitivities remain fixed throughout quantization.

TABLE I  
PARAMETER DISTRIBUTION (%) FOR GROUPING STRATEGIES

	Transformer (1 group)	Attention (2 groups)	Balance (4 groups)
LLaMA3.2-1B	100	17/83	17/28/28/28
LLaMA2-7B	100	33/67	33/22/22/22
MobileLLM-125M	100	25/75	25/25/25/25

## B. Stabilizing the Search Process

1) *Momentum for Robust Layer Selection:* While greedy iteration provides adaptivity, it is still vulnerable to instability, particularly when calibration sets are limited. A decision based solely on the current estimation may overfit to local token samples, leading to fluctuations or local optima. To mitigate this, IterQuant introduces a momentum mechanism that aggregates recent evaluation results into a moving average. Specifically, the layer score at step  $t$  is defined as follows:

$$M_t = \frac{1}{m} \sum_{k=0}^{m-1} x_{t-k}, \quad (1)$$

where  $x_{t-k}$  denotes the evaluation metric at step  $t - k$ . This smoothing effect reduces variance, dampens the influence of outlier batches and guides the search toward more stable allocation trajectories. In practice, momentum enables IterQuant to consistently select layers that minimize long-term degradation rather than being swayed by short-term noise.

2) *Grouping for Parameter Imbalance:* A further challenge is the uneven distribution of parameters across different layer types. For example, in LLaMA3.2-1B, attention projection layers contain tens of millions of parameters, whereas MLP layers contain several times more. A naïve greedy search tends to push smaller layers, such as attention projections, into low precision early on, since the immediate degradation appears smaller. However, from the perspective of overall compression, quantizing one MLP layer may actually provide a better trade-off than quantizing multiple small self-attention layers. IterQuant introduces layer grouping to ensure decisions across layers of different sizes are evaluated on equal grounds.

We experiment with three strategies: **Transformer grouping**, which treats the entire transformer block as a unit; **Attention grouping**, which separates self-attention and MLP blocks; and **Balance grouping**, which further divides the MLP block into projection sub-layers for a more uniform parameter distribution. The parameter ratios under each strategy are reported in Table I, and Fig. 2 shows how groups evolve at each step. As shown in the table, the balance grouping provides a more even distribution, reducing bias and improving robustness.

## C. Token-Level Estimation Metric

Choosing an appropriate estimation metric is critical for effective bit-width allocation. Previous works typically estimated sensitivity by measuring weight reconstruction error [14], activation error [20], or Hessian traces [18], [19], [21] at the layer level. However, these proxies fail to capture two essential

aspects, namely: 1) quantization errors in one layer propagate to subsequent layers, so layer-wise errors do not directly reflect the model’s final performance (error propagation), and 2) minimizing weight or activation loss does not necessarily translate to preserving generation quality in LLMs (task relevance). To overcome these limitations, IterQuant directly measures perplexity (PPL) on held-out tokens at every step:

$$PPL = \exp\left(-\frac{1}{t} \sum_{i=1}^t \log p(x_i|x_{<i})\right). \quad (2)$$

This token-level estimation aligns with the fundamental goal of LLM quantization, which is to preserve the quality of generated outputs. By directly estimating candidate quantization in terms of PPL, IterQuant ensures that bit-width assignments are optimized for end-task performance rather than misled by imperfect proxy estimations.

Computing perplexity on the full training corpus is prohibitively expensive, so we instead evaluate PPL on a small calibration set, which offers a practical balance between fidelity and efficiency. Naïvely, mixed-precision search requires exploring  $O(K^N)$  configurations, which is computationally infeasible. IterQuant mitigates this by operating at the group level, where  $G$  denotes the number of groups formed from layers (e.g., attention and MLP projections). At each iteration, as shown in Algorithm 1, we evaluate all  $G$  candidates by reducing the bit-width of each group by one. Since decreasing the average precision by one bit requires (empirically) up to  $G$  iterations, the overall complexity with  $K$  quantization options becomes  $O(K \times G^2)$ . As a result, IterQuant significantly relaxes the search cost compared to the naïve approach while still capturing inter-group dependencies. For example, generating a mixed-precision LLaMA3.2-1B model across 6–3 bits was completed within 16 hours on four NVIDIA TITAN XP GPUs.

#### IV. EXPERIMENTAL RESULTS

We evaluate IterQuant on three representative LLMs: MobileLLM-125M [25], LLaMA3.2-1B [26], and LLaMA2-7B [27], using the HuggingFace framework. Calibration is performed with subsets of the C4 [28] and WikiText [29] datasets, where each step samples 256 sequences randomly. Model accuracy is assessed through zero-shot evaluation using the LM Evaluation Harness [30], covering PIQA [31], Winogrande [32], ARC-Easy, ARC-Challenge [33], OBQA [34], and MMLU [35]. For MMLU, more specifically, we evaluate the first 30 samples from each task (1,770 in total), which we found sufficient to capture relative performance trends while keeping runtime practical for large-scale LLM benchmarking.

As a baseline, we consider three PTQ methods: RTN, GPTQ, and APTQ. For RTN and GPTQ, all layers are quantized using the respective method. APTQ, however, was originally proposed only for self-attention blocks and cannot be directly applied to MLP layers. For a fair comparison, we quantize self-attention layers with APTQ while applying GPTQ to the remaining MLP layers, and we report the combined result as the APTQ baseline. Calibration for Hessian-based methods is

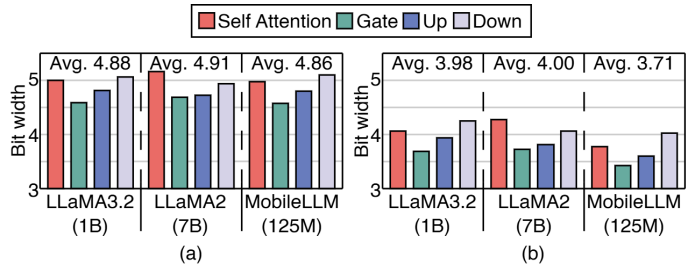


Fig. 3. Bit allocation across groups for target bit-widths of (a)  $< 5$  bits and (b)  $< 4$  bits under the balance grouping strategy.

performed with WikiText sequences. For all models, we additionally adopt a row-wise quantization scheme, which assigns scale factors and zero points per row, ensuring stability under aggressive compression, consistent with prior works [13], [19]. In IterQuant, the Quantize() function in Algorithm 1 can adopt any PTQ method. For evaluation, we instantiated it with APTQ, which provided strong baseline accuracy.

##### A. Bit-Width Allocation Analysis

We evaluate IterQuant across average precisions from 6 to 3 bits, where clear compression-accuracy trade-offs can be observed. Our framework shown in Algorithm 1 starts from 6-bit precision and iteratively reduces less sensitive layers, resulting in an optimized allocation at each step. The effective average bit-width is calculated as a parameter-weighted average:

$$Average\ Bits = \frac{\sum_{i=0}^{G-1} B_i \times P_i}{\sum_{i=0}^{G-1} P_i} \quad (3)$$

Fig. 3 illustrates the group-wise bit allocation. After the IterQuant procedure, self-attention and down-projection layers maintain higher precision, while MLP up- and gate- projection layers are reduced. This pattern shows that these layers are less sensitive, clearly reflecting the asymmetric importance of different layer types. Unlike uniform quantization, which enforces identical precision across layers, IterQuant discovers allocation strategies that balance compression and accuracy.

##### B. Zero-Shot Accuracy

Table II reports detailed zero-shot accuracies of the proposed IterQuant and uniform quantization baselines across multiple LLMs, enabling side-by-side comparison at similar average bit-widths. In addition, Fig. 4 illustrates the accuracy-compression trade-off on LLaMA3.2-1B, where prior mixed-precision approaches using are also included [19]–[21]. As depicted in the figure, existing mixed-precision methods face clear limitations. PMP relies on sensitivity estimated from the full-precision model and fails to capture error propagation, often trailing behind uniform baselines. The mixed-precision extension of APTQ suffers from binary-only allocation and limited sensitivity modeling, causing sharp degradation under low-bit settings. The recent ShiftAddLLM also falls short, since its ratio-based heuristics lack adaptivity and may allocate bits inefficiently across layers, resulting in suboptimal trade-offs.

TABLE II  
ZERO-SHOT ACCURACY (%) WITH VARIOUS QUANTIZATION METHODS

Model	Method	Bits	PIQA	Winogrande	ARC-Easy	ARC-Challenge	OBQA	MMLU	Avg.
LLaMA3.2-1B	full	16	74.48	60.54	65.57	36.35	37.20	40.58	52.45
	APTQ	5	74.37	58.33	63.93	36.01	36.00	33.45	50.35
	GPTQ	5	74.27	59.75	63.72	36.35	36.00	32.63	50.45
	RTN	5	73.18	57.61	61.07	34.73	36.20	34.91	49.62
	IterQuant	4.88	74.27	60.54	64.77	37.20	36.40	38.13	51.88
	APTQ	4	69.97	57.85	59.68	32.08	32.80	24.21	46.10
	GPTQ	4	71.00	56.20	58.29	31.06	33.60	26.73	46.15
	RTN	4	70.84	55.25	57.03	31.48	32.00	26.08	45.45
	IterQuant	3.98	70.62	59.59	58.54	31.91	34.60	31.29	47.76
LLaMA2-7B	full	16	78.02	69.14	76.35	46.25	44.20	43.57	59.59
	APTQ	5	77.80	69.53	75.67	45.48	43.60	41.70	58.96
	GPTQ	5	78.18	69.30	76.18	45.56	43.60	40.58	58.90
	RTN	5	77.80	68.67	76.43	46.59	43.80	41.05	59.06
	IterQuant	4.91	78.02	70.24	76.09	45.73	43.60	41.75	59.24
	APTQ	4	76.82	68.82	74.28	44.88	41.80	36.32	57.15
	GPTQ	4	76.93	67.80	74.45	42.49	41.60	36.78	56.68
	RTN	4	76.71	68.27	73.99	43.34	43.60	36.96	57.15
	IterQuant	4.00	76.88	69.14	74.66	45.22	43.00	36.49	57.57
MobileLLM-125m	full	16	65.13	52.33	46.38	24.32	28.60	23.63	40.06
	APTQ	5	64.42	52.09	46.55	24.32	28.00	23.57	39.82
	GPTQ	5	63.93	52.33	45.54	23.72	27.80	23.74	39.51
	RTN	5	63.76	51.93	45.41	23.98	28.00	23.39	39.41
	IterQuant	4.86	63.93	52.80	46.38	24.32	29.80	23.45	40.11
	APTQ	4	62.62	51.70	42.97	22.78	30.40	23.33	38.97
	GPTQ	4	62.62	50.75	44.40	23.38	29.00	23.57	38.95
	RTN	4	62.57	51.07	42.68	23.12	26.40	23.33	38.19
	IterQuant	3.71	61.64	53.04	43.35	23.98	30.00	23.63	39.27

IterQuant shows higher robustness over different bit ranges. On the LLaMA3.2-1B, for example, IterQuant improves average accuracy by 2.6% over PMP near 5 bits and by 2.8% near 4 bits. Similar improvements over uniform quantization are observed in Table II, where IterQuant matches or even exceeds FP16 performance at sub-5-bit cases. These results highlight that IterQuant provides more effective bit allocation, achieving superior accuracy under aggressive compression.

### C. Hardware Efficiency

We further evaluate IterQuant from a hardware perspective using two representative ML accelerator designs: a conventional FP-INT systolic-array engine and a LUT-based bit-serial engine (FIGLUT) [22]. To match the mixed-precision settings used in this work (6–3 bits), the baseline systolic array supports weights up to INT6 with FP16 activations on a  $64 \times 64$  FP-INT MAC array. While INT6 is the maximum precision, the same engine can execute lower precisions (e.g., INT5 or INT3) without hardware changes, but compute cost remains fixed regardless of allocation. By contrast, FIGLUT processes integer weights with FP activations in a bit-serial fashion, with cycles proportional to the assigned bit-width. For fair comparison with the  $64 \times 64$  systolic baseline, each FIGLUT read-accumulate (RAC) unit is equipped with a  $2 \times 16 \times 4$  LUT array for comparable computing capacity [22]. Both designs are synthesized in 28 nm CMOS

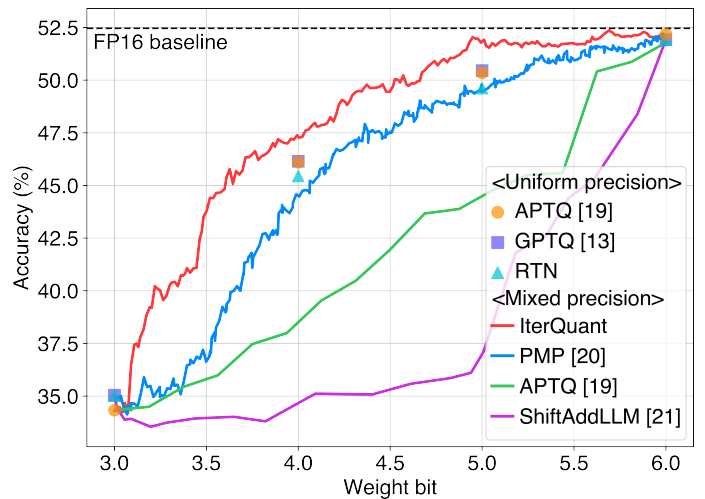


Fig. 4. Accuracy-compression trade-off in LLaMA3.2-1B quantization.

bitat 100 MHz using Synopsys Design Compiler, with on-chip SRAM buffers and CACTI-modeled off-chip DRAM [36].

Fig. 5 compares zero-shot accuracy versus TOPS/W efficiency when different mixed-precision LLaMA3.2-1B models are deployed on systolic array (SA) and FIGLUT accelerators [22]. Mapping to the SA design, reducing average bit-

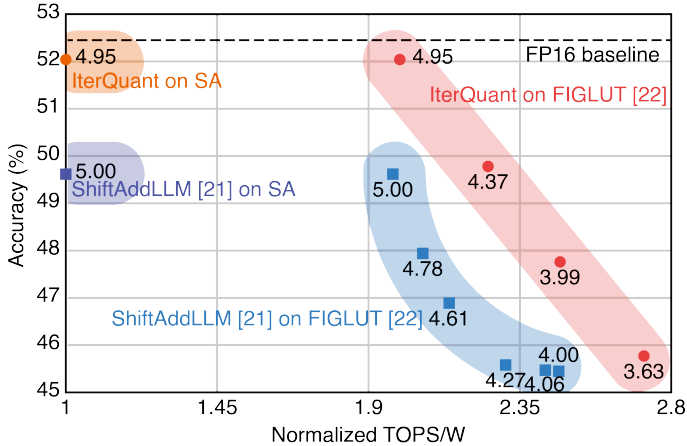


Fig. 5. Zero-shot accuracy versus TOPS/W on FIGLUT and SA accelerators. Numbers denote the average bit-width per configuration.

width does not improve efficiency and only lowers accuracy. Normalized to this SA baseline, however, FIGLUT shows clear advantages, since its inference time scales with bit-width. As the FIGLUT study itself adopted ShiftAddLLM as the reference mixed-precision method to show efficiency benefits, we also use it here as the baseline for hardware-level comparison. On the same hardware, note that replacing the heuristic allocation of ShiftAddLLM with IterQuant leads to better accuracy-efficiency trade-offs. For example, at sub-5-bit precision, IterQuant improves accuracy by up to 2.4% at equal efficiency, or delivers  $1.14\times$  higher TOPS/W at comparable accuracy. Overall, IterQuant not only preserves accuracy under compression but also provides meaningful hardware-level benefits on low-precision bit-serial accelerators.

## V. ABLATION STUDY

We conduct ablation studies on grouping strategy and momentum to validate their contributions to IterQuant framework. All experiments use LLaMA3.2-1B as baseline, and accuracy is reported on the same zero-shot tasks from the main evaluation.

### A. Grouping Strategy

Fig. 6(a) shows the effect of different grouping schemes: *Transformer* (treating an entire block as one unit), *Attention* (separating self-attention and MLP blocks), and *Balance* (further splitting MLP projections into sub-layers). We observe that *Attention* grouping performs worse than *Transformer*, even though it exposes more groups to the search process. This indicates that simply expanding the search space does not guarantee better outcomes. The degradation arises because self-attention layers, which contain relatively fewer parameters, are aggressively pushed to lower precision early on, while large MLP layers remain at higher precision. For instance, at an average precision near 4 bits, most self-attention layers collapse to the minimum bit-width, causing severe accuracy loss. In contrast, *Balance* produces more balanced parameter distributions between groups, reducing this bias and providing the best accuracy among the three strategies.

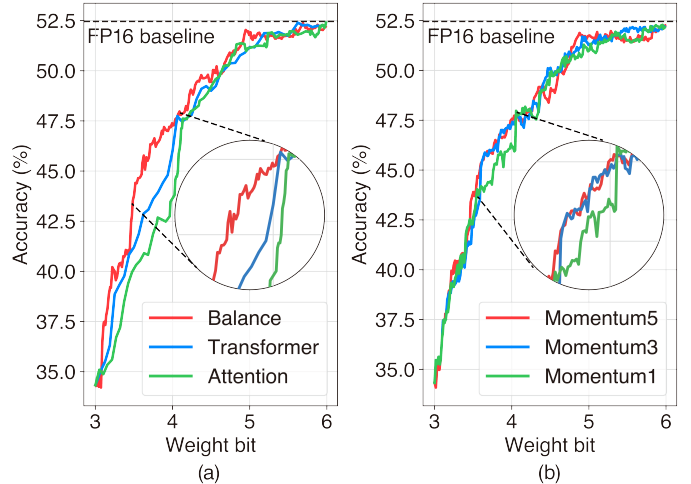


Fig. 6. Zero-shot accuracy of quantized LLaMA3.2-1B models: (a) accuracy for three grouping strategies; (b) accuracy for different momentum values.

### B. Momentum

We next examine the effect of momentum in stabilizing layer selection. Using the *Balance* grouping, we vary momentum values from 1 (pure greedy search) to 5. As shown in Fig. 6(b), small momentum values (1) lead to unstable allocations because decisions rely too heavily on the noisy outcomes of a single calibration subset. As momentum increases, the algorithm gradually averages information from multiple past steps, smoothing fluctuations and improving stability. We find that momentum values in the range of 3–5 consistently improve zero-shot accuracy compared to greedy search, clearly demonstrating that momentum mitigates overfitting to local calibration noise and guide the search toward better long-term solution. Beyond this range, further increases provide only marginal gains, suggesting that moderate momentum achieves the best balance between stability and computational cost. In summary, introducing momentum smooths the search trajectory and prevents premature convergence to suboptimal allocations.

## VI. CONCLUSION

In this paper, we have presented IterQuant, a practical framework for mixed-precision post-training quantization of large language models. IterQuant combines token-level sensitivity estimation, momentum-based scoring, and parameter grouping to systematically reduce average bit-widths while preserving generation quality. Importantly, the framework is agnostic to the underlying quantization method. As a result, it can flexibly incorporate existing PTQ algorithms such as RTN, GPTQ, or APTQ, making it broadly applicable to diverse deployment scenarios. Our experiments on LLaMA and MobileLLM families show that IterQuant consistently outperforms uniform quantization and prior mixed-precision methods, with clear advantages at sub-5-bit precision where accuracy retention is most challenging. IterQuant also offers tangible hardware benefits, as optimized allocations achieve higher energy efficiency and accuracy on a bit-serial accelerator, confirming that algorithmic advances translate effectively to the system level.

## REFERENCES

- [1] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, *et al.*, “Ai and memory wall,” *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.
- [2] OpenAI, “Introducing openai o1,” 2024.
- [3] Anthropic, “Claude 3.7 sonnet,” 2025.
- [4] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, *et al.*, “Qwen3 technical report,” *arXiv preprint arXiv:2505.09388*, 2025.
- [5] X. Ma, G. Fang, and X. Wang, “Llm-pruner: On the structural pruning of large language models,” *Advances in neural information processing systems*, vol. 36, pp. 21702–21720, 2023.
- [6] M. Ji, B. Heo, and S. Park, “Show, attend and distill: Knowledge distillation via attention-based feature matching,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 7945–7952, 2021.
- [7] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, *et al.*, “Awq: Activation-aware weight quantization for on-device llm compression and acceleration,” *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024.
- [8] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, *et al.*, “Smoothquant: Accurate and efficient post-training quantization for large language models,” in *International conference on machine learning*, pp. 38087–38099, PMLR, 2023.
- [9] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, *et al.*, “Qserve: W4a8kv4 quantization and system co-design for efficient llm serving,” *arXiv preprint arXiv:2405.04532*, 2024.
- [10] S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, *et al.*, “Quarot: Outlier-free 4-bit inference in rotated llms,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 100213–100240, 2024.
- [11] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, *et al.*, “A survey of quantization methods for efficient neural network inference,” in *Low-power computer vision*, pp. 291–326, Chapman and Hall/CRC, 2022.
- [12] E. Frantar and D. Alistarh, “Optimal brain compression: A framework for accurate post-training quantization and pruning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 4475–4488, 2022.
- [13] E. Frantar, S. Ashkboos, T. Hoeftler, and D. Alistarh, “Gptq: Accurate post-training quantization for generative pre-trained transformers,” *arXiv preprint arXiv:2210.17323*, 2022.
- [14] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, “Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 13355–13364, 2024.
- [15] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, *et al.*, “A white paper on neural network quantization,” *arXiv preprint arXiv:2106.08295*, 2021.
- [16] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “Haq: Hardware-aware automated quantization with mixed precision,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8612–8620, 2019.
- [17] C. Tang, K. Ouyang, Z. Wang, Y. Zhu, W. Ji, *et al.*, “Mixed-precision neural network quantization via learned layer-wise importance,” in *European conference on computer vision*, pp. 259–275, Springer, 2022.
- [18] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “Hawq: Hessian aware quantization of neural networks with mixed-precision,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 293–302, 2019.
- [19] Z. Guan, H. Huang, Y. Su, H. Huang, N. Wong, *et al.*, “Aptq: Attention-aware post-training mixed-precision quantization for large language models,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference, DAC ’24*, p. 1–6, ACM, June 2024.
- [20] N. P. Pandey, M. Nagel, M. van Baalen, Y. Huang, C. Patel, *et al.*, “A practical mixed precision algorithm for post-training quantization,” *arXiv preprint arXiv:2302.05397*, 2023.
- [21] H. You, Y. Guo, Y. Fu, W. Zhou, H. Shi, *et al.*, “Shiftaddllm: Accelerating pretrained llms via post-training multiplication-less reparameterization,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 24822–24848, 2024.
- [22] G. Park, H. Kwon, J. Kim, J. Bae, B. Park, *et al.*, “Figlut: An energy-efficient accelerator design for fp-int gemm using look-up tables,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1098–1111, IEEE, 2025.
- [23] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, “Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale,” *Advances in neural information processing systems*, vol. 35, pp. 30318–30332, 2022.
- [24] Z. Liu, C. Zhao, I. Fedorov, B. Soran, D. Choudhary, *et al.*, “Spinquant: Llm quantization with learned rotations,” *arXiv preprint arXiv:2405.16406*, 2024.
- [25] Z. Liu, C. Zhao, F. Iandola, C. Lai, Y. Tian, *et al.*, “Mobilellm: Optimizing sub-billion parameter language models for on-device use cases,” in *Forty-first International Conference on Machine Learning*, 2024.
- [26] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, *et al.*, “The llama 3 herd of models,” *arXiv e-prints*, pp. arXiv–2407, 2024.
- [27] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.
- [28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
- [29] S. Merity, C. Xiong, J. Bradbury, and R. Socher, “Pointer sentinel mixture models,” *arXiv preprint arXiv:1609.07843*, 2016.
- [30] L. Gao, J. Tow, S. Biderman, C. Llovering, J. Phang, *et al.*, “Eleutherai/llm-evaluation-harness: v0.3.0,” Dec. 2022.
- [31] Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi, “Piqa: Reasoning about physical commonsense in natural language,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 7432–7439, 2020.
- [32] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, “Winogrande: An adversarial winograd schema challenge at scale,” *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.
- [33] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, *et al.*, “Think you have solved question answering? try arc, the ai2 reasoning challenge,” *arXiv preprint arXiv:1803.05457*, 2018.
- [34] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, “Can a suit of armor conduct electricity? a new dataset for open book question answering,” *arXiv preprint arXiv:1809.02789*, 2018.
- [35] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, *et al.*, “Measuring massive multitask language understanding,” *arXiv preprint arXiv:2009.03300*, 2020.
- [36] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, “Cacti 7: New tools for interconnect exploration in innovative off-chip memories,” vol. 14, June 2017.