iKVC: Inference-Time KV-Cache Correction for Quantized LLMs in Long-Horizon Tasks

Anonymous authors
Paper under double-blind review

Abstract

Post-training quantization (PTQ) reduces the memory footprint of large language models (LLMs) and can improve inference throughput. However, the resulting quantization errors can significantly degrade accuracy, creating a fundamental trade-off between the efficiency gained from quantization and the accuracy retained from the full-precision model, particularly in long-horizon generation. Prior work mitigates this degradation by preserving the KV cache of selected prompt tokens at full precision, thereby reducing errors introduced during prefill. This strategy, however, does not address errors introduced during decoding, as quantized weights continuously perturb the key and value representations of newly generated tokens, and these errors *accumulate* in the KV cache, altering subsequent predictions. We empirically observe that this error accumulation becomes non-negligible as a large number of tokens are generated, and we theoretically derive that the accumulated KV-cache errors can cause the attention output to deviate from the values computed by the original full-precision model. Motivated by this finding, we propose **inference-time KV-cache Correction (iKVC)**, a training-free inference method that periodically corrects the KV cache for both prompt and generated tokens before decoding errors substantially accumulate. The correction interval is configurable, allowing iKVC to explicitly control the accuracy–throughput trade-off. Across experiments, iKVC consistently improves the trade-off over PTQ alone in long-horizon tasks. Our results identify KV-cache error accumulation as an important, previously underexplored factor in achieving accurate and efficient long-horizon tasks with quantized LLMs, suggesting that managing such errors during inference can complement PTQ techniques.

1 Introduction

Large Language Models (LLMs), which utilize multi-layer neural networks with billions of parameters to infer natural language processing tasks, have demonstrated remarkable performance (Zhao et al., 2026). LLM inference consists of a prefill phase that processes the input sequence, followed by a decoding phase that autoregressively generates subword units (tokens). During inference, each attention layer obtains keys and values (KV), intermediate representations of tokens, using some pretrained parameters (weights) when computing the raw, unnormalized numerical scores (logit) that LLMs assign to every possible token in its vocabulary. Once computed, KV pairs are persistently stored in the memory region called KV cache and reused in subsequent decoding steps. Consequently, LLM inference is memory-intensive for storing both billions of parameters and the KV cache. This remains a major obstacle to deploying LLMs in memory-constrained environments, motivating extensive research on model compression (Wang et al., 2025b; Jingcun Wang, 2025).

Post-training quantization (PTQ), which converts pretrained parameters to low-bit precision (e.g., INT4, INT8), is a prominent approach to model compression (Nagel et al., 2020; Hubara et al., 2021; Frantar et al., 2022). Nevertheless, most existing PTQ techniques cannot reproduce exactly arbitrary full-precision parameters at low precision. This sometimes makes the quantized model's inference path deviate from that of the original full-precision model, thereby can degrade accuracy.

To mitigate this accuracy degradation, prior work (Liu et al., 2024) has proposed a prefill-time approach that eliminates errors in the KV cache for certain tokens in the system prompt. This approach enables PTQ-quantized models to begin inference with error-free KV pairs for the selected prompt

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

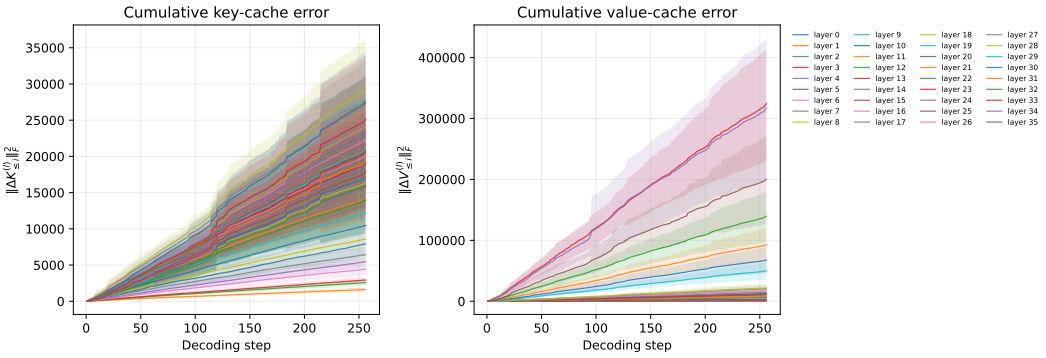

Figure 1: The error accumulation in the KV caches across decoding steps. The target model is Qwen3-4B-Thinking-2507 (BF16) and the quantization method is GPTQ (Frantar et al., 2023) ($n_{\text{bit}} = 4$, $n_{\text{group size}} = 128$). We here plot the squared Frobenius norm of the difference. See Section A for other PTQ techniques' results.

tokens, thereby recovering the accuracy in short- or middle-horizon tasks. However, eliminating only errors in the input KV cache is insufficient in long-horizon generation, because quantization errors in weights continuously propagate into newly computed KV pairs, *accumulating* in the KV cache during inference. We empirically observe that this error accumulation becomes non-negligible as a large number of tokens are generated (Figure 1), and theoretically derive that the accumulated KV-cache errors can cause the attention output to deviate from the values computed by the original full-precision model (see Section 3).

From the above insight, we propose **inference-time KV-cache Correction (iKVC)**, a training-free inference method that controls the accuracy–throughput trade-off of PTQ-quantized models in long-horizon tasks (Figure 2). It is achieved by periodically correcting the KV cache for generated tokens not only input tokens, before KV-cache errors accumulate during decoding with a configurable correction interval.

Experimental results demonstrate that iKVC overall recovers the mean accuracy of PTQ-quantized models and controls the accuracy–throughput trade-off in long-horizon tasks even under memory-constrained settings. Our results identify KV-cache error accumulation as an important, previously underexplored factor in achieving accurate and efficient long-horizon tasks with quantized LLMs, suggesting that managing such errors during inference can complement PTQ techniques.

Our contributions are threefold:

1. We empirically observe that this error accumulation becomes non-negligible as a large number of tokens are generated, and we theoretically derive that the accumulated KV-cache errors can cause the attention output to deviate from the values computed by the original full-precision model.

2. We propose iKVC, a training-free inference method that periodically corrects the KV cache during not only prefill phase but also decoding phase, preventing PTQ-induced errors from accumulating in the KV cache.

3. Experimental results demonstrate that iKVC overall recovers the mean accuracy of PTQ-quantized models and controls the accuracy–throughput trade-off in long-horizon tasks even under memory-constrained settings.

2 PRELIMINARIES

In this section, we briefly introduce the definitions and properties of internal states in the Transformer attention mechanism (Vaswani et al., 2017), and provide an overview of existing PTQ techniques.

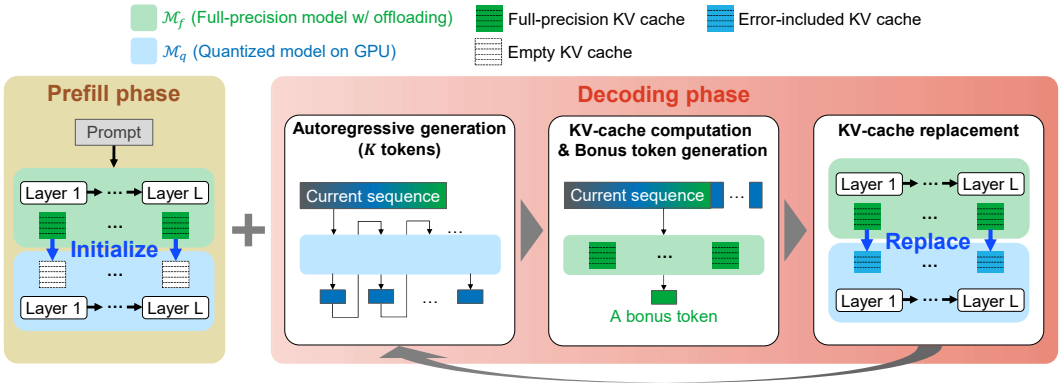

Figure 2: Illustration of iKVC. During the prefill phase, the full-precision model $\mathcal{M}_f$ computes the KV cache for the initial prompt and substitutes it into the quantized model $\mathcal{M}_q$. During the decoding phase, $\mathcal{M}_q$ autoregressively generates K tokens using its own KV cache. Then, $\mathcal{M}_f$ recomputes the KV cache for the same K tokens in parallel and generates a bonus token. Finally, $\mathcal{M}_q$'s KV cache is replaced with the corresponding full-precision KV cache recomputed by $\mathcal{M}_f$. This cycle will be repeated until the inference ends. After the first cycle, $\mathcal{M}_f$ processes the previous bonus token together with the K newly generated tokens in a single forward pass.

2.1 Formulation of Internal States in Attention Mechanism

Embedding and weight. Let $\mathcal{M}$ be a Transformer-based model, i be the token index, l be the layer index, d be the feature dimensionality, d_k be the dimensionality of the key vector, and d_v be the dimensionality of the value vector. $\mathcal{M}$ embeds the input texts into vector representations $\boldsymbol{X}_{\leq i}^{(l)} = [\boldsymbol{x}_1^{(l)}, \ldots, \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{i \times d}$. Then the high-dimensional representations are computed as *query*, *key*, and *value* (QKV) vectors $\boldsymbol{q}_i^{(l)} = \boldsymbol{x}_i^{(l)} \boldsymbol{W}_Q^{(l)}$, $\boldsymbol{k}_i^{(l)} = \boldsymbol{x}_i^{(l)} \boldsymbol{W}_K^{(l)}$, $\boldsymbol{v}_i^{(l)} = \boldsymbol{x}_i^{(l)} \boldsymbol{W}_V^{(l)}$, where $\boldsymbol{W}_{\mathrm{Q:V}}^{(l)} = \{\boldsymbol{W}_Q^{(l)} \in \mathbb{R}^{d \times d_k}, \boldsymbol{W}_K^{(l)} \in \mathbb{R}^{d \times d_k}, \boldsymbol{W}_V^{(l)} \in \mathbb{R}^{d \times d_v}\}$ are the projection weights for QKV vectors.

Attention output. $\mathcal{M}$ measures how much attention is focused on each token from all other tokens (i.e., the *attention output*) every layer as follows:

$$\boldsymbol{h}_i^{(l)} := \mathrm{softmax}\left(\frac{(\boldsymbol{x}_i^{(l)} \boldsymbol{W}_Q^{(l)})(\boldsymbol{X}_{\leq i}^{(l)} \boldsymbol{W}_K^{(l)})^\top}{\sqrt{d_k}} \right) (\boldsymbol{X}_{\leq i}^{(l)} \boldsymbol{W}_V^{(l)}) \tag{1}$$

$$= \mathrm{softmax}\left(\frac{\boldsymbol{q}_i^{(l)} \boldsymbol{K}_{\leq i}^{(l)\top}}{\sqrt{d_k}} \right) \boldsymbol{V}_{\leq i}^{(l)}, \tag{2}$$

where

$$\boldsymbol{K}_{\leq i}^{(l)} := \begin{bmatrix} \boldsymbol{k}_1^{(l)} \\ \vdots \\ \boldsymbol{k}_i^{(l)} \end{bmatrix} \in \mathbb{R}^{i \times d_k}, \ \boldsymbol{V}_{\leq i}^{(l)} := \begin{bmatrix} \boldsymbol{v}_1^{(l)} \\ \vdots \\ \boldsymbol{v}_i^{(l)} \end{bmatrix} \in \mathbb{R}^{i \times d_v}. \tag{3}$$

Throughout this paper, let us denote the *attention weight* as

$$\boldsymbol{\alpha}_i^{(l)} := \mathrm{softmax}\left(\frac{\boldsymbol{q}_i^{(l)} \boldsymbol{K}_{\leq i}^{(l)\top}}{\sqrt{d_k}} \right). \tag{4}$$

KV cache. To avoid recomputing $\boldsymbol{K}_{\leq i}^{(l)}$ and $\boldsymbol{V}_{\leq i}^{(l)}$ every step, the row-wise vectors for each token $\boldsymbol{k}_i^{(l)}$ and $\boldsymbol{v}_i^{(l)}$ are stored in a memory structure called *KV cache* once $\mathcal{M}$ computes them. Throughout this paper, we denote the KV cache of the l-th layer up to step i computed by $\mathcal{M}$ as

$$\mathcal{C}_{\mathcal{M},\leq i}^{(l)} := \left(\boldsymbol{K}_{\leq i}^{(l)}, \boldsymbol{V}_{\leq i}^{(l)} \right)_{\mathcal{M}}.$$

2.2 Overview of Post-Training Quantization

Post-training quantization. Let $Q(\boldsymbol{W}^{(l)})$ be a quantization operator which converts a weight $\boldsymbol{W}^{(l)}$ to a low-bit precision. The converted weight is then *dequantized* (i.e., *reconstructed*) when needed for attention computation during inference as $\hat{\boldsymbol{W}}^{(l)} := Q^{-1}(Q(\boldsymbol{W}^{(l)}))$, where $Q^{-1}(\cdot)$ is a dequantization operator. The reconstructed weight $\hat{\boldsymbol{W}}^{(l)}$ in the quantized model $\mathcal{M}_q$ entails a deviation from that of the original full-precision model $\mathcal{M}_f$ as $\Delta\boldsymbol{W}^{(l)} := \hat{\boldsymbol{W}}^{(l)} - \boldsymbol{W}^{(l)}$. This deviation $\Delta\boldsymbol{W}^{(l)}$ is generally referred to as *quantization error*.

Many efforts (Yao et al., 2022; Frantar et al., 2023; Lin et al., 2024; Chee et al., 2023) have been explored to find optimum reconstructed weights that minimize the discrepancy between $\mathcal{M}_q$'s layer output $\hat{\boldsymbol{W}}^{(l)}\boldsymbol{X}^{(l)}$ and $\mathcal{M}_f$'s layer output $\boldsymbol{W}^{(l)}\boldsymbol{X}^{(l)}$ (i.e., *layer-output error*) by solving the following optimization problem:

$$\min_{\hat{\boldsymbol{W}}^{(l)}} \|\boldsymbol{W}^{(l)}\boldsymbol{X}^{(l)} - \hat{\boldsymbol{W}}^{(l)}\boldsymbol{X}^{(l)}\|_F^2.$$

Recent studies have explored both calibration-free and gradient-based approaches to optimize $\hat{\boldsymbol{W}}^{(l)}$. Badri & Shaji (2023) optimize $\hat{\boldsymbol{W}}^{(l)}$ using only the minimum and maximum weight values and the distribution of quantization residuals, without requiring activation data $\boldsymbol{X}^{(l)}$. In contrast, Cheng et al. (2024) use a small calibration dataset and performs SGD-based optimization for $\hat{\boldsymbol{W}}^{(l)}$ in the block-output space. Despite these advances, the substantial accuracy degradation remains and can be pronounced in long-horizon generation tasks.

3 Analysis and Insight

In this section, we provide the theoretical analysis and insight that inspired the design of iKVC: (1) Quantization errors can propagate from weights to KV pairs, (2) Errors propagated to KV pairs accumulate in the KV cache during decoding, and (3) The accumulated KV-cache errors can cause the attention output to deviate from the values computed by the original full-precision model.

3.1 Quantization Error Propagation to the KV pairs

Quantization errors $\Delta\boldsymbol{W}^{(l)}$ can propagate to QKV representations, while errors propagated from preceding layers can additionally divert them. Hence, $\Delta\boldsymbol{W}^{(l)}$ induces QKV errors unless the weight-induced deviation is exactly canceled by the input-embedding deviation $\Delta\boldsymbol{x}^{(l)}$:

Lemma 1. *Let $\boldsymbol{a}_i^{(l)}$, $\boldsymbol{a} \in \{\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}\}$, denote a QKV vector for the i-th token of the l-th layer in the full-precision model, and let $\boldsymbol{a}_i^{(l)} + \Delta\boldsymbol{a}_i^{(l)}$ denote its counterpart in the quantized model. For the corresponding projection matrix $\boldsymbol{W}_A^{(l)}$, $A \in \{Q, K, V\}$, we have*

$$\Delta\boldsymbol{a}_i^{(l)} = \boldsymbol{x}_i^{(l)}\Delta\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{x}_i^{(l)}\left(\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{W}_A^{(l)}\right). \tag{5}$$

For $l = 1$, where $\Delta\boldsymbol{x}_i^{(1)} = 0$, $(\boldsymbol{x}_i^{(1)})^\top \notin \ker((\Delta\boldsymbol{W}_A^{(1)})^\top)$ is required.

Proof. See the proof of Lemma B.1. □

3.2 Error Accumulation in the KV Cache

The errors propagated to the KV pairs $\Delta\boldsymbol{k}_i^{(l)}$, $\Delta\boldsymbol{v}_i^{(l)}$ accumulate in the KV cache during decoding. Let us denote the error-included KV cache as

$$\begin{aligned}\mathcal{C}_{\mathcal{M}_q,\le i}^{(l)} &= \left(\begin{bmatrix}\boldsymbol{k}_1^{(l)} + \Delta\boldsymbol{k}_1^{(l)}\\ \vdots \\ \boldsymbol{k}_i^{(l)} + \Delta\boldsymbol{k}_i^{(l)}\end{bmatrix}, \begin{bmatrix}\boldsymbol{v}_1^{(l)} + \Delta\boldsymbol{v}_1^{(l)}\\ \vdots \\ \boldsymbol{v}_i^{(l)} + \Delta\boldsymbol{v}_i^{(l)}\end{bmatrix}\right)\\ &=: \left(\boldsymbol{K}_{\le i}^{(l)} + \Delta\boldsymbol{K}_{\le i}^{(l)}, \boldsymbol{V}_{\le i}^{(l)} + \Delta\boldsymbol{V}_{\le i}^{(l)}\right).\end{aligned}$$

The norms of KV-cache errors $\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2, \|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2$ increase in proportion to the number of decoding steps (Figure 1). We refer to this phenomenon as *KV-cache error accumulation*:

Lemma 2. *Let us define the KV-cache errors as*

$$\Delta \boldsymbol{K}_{\leq i}^{(l)} := \begin{bmatrix} \Delta \boldsymbol{k}_1^{(l)} \\ \vdots \\ \Delta \boldsymbol{k}_i^{(l)} \end{bmatrix}, \ \Delta \boldsymbol{V}_{\leq i}^{(l)} := \begin{bmatrix} \Delta \boldsymbol{v}_1^{(l)} \\ \vdots \\ \Delta \boldsymbol{v}_i^{(l)} \end{bmatrix}.$$

Assume that there exist constants $\sigma_K^2, \sigma_V^2 > 0$ such that $\mathbf{E}\left(\|\Delta \boldsymbol{k}_i^{(l)}\|_2^2\right) \geq \sigma_K^2$, $\mathbf{E}\left(\|\Delta \boldsymbol{v}_i^{(l)}\|_2^2\right) \geq \sigma_V^2$,. Then, we have

$$\forall l \in \{1, \cdots, L\}, \quad \mathbf{E}\left(\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2\right) \geq i\sigma_K^2, \ \mathbf{E}\left(\|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2\right) \geq i\sigma_V^2. \tag{6}$$

Proof. See the proof of Lemma B.2. □

3.3 The Impact of Errors in the KV Cache on Attention Outputs

In each attention layer, the KV-cache errors $\Delta \boldsymbol{K}_{\leq i}^{(l)}, \Delta \boldsymbol{V}_{\leq i}^{(l)}$ deviate the attention output at each layer from that of the original full-precision model, as KV matrices are used to compute it (Equation (2)). We refer to this difference as *attention-output error*. Importantly, the norm of the attention-output error in each layer is upper-bounded by a variable that depends on $\Delta \boldsymbol{q}_i^{(l)}$, $\Delta \boldsymbol{K}_{\leq i}^{(l)}$, and $\Delta \boldsymbol{V}_{\leq i}^{(l)}$:

Lemma 3. *Let $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{h}_i^{(l)} + \Delta \boldsymbol{h}_i^{(l)}$ denote the attention outputs of $\mathcal{M}_f$ and $\mathcal{M}_q$, respectively. Then we have*

$$\|\Delta \boldsymbol{h}_i^{(l)}\|_2 \leq B_{V,i}^{(l)} \left(e^{2\epsilon_i^{(l)}} - 1\right) + \|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F, \tag{7}$$

where

$$B_{V,i}^{(l)} := \max_{j \leq i} \|\boldsymbol{v}_j^{(l)}\|_2$$

and

$$\epsilon_i^{(l)} := \frac{1}{\sqrt{d_k}} \left[\|\Delta \boldsymbol{q}_i^{(l)}\|_2 \|\boldsymbol{K}_{\leq i}^{(l)}\|_F + \left(\|\boldsymbol{q}_i^{(l)}\|_2 + \|\Delta \boldsymbol{q}_i^{(l)}\|_2\right) \|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F\right]. \tag{8}$$

Proof. See the proof of Lemma B.3. □

3.4 Insight

Based on the analysis above, the following can be stated: *In inference using a PTQ-quantized model, errors accumulated in the KV cache can cause the attention output of each layer to deviate from corresponding values computed by the original full-precision model. This attention-output deviation can be significant enough to alter the inference path as the number of generated tokens increases.* Especially for long-horizon tasks, eliminating errors in the KV cache not only during prefill but also during decoding, before such errors accumulate, can be crucial for maintaining high accuracy.

4 Inference-time KV-Cache Correction

Motivated by the above analysis and insight, we seek to develop a novel inference method that removes error accumulation in the KV cache during both prefill and decoding, thereby suppressing the attention-output error upper bound, introducing modest throughput overhead. Given a quantized model $\mathcal{M}_q$ and its full-precision counterpart $\mathcal{M}_f$, we describe below our proposal, inference-time KV-cache correction (iKVC). See Appendix C for the explicit expression of the KV cache under iKVC. See Appendix D for the complete algorithm.

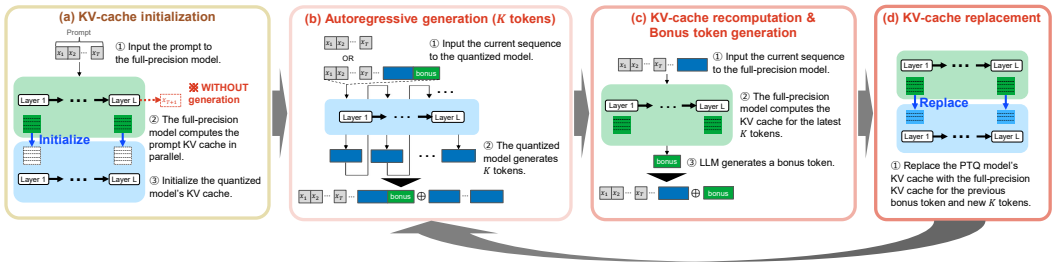

Figure 3: Details of iKVC. (a) During prefill, the full-precision model $\mathcal{M}_f$ computes the prompt KV cache, which initializes the cache of the quantized model $\mathcal{M}_q$. (b) $\mathcal{M}_q$ then autoregressively generates K tokens and appends their quantized KV pairs to its running cache. (c) After each generation interval, $\mathcal{M}_f$ processes the current sequence to recompute full-precision KV pairs for the correction interval. The same forward pass also provides the next-token distribution, from which a bonus token x_{b+1} is sampled without an additional decoding step by $\mathcal{M}_q$. (d) The corresponding recent KV pairs of $\mathcal{M}_q$ are replaced by their full-precision counterparts, and decoding resumes. Steps (b)–(d) are repeated until an EOS token is produced or the generation limit $N_{\max}$ is reached.

4.1 Algorithm

Prefill Phase. Given a prompt embedding $\boldsymbol{X}_{1:T} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, the original full-precision model $\mathcal{M}_f$ computes logits, obtaining the full-precision KV cache for the prompt. The empty KV cache of the quantized model $\mathcal{M}_q$ is then initialized with the full-precision KV cache (Figure 3 (a)). In Algorithm 1, we denote the above procedure as follows:

$$\{\mathcal{C}^{(l)}_{\mathcal{M}_q,1:T}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_f,1:T}\}_{l=1}^{L}.$$

Decoding Phase. In the first cycle, $\mathcal{M}_q$ receives the prompt and autoregressively generates K tokens $\boldsymbol{X}_{T+1:T+K}$ using its own KV cache (Figure 3 (b)), obtaining KV cache of length $K-1$. Next, $\mathcal{M}_f$ receives current sequence $\boldsymbol{X}_{1:T+K}$ and generates the bonus token x_{T+K+1}, obtaining the full-precision KV cache of length K in parallel (Figure 3 (c)). Then, reeplace the full-precision KV cache of length K for $\mathcal{M}_q$, and add the bonus token to the current sequence. In Algorithm 1, we denote the above procedure as follows:

$$\{\mathcal{C}^{(l)}_{\mathcal{M}_q,T+1:T+K-1}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_f,T+1:T+K}\}_{l=1}^{L},$$
$$\boldsymbol{X}_{1:T+K} \oplus x_{T+K+1}.$$

In the next cycle, $\mathcal{M}_q$ receives the current sequence $\boldsymbol{X}_{1:T+K+1}$ and autoregressively generates K tokens $\boldsymbol{X}_{T+K+2:T+2K+1}$(Figure 3 (b)), obtaining KV cache of length K. This KV cache is associated with tokens $\boldsymbol{X}_{T+K+1:T+2K}$. Next, $\mathcal{M}_f$ receives current sequence $X_{1:T+2K+1}$ and generates the bonus token x_{T+2K+2}, obtaining the full-precision KV cache of length $K+1$ in parallel (Figure 3 (c)). This full-precision KV cache is associated with tokens $\boldsymbol{X}_{T+K+1:T+2K+1}$. Then, reeplace the full-precision KV cache of length $K+1$ for $\mathcal{M}_q$, and add the bonus token to the current sequence. In Algorithm 1, we denote the above procedure as follows:

$$\{\mathcal{C}^{(l)}_{\mathcal{M}_q,T+K+1:T+2K}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_f,T+K+1:T+2K+1}\}_{l=1}^{L},$$
$$\boldsymbol{X}_{1:T+2K+1} \oplus x_{T+2K+2}.$$

iKVC repeatedly performs K-token generation, bonus-token generation, and KV-cache correction until an EOS token is generated or the number of generated output tokens reaches the maximum.

4.2 Theoretical Guarantee

Suppression for Attention-output error upper bound. Let r be the round index of the KV-cache correction. By eliminating errors accumulated in the quantized model's KV cache for up to $(r+1)K$-th token, iKVC suppresses the quantization-induced attention-output error described in

Section 3.3. Specifically, in the PTQ-quantized model to which iKVC is applied, the upper bound of the attention output error norm is reduced compared to the PTQ-quantized model alone. For simplicity, we present below the proposition where iKVC correction interval is fixed as $K = 1$:

Proposition 1. *Under the same assumptions as in Lemma 3, fix iKVC correction interval as $K = 1$. Let $\Delta \boldsymbol{h}_i^{(l),iKVC}, \Delta \boldsymbol{h}_i^{(l)}$ be the attention-output errors of the quantized models with and without iKVC, respectively. Let $U_i^{(l),iKVC} > 0, U_i^{(l)} > 0$ be their upper bounds in L_2 norm, i.e.,*

$$\|\Delta \boldsymbol{h}_i^{(l),iKVC}\|_2 \le U_i^{(l),iKVC},\ \|\Delta \boldsymbol{h}_i^{(l)}\|_2 \le U_i^{(l)}.$$

Then we have,

$$U_i^{(l),iKVC} \le U_i^{(l)}. \tag{9}$$

Proof. See the proof of Proposition B.1. □

Toy model to approximate the throughput overhead. We develop a toy model to approximate the throughput overhead introduced by iKVC. In each iKVC cycle after the first cycle, the following operations are performed until a total of $K + 1$ tokens are generated: K steps of autoregressive decoding using $\mathcal{M}_q$, parameter transfer of $\mathcal{M}_f$ if parameter offloading is applied, acquisition of the KV cache for $K + 1$ tokens using $\mathcal{M}_f$, and KV cache correction. Therefore, the throughput overhead introduced by iKVC can be approximated as

$$\mathrm{TPS}_{\mathrm{iKVC}}(K) = \frac{K+1}{K/\mathrm{TPS}_{\mathrm{PTQ}} + W_{\mathrm{off}}/B_{\mathrm{eff}} + T_{\mathrm{comp}}(K+1) + T_{\mathrm{KV}}(K+1)}, \tag{10}$$

where K is the correction interval, $\mathrm{TPS}_{\mathrm{PTQ}}$ is the throughput of standalone quantized model, W_{off} is the full-precision parameters residing in CPU, B_{eff} is the CPU-to-GPU transfer bandwidth, and T_{comp} and T_{KV} are the time for full-precision KV-cache computation and KV-cache replacement. In iKVC, we assume that $\mathcal{M}_q$ remains on the GPU while $\mathcal{M}_f$ is offloaded, making parameter communication a fixed overhead per iKVC cycle. Consequently, this overhead is expected to dominate for small K, whereas increasing K amortizes it across more generated tokens, bringing throughput closer to that of the standalone quantized model. Section 5.2 verifies whether established toy model well reproduce measured throughput.

5 Experiments

In this section, we evaluate (1) whether iKVC consistently recovers the mean accuracy of PTQ-quantized models, and (2) how much iKVC introduces the throughput overhead under memory-constrained settings. Finally, we discuss whether iKVC achieves better accuracy–throughput trade-off beyond PTQ. See Appendix E for experimental details.

5.1 Long-horizon generation quality

Given a language model with bfloat16 precision, we assess the capability of iKVC. As a baseline for iKVC, we consider IntactKV (Liu et al., 2024), which simply keeps a full-precision KV cache for certain tokens in the system prompt. As a quantization configuration, we consider 4 bits and 128 group sizes. As representative PTQ techniques, we consider HQQ (Badri & Shaji, 2023), GPTQ (Frantar et al., 2023), and AutoRound (Cheng et al., 2024). In Figure 4, we evaluated the mean accuracy of models with full-precision parameters (vanilla), quantized parameters (PTQ only), quantized parameters with the baseline (IntactKV with/without training), quantized parameters, and with iKVC on the benchmark AIME2025. The results show that iKVC overall recovers the mean accuracy of quantized models, exceeding the baseline. This suggests that our strategy to eliminate KV-cache errors before they accumulate during decoding works for long-horizon generations. In the case of AutoRound, additional calibration may occasionally yield stronger task-level performance, as observed for the baseline with training in our experiment. One possible explanation is that additional calibration may occasionally yield stronger task-level performance than direct KV-cache correction, because the weight distributions deviate significantly from their original values due to AutoRound's optimization for weight rounding and clipping.

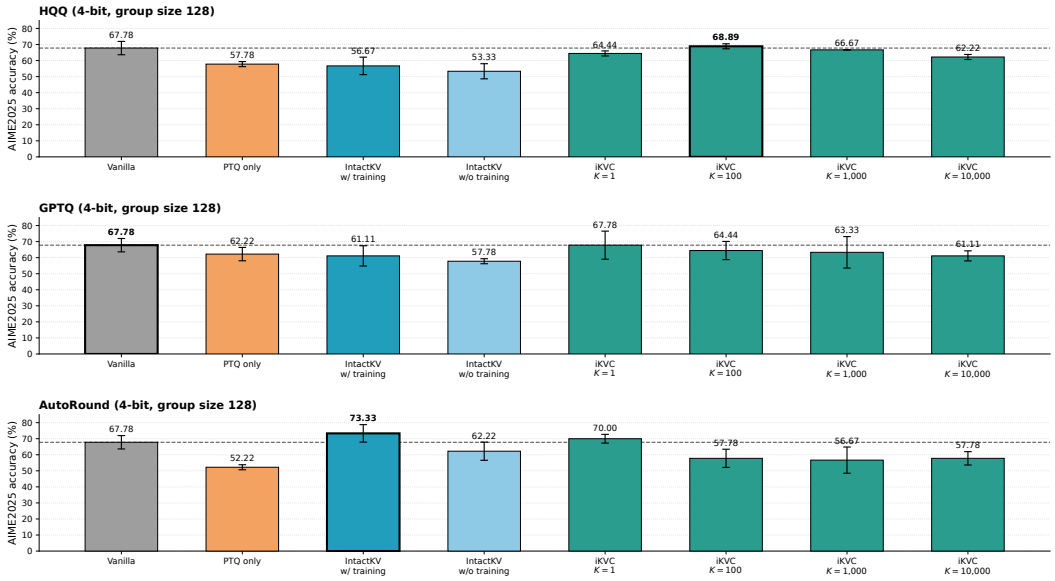

Figure 4: Results of mean accuracy averaged over three runs for models quantized via HQQ, GPTQ, and AutoRound, respectively. Vanilla (full-precision) refers to the original full-precision model (Qwen3-8B), PTQ only refers to the standalone quantized model to which iKVC is not applied, IntactKV w/ training and IntactKV w/o training refer to the quantized model to which the baseline is applied with/without training, and iKVC refers to our method with varying correction interval K from 1 to 10,000 tokens. See Appendix F for results of other model sizes.

5.2 Accuracy–throughput trade-off

Given a language model, a baseline, a quantization configuration, and PTQ techniques, as in Section 5.1, we assess (i) the efficiency of iKVC, and (ii) the toy model established in Section 4.2. As a memory-constrained setting, we use a GeForce RTX 3080 Ti GPU and an Intel i9-11900KF CPU (128GB of DDR4 RAM). The full-precision model is offloaded with memory limits based on each experimental mode (i.e., the full-precision model alone, or when used for KV cache correction in iKVC). We omit measurements for the quantized models to which the baseline is applied, as they do not introduce inference overhead and are expected to achieve the same results as the standalone quantized model. In Figure 5, we evaluated the real-world throughput overhead introduced by iKVC on the benchmark AIME2025. The results show that our toy model well approximates measured throughput, and that increasing K brings throughput closer to that of the standalone quantized model as expected in Section 4.2.

5.3 Summary

Combining the results in Sections 5.1 and 5.2, we found that iKVC provides a configurable accuracy–throughput trade-off, recovering accuracy relative to PTQ alone at suitable correction intervals in long-horizon tasks.

6 Related work

Accuracy recovery for PTQ-quantized LLMs. Several methods have been proposed to mitigate the accuracy degradation caused by PTQ. Xiao et al. (2023); Zhang et al. (2024a) reshape weight or activation distributions prior to quantization to better preserve layer outputs, while Liang Li (2024) adjusts model parameters after quantization to reduce deviations from the full-precision model. These methods rely on offline calibration and therefore do not directly address errors introduced during inference in practice. Liu et al. (2024) instead preserves the full-precision KV cache of selected prompt tokens, thereby reducing errors propagated from the prefill phase. However, because

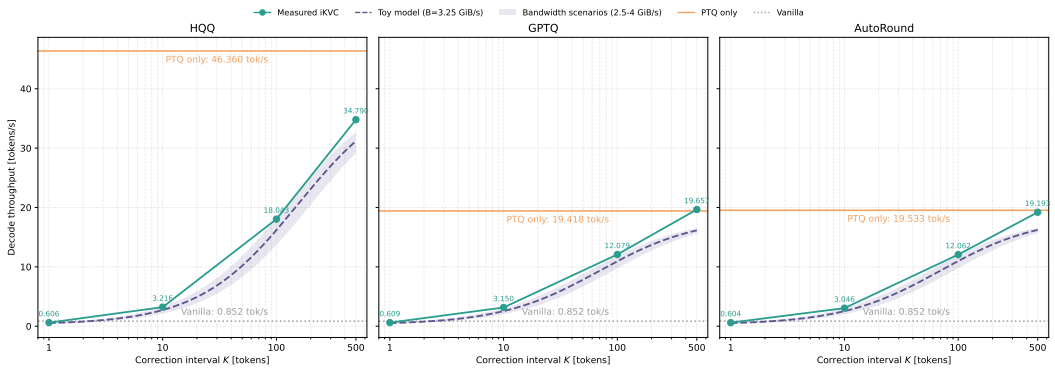

Figure 5: Results for the throughputs of models quantized via HQQ, GPTQ, and AutoRound, respectively, on AIME2025 using a GeForce RTX 3080 Ti GPU and an Intel i9-11900KF CPU, along with predictions from our toy model. Vanilla (full-precision) refers to the original full-precision model (Qwen3-8B), PTQ only refers to the standalone quantized model to which iKVC is not applied, iKVC refers to our method with varying correction interval K from 1 to 500 tokens, and a dotted line with an error band refers to the prediction from the toy model (Equation (10)).

its KV-cache correction is restricted to tokens available before decoding, it does not correct KV-cache errors newly introduced as tokens are generated autoregressively. Our method extends this idea to decoding by dynamically correcting the KV cache of generated tokens.

Collaborative decoding As a research direction distinct from PTQ, collaborative decoding has been actively studied to accelerate LLM inference. A common paradigm employs a lightweight model to rapidly generate candidate tokens, which are then verified in parallel by a more accurate model, with invalid candidates corrected as necessary (Leviathan et al., 2023; Li et al., 2024; Miao et al., 2024; Chen et al., 2024; Svirschevski et al., 2024; Zhang et al., 2024b; Wang et al., 2025a). Although both collaborative decoding and our method employ two language models during inference, their objectives fundamentally differ. Collaborative decoding uses the more accurate model primarily to verify candidate tokens generated by the lightweight model. In contrast, our method uses the full-precision model to recompute and correct the KV cache of the quantized model, rather than to verify its generated tokens.

7 Conclusion

In this paper, we investigated accuracy degradation in PTQ-quantized LLMs from the perspective of errors accumulated in the KV cache during autoregressive decoding. From empirical observations, we confirmed that quantization errors originating from low-precision weights propagate into KV pairs for newly generated tokens and accumulate in the KV cache, constituting a non-negligible as a large number of tokens are generated. Through theoretical analysis, we showed that the norm of the attention-output error in the quantized model is upper-bounded by a variable that depends on such accumulated KV-cache errors. Based on these insights, we proposed iKVC, a training-free inference method that periodically corrects the KV cache during not only prefill phase but also decoding phase. By adjusting the correction interval, iKVC provides explicit control over the trade-off between accuracy and throughput. Across the evaluated LLMs and PTQ techniques, iKVC consistently improves the accuracy–throughput trade-off compared to PTQ alone even under memory-constrained settings. Our findings highlight KV-cache error accumulation as an important factor in quantized LLM inference and suggest that managing errors during inference can complement PTQ techniques in long-horizon tasks.

AI use statement

In this work, we utilized generative AI tools to formulate mathematical claims, assist with writing proofs, and implement the experimental environment. We did not use these tools to develop con-

ceptual frameworks, design research methodologies, implement the proposed method, or interpret the results. Furthermore, tasks such as synthetic data generation, theoretical modeling, hypothesis formulation, data cleaning, and qualitative analysis were not applicable to this study. Additionally, generative AI was employed for code creation and editing, brainstorming, literature search, and manuscript copyediting. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

Reproducibility statement

All proofs are provided in Appendix B. The complete algorithm is provided in Appendix D. The experimental details are provided in Appendix E.

References

Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models. 2023. URL `https://dropbox.github.io/hqq_blog/`.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of llms. *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *The International Conference on Learning Representations (ICLR)*, 2023.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. *The International Conference on Machine Learning (ICML)*, 2021.

Ing-Chao Lin Bing Li Grace Li Zhang Jingcun Wang, Yu-Guang Chen. Basis sharing: Cross-layer parameter sharing for large language model compression. *The International Conference on Learning Representations (ICLR)*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *The International Conference on Machine Learning (ICML)*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Bo Zhang-Xiangxiang Chu Liang Li, Qingyuan Li. Norm tweaking: High-performance low-bit quantization of large language models. *The Association for the Advancement of Artificial Intelligence (AAAI)*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *The Conference on Machine Learning and Systems (MLSys)*, 2024.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. IntactKV: Improving large language model quantization by keeping pivot tokens intact. *The Association for Computational Linguistics (ACL)*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *The International Conference on Machine Learning (ICML)*, 2020.

Hikari Otsuka, Daiki Chijiwa, Yasuyuki Okoshi, Daichi Fujiki, Susumu Takeuchi, and Masato Motomura. The strong lottery ticket hypothesis for multi-head attention mechanisms. *The Association for the Advancement of Artificial Intelligence (AAAI)*, 2026.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Pei-Shuo Wang, Jian-Jia Chen, Chun-Che Yang, Chi-Chih Chang, Ning-Chi Huang, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Speculate deep and accurate: Lossless and training-free acceleration for offloaded llms via substitute speculative decoding. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2025a.

Xinghao Wang, Pengyu Wang, Bo Wang, Dong Zhang, Yunhua Zhou, and Xipeng Qiu. Bitstack: Any-size compression of large language models in variable memory environments. *The International Conference on Learning Representations (ICLR)*, 2025b.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *The International Conference on Machine Learning (ICML)*, 2023.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Aozhong Zhang, Naigang Wang, Yanxia Deng, Xin Li, Zi Yang, and Penghang Yin. Magr: Weight magnitude reduction for enhancing post-training quantization. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2024a.

Libo Zhang, Zhaoning Zhang, Baizhou Xu, Rui Li, Zhiliang Tian, Songzhu Mei, and Dongsheng Li. Dovetail: A cpu/gpu heterogeneous speculative decoding for llm inference. *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024b.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Front. Comput. Sci. 20, 2012627*, 2026.

A ADDITIONAL RESULTS OF ERROR ACCUMULATION IN THE KV CACHES

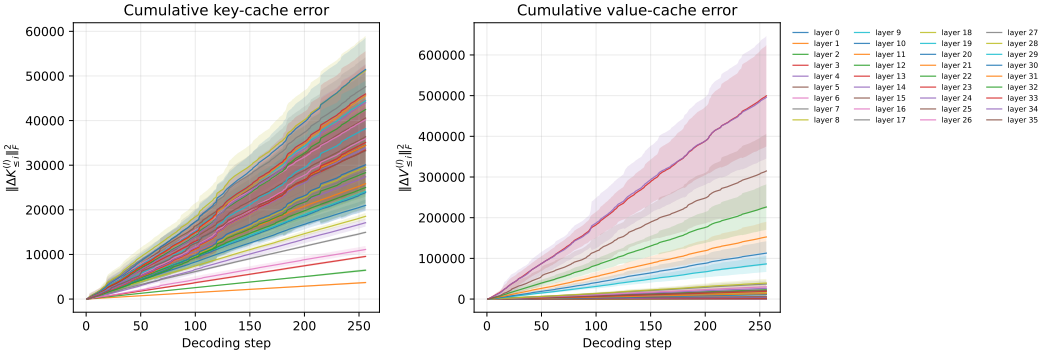

Figure 6: The error accumulation in the KV caches across decoding steps. The target model is Qwen3-4B-Thinking-2507 (bfloat16) and the quantization method is HQQ (Badri & Shaji, 2023) ($n_{\text{bit}} = 4$, $n_{\text{group size}} = 128$). We here plot the squared Frobenius norm of the difference.

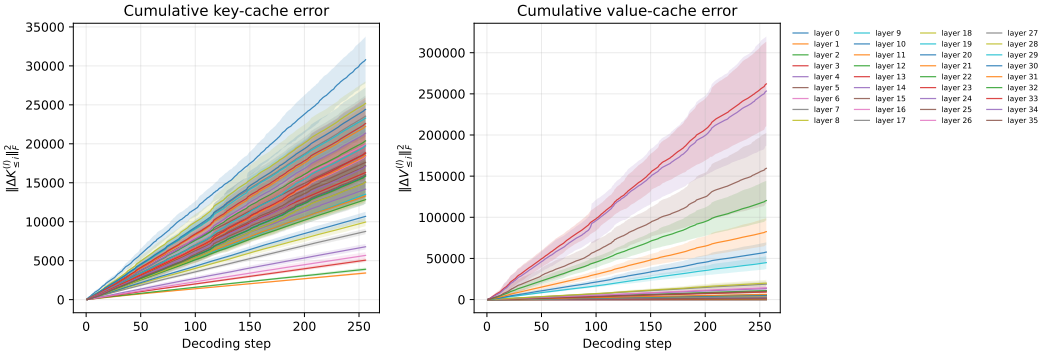

Figure 7: The error accumulation in the KV caches across decoding steps. The target model is Qwen3-4B-Thinking-2507 (bfloat16) and the quantization method is AutoRound (Cheng et al., 2024) ($n_{\text{bit}} = 4$, $n_{\text{group size}} = 128$). We here plot the squared Frobenius norm of the difference.

B PROOFS OF SECTIONS 3 AND 4

Lemma B.1. *Let $\boldsymbol{a}_i^{(l)}$, $\boldsymbol{a} \in \{\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}\}$, denote a QKV vector for the i-th token of the l-th layer in the full-precision model, and let $\boldsymbol{a}_i^{(l)} + \Delta\boldsymbol{a}_i^{(l)}$ denote its counterpart in the quantized model. For the corresponding projection matrix $\boldsymbol{W}_A^{(l)}$, $A \in \{Q, K, V\}$, we have*

$$\Delta\boldsymbol{a}_i^{(l)} = \boldsymbol{x}_i^{(l)}\Delta\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{x}_i^{(l)}\left(\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{W}_A^{(l)}\right). \tag{11}$$

For $l = 1$, where $\Delta\boldsymbol{x}_i^{(1)} = 0$, $(\boldsymbol{x}_i^{(1)})^\top \notin \ker((\Delta\boldsymbol{W}_A^{(1)})^\top)$ is required.

Proof. By definition, the QKV vector computed by the quantized model is

$$\boldsymbol{a}_i^{(l)} + \Delta\boldsymbol{a}_i^{(l)} = \left(\boldsymbol{x}_i^{(l)} + \Delta\boldsymbol{x}_i^{(l)}\right)\left(\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{W}_A^{(l)}\right) \tag{12}$$

$$= \boldsymbol{x}_i^{(l)}\boldsymbol{W}_A^{(l)} + \boldsymbol{x}_i^{(l)}\Delta\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{x}_i^{(l)}\boldsymbol{W}_A^{(l)} + \Delta\boldsymbol{x}_i^{(l)}\Delta\boldsymbol{W}_A^{(l)}. \tag{13}$$

Since

$$\boldsymbol{a}_i^{(l)} = \boldsymbol{x}_i^{(l)}\boldsymbol{W}_A^{(l)},$$

subtracting $\boldsymbol{a}_i^{(l)}$ from both sides gives

$$\Delta a_i^{(l)} = \boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{x}_i^{(l)} \boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)}, \tag{14}$$

$$= \boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{x}_i^{(l)} \left(\boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{W}_A^{(l)} \right). \tag{15}$$

Applying the triangle inequality and submultiplicativity of the spectral norm gives

$$\begin{aligned} \left\| \Delta \boldsymbol{a}_i^{(l)} \right\|_2 &\leq \left\| \boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)} \right\|_2 + \left\| \Delta \boldsymbol{x}_i^{(l)} \left(\boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{W}_A^{(l)} \right) \right\|_2 \\ &\leq \left\| \boldsymbol{x}_i^{(l)} \right\|_2 \left\| \Delta \boldsymbol{W}_A^{(l)} \right\|_2 + \left\| \Delta \boldsymbol{x}_i^{(l)} \right\|_2 \left\| \boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{W}_A^{(l)} \right\|_2 \\ &\leq \left\| \boldsymbol{x}_i^{(l)} \right\|_2 \left\| \Delta \boldsymbol{W}_A^{(l)} \right\|_2 + \left\| \Delta \boldsymbol{x}_i^{(l)} \right\|_2 \left(\left\| \boldsymbol{W}_A^{(l)} \right\|_2 + \left\| \Delta \boldsymbol{W}_A^{(l)} \right\|_2 \right). \end{aligned}$$

For $l = 1$, the two models receive the same input embedding before any quantized Transformer layer has been evaluated. Thus $\Delta \boldsymbol{x}_i^{(1)} = 0$, and Equation (15) reduces to

$$\Delta \boldsymbol{a}_i^{(1)} = \boldsymbol{x}_i^{(1)} \Delta \boldsymbol{W}_A^{(1)}.$$

The product $\boldsymbol{x}_i^{(1)} \Delta \boldsymbol{W}_A^{(1)}$ is zero if and only if

$$\left(\Delta \boldsymbol{W}_A^{(1)} \right)^\top \left(\boldsymbol{x}_i^{(1)} \right)^\top = 0,$$

or equivalently,

$$\left(\boldsymbol{x}_i^{(1)} \right)^\top \in \ker\left(\left(\Delta \boldsymbol{W}_A^{(1)} \right)^\top \right).$$

Hence the null-space condition in Lemma B.1 guarantees $\Delta \boldsymbol{a}_i^{(1)} \neq 0$.

For $l > 1$, both the projection weight and its input hidden state may differ between $\mathcal{M}_q$ and $\mathcal{M}_f$. From Equation (15), $\Delta \boldsymbol{a}_i^{(l)} = 0$ occurs exactly when

$$\boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)} = -\Delta x_i^{(l)} \left(\boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{W}_A^{(l)} \right).$$

Therefore, under the non-cancellation condition

$$\boldsymbol{x}_i^{(l)} \Delta \boldsymbol{W}_A^{(l)} \neq -\Delta \boldsymbol{x}_i^{(l)} \left(\boldsymbol{W}_A^{(l)} + \Delta \boldsymbol{W}_A^{(l)} \right),$$

we have $\Delta \boldsymbol{a}_i^{(l)} \neq 0$. □

Lemma B.2. *Let us define errors in the KV matrices (i.e., KV caches) $\Delta \boldsymbol{K}_{\leq i}^{(l)}$, $\Delta \boldsymbol{V}_{\leq i}^{(l)}$ as*

$$\Delta \boldsymbol{K}_{\leq i}^{(l)} := \begin{bmatrix} \Delta \boldsymbol{k}_1^{(l)} \\ \vdots \\ \Delta \boldsymbol{k}_i^{(l)} \end{bmatrix}, \; \Delta \boldsymbol{V}_{\leq i}^{(l)} := \begin{bmatrix} \Delta \boldsymbol{v}_1^{(l)} \\ \vdots \\ \Delta \boldsymbol{v}_i^{(l)} \end{bmatrix}.$$

Assume that there exist constants $\sigma_K^2, \sigma_V^2 > 0$ such that $\mathbf{E}\left(\|\Delta \boldsymbol{k}_i^{(l)}\|_2^2 \right) \geq \sigma_K^2$, $\mathbf{E}\left(\|\Delta \boldsymbol{v}_i^{(l)}\|_2^2 \right) \geq \sigma_V^2$, $\quad \forall l \in \{1, \cdots, L\}$. Then, we have

$$\forall l \in \{1, \cdots, L\}, \quad \mathbf{E}\left(\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2 \right) \geq i\sigma_K^2, \; \mathbf{E}\left(\|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2 \right) \geq i\sigma_V^2. \tag{16}$$

Proof. From the definition of the Frobenius norm, we have

$$\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2 = \sum_{j=1}^{i} \|\Delta \boldsymbol{k}_j^{(l)}\|_2^2, \; \|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2 = \sum_{j=1}^{i} \|\Delta \boldsymbol{v}_j^{(l)}\|_2^2.$$

Taking expectations on both sides and using the linearity of expectation, we have

$$\mathbf{E}\left(\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2\right) = \sum_{j=1}^{i} \mathbf{E}\left(\|\Delta \boldsymbol{k}_j^{(l)}\|_2^2\right), \tag{17}$$

$$\mathbf{E}\left(\|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2\right) = \sum_{j=1}^{i} \mathbf{E}\left(\|\Delta \boldsymbol{v}_j^{(l)}\|_2^2\right). \tag{18}$$

Substituting the lower bounds σ_K^2, σ_V^2 into Equations (17) and (18) yields

$$\forall l \in \{1, \cdots, L\}, \quad \mathbf{E}\left(\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F^2\right) \geq \sum_{j=1}^{i} \sigma_K^2 = i\sigma_K^2, \ \mathbf{E}\left(\|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F^2\right) \geq \sum_{j=1}^{i} \sigma_V^2 = i\sigma_V^2.$$

□

Lemma B.3. *Let $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{h}_i^{(l)} + \Delta \boldsymbol{h}_i^{(l)}$ denote the attention outputs of $\mathcal{M}_f$ and $\mathcal{M}_q$, respectively. Then we have*

$$\|\Delta \boldsymbol{h}_i^{(l)}\|_2 \leq B_{V,i}^{(l)} \left(e^{2\epsilon_i^{(l)}} - 1\right) + \|\Delta \boldsymbol{V}_{\leq i}^{(l)}\|_F, \tag{19}$$

where

$$B_{V,i}^{(l)} := \max_{j \leq i} \|\boldsymbol{v}_j^{(l)}\|_2$$

and

$$\epsilon_i^{(l)} := \frac{1}{\sqrt{d_k}} \left[\|\Delta \boldsymbol{q}_i^{(l)}\|_2 \|\boldsymbol{K}_{\leq i}^{(l)}\|_F + \left(\|\boldsymbol{q}_i^{(l)}\|_2 + \|\Delta \boldsymbol{q}_i^{(l)}\|_2\right) \|\Delta \boldsymbol{K}_{\leq i}^{(l)}\|_F\right]. \tag{20}$$

Proof. Let us define

$$\boldsymbol{\alpha}_i^{(l)} := \mathrm{softmax}\left(\frac{\boldsymbol{q}_i^{(l)} \left(\boldsymbol{K}_{\leq i}^{(l)}\right)^\top}{\sqrt{d_k}}\right), \qquad \boldsymbol{\alpha}_i^{(l)} + \Delta \boldsymbol{\alpha}_i^{(l)} := \mathrm{softmax}\left(\frac{\left(\boldsymbol{q}_i^{(l)} + \Delta \boldsymbol{q}_i^{(l)}\right) \left(\boldsymbol{K}_{\leq i}^{(l)} + \Delta \boldsymbol{K}_{\leq i}^{(l)}\right)^\top}{\sqrt{d_k}}\right),$$

$$B_{V,i}^{(l)} := \max_{j \leq i} \left\|\boldsymbol{v}_j^{(l)}\right\|_2, \tag{21}$$

and

$$\epsilon_i^{(l)} := \frac{1}{\sqrt{d_k}} \left[\left\|\Delta \boldsymbol{q}_i^{(l)}\right\|_2 \left\|\boldsymbol{K}_{\leq i}^{(l)}\right\|_F + \left(\left\|\boldsymbol{q}_i^{(l)}\right\|_2 + \left\|\Delta \boldsymbol{q}_i^{(l)}\right\|_2\right) \left\|\Delta \boldsymbol{K}_{\leq i}^{(l)}\right\|_F\right]. \tag{22}$$

By definition,

$$\boldsymbol{h}_i^{(l)} = \boldsymbol{\alpha}_i^{(l)} \boldsymbol{V}_{\leq i}^{(l)}, \qquad \boldsymbol{h}_i^{(l)} + \Delta \boldsymbol{h}_i^{(l)} = \left(\boldsymbol{\alpha}_i^{(l)} + \Delta \boldsymbol{\alpha}_i^{(l)}\right) \left(\boldsymbol{V}_{\leq i}^{(l)} + \Delta \boldsymbol{V}_{\leq i}^{(l)}\right).$$

Then

$$\Delta \boldsymbol{h}_i^{(l)} = \Delta \boldsymbol{\alpha}_i^{(l)} \boldsymbol{V}_{\leq i}^{(l)} + \left(\boldsymbol{\alpha}_i^{(l)} + \Delta \boldsymbol{\alpha}_i^{(l)}\right) \Delta \boldsymbol{V}_{\leq i}^{(l)}. \tag{23}$$

Applying the triangle inequality gives

$$\left\|\Delta \boldsymbol{h}_i^{(l)}\right\|_2 \leq \left\|\Delta \boldsymbol{\alpha}_i^{(l)} \boldsymbol{V}_{\leq i}^{(l)}\right\|_2 + \left\|\widehat{\boldsymbol{\alpha}}_i^{(l)} \Delta \boldsymbol{V}_{\leq i}^{(l)}\right\|_2. \tag{24}$$

We first bound the first term of Equation (24). Since

$$\Delta \boldsymbol{\alpha}_i^{(l)} \boldsymbol{V}_{\leq i}^{(l)} = \sum_{j=1}^{i} \Delta \boldsymbol{\alpha}_{i,j}^{(l)} \boldsymbol{v}_j^{(l)},$$

we obtain

$$\begin{aligned} \left\|\Delta \boldsymbol{\alpha}_i^{(l)} \boldsymbol{V}_{\leq i}^{(l)}\right\|_2 &\leq \sum_{j=1}^{i} \left|\Delta \boldsymbol{\alpha}_{i,j}^{(l)}\right| \left\|\boldsymbol{v}_j^{(l)}\right\|_2 \\ &\leq \left\|\Delta \boldsymbol{\alpha}_i^{(l)}\right\|_1 \max_{j \leq i} \left\|\boldsymbol{v}_j^{(l)}\right\|_2 \\ &= B_{V,i}^{(l)} \left\|\Delta \boldsymbol{\alpha}_i^{(l)}\right\|_1. \end{aligned} \tag{25}$$

For the second term of Equation (24), because $\boldsymbol{\alpha}_i^{(l)} + \Delta\boldsymbol{\alpha}_i^{(l)}$ is a probability vector,

$$\widehat{\boldsymbol{\alpha}}_{i,j}^{(l)} \geq 0, \qquad \sum_{j=1}^{i} \widehat{\boldsymbol{\alpha}}_{i,j}^{(l)} = 1.$$

Hence

$$\begin{aligned}
\left\| \left(\boldsymbol{\alpha}_i^{(l)} + \Delta\boldsymbol{\alpha}_i^{(l)} \right) \Delta \boldsymbol{V}_{\leq i}^{(l)} \right\|_2 &= \left\| \sum_{j=1}^{i} \left(\boldsymbol{\alpha}_i^{(l)} + \Delta\boldsymbol{\alpha}_i^{(l)} \right) \Delta \boldsymbol{v}_j^{(l)} \right\|_2 \\
&\leq \sum_{j=1}^{i} \left(\boldsymbol{\alpha}_i^{(l)} + \Delta\boldsymbol{\alpha}_i^{(l)} \right) \left\| \Delta \boldsymbol{v}_j^{(l)} \right\|_2 \\
&\leq \left(\sum_{j=1}^{i} \left(\boldsymbol{\alpha}_i^{(l)} + \Delta\boldsymbol{\alpha}_i^{(l)} \right) \right) \left(\sum_{j=1}^{i} \left\| \Delta \boldsymbol{v}_j^{(l)} \right\|_2^2 \right)^{1/2} \\
&= \left\| \Delta \boldsymbol{V}_{\leq i}^{(l)} \right\|_F . \qquad (26)
\end{aligned}$$

It remains to bound $\|\Delta\boldsymbol{\alpha}_i^{(l)}\|_1$. Here, we refer to Otsuka et al. (2026). Define the pre-softmax attention-score vectors as

$$\boldsymbol{s}_i^{(l)} := \frac{\boldsymbol{q}_i^{(l)} \left(\boldsymbol{K}_{\leq i}^{(l)} \right)^\top}{\sqrt{d_k}}, \qquad \boldsymbol{s}_i^{(l)} + \Delta\boldsymbol{s}_i^{(l)} := \frac{\left(\boldsymbol{q}_i^{(l)} + \Delta\boldsymbol{q}_i^{(l)} \right) \left(\boldsymbol{K}_{\leq i}^{(l)} + \Delta\boldsymbol{K}_{\leq i}^{(l)} \right)^\top}{\sqrt{d_k}}.$$

Their difference is

$$\Delta\boldsymbol{s}_i^{(l)} = \frac{1}{\sqrt{d_k}} \left[\Delta\boldsymbol{q}_i^{(l)} \left(\boldsymbol{K}_{\leq i}^{(l)} \right)^\top + \left(\boldsymbol{q}_i^{(l)} + \Delta\boldsymbol{q}_i^{(l)} \right) \left(\Delta\boldsymbol{K}_{\leq i}^{(l)} \right)^\top \right]. \qquad (27)$$

For every coordinate $j \leq i$, Cauchy–Schwarz gives

$$\begin{aligned}
\left| \Delta\boldsymbol{s}_{i,j}^{(l)} \right| &\leq \frac{1}{\sqrt{d_k}} \left[\left\| \Delta\boldsymbol{q}_i^{(l)} \right\|_2 \left\| \boldsymbol{k}_j^{(l)} \right\|_2 + \left\| \boldsymbol{q}_i^{(l)} + \Delta\boldsymbol{q}_i^{(l)} \right\|_2 \left\| \Delta\boldsymbol{k}_j^{(l)} \right\|_2 \right] \\
&\leq \frac{1}{\sqrt{d_k}} \left[\left\| \Delta\boldsymbol{q}_i^{(l)} \right\|_2 \left\| \boldsymbol{K}_{\leq i}^{(l)} \right\|_F + \left(\left\| \boldsymbol{q}_i^{(l)} \right\|_2 + \left\| \Delta\boldsymbol{q}_i^{(l)} \right\|_2 \right) \left\| \Delta\boldsymbol{K}_{\leq i}^{(l)} \right\|_F \right] \\
&= \epsilon_i^{(l)}. \qquad (28)
\end{aligned}$$

Therefore,

$$\left\| \Delta\boldsymbol{s}_i^{(l)} \right\|_\infty \leq \epsilon_i^{(l)}. \qquad (29)$$

For each coordinate $m \leq i$,

$$\alpha_{i,m}^{(l)} = \frac{e^{s_{i,m}^{(l)}}}{\sum_{j=1}^{i} e^{s_{i,j}^{(l)}}}, \qquad \alpha_{i,m}^{(l)} + \Delta\alpha_{i,m}^{(l)} = \frac{e^{s_{i,m}^{(l)} + \Delta s_{i,m}^{(l)}}}{\sum_{j=1}^{i} e^{s_{i,j}^{(l)} + \Delta s_{i,j}^{(l)}}}.$$

Hence

$$\frac{\alpha_{i,m}^{(l)} + \Delta\alpha_{i,m}^{(l)}}{\alpha_{i,m}^{(l)}} = \frac{e^{\Delta s_{i,m}^{(l)}}}{\sum_{j=1}^{i} \alpha_{i,j}^{(l)} e^{\Delta s_{i,j}^{(l)}}}. \qquad (30)$$

Equation (29) implies

$$e^{-\epsilon_i^{(l)}} \leq e^{\Delta s_{i,j}^{(l)}} \leq e^{\epsilon_i^{(l)}} \qquad \forall j \leq i.$$

Since $\sum_j \alpha_{i,j}^{(l)} = 1$, it follows that

$$e^{-\epsilon_i^{(l)}} \leq \sum_{j=1}^{i} \alpha_{i,j}^{(l)} e^{\Delta s_{i,j}^{(l)}} \leq e^{\epsilon_i^{(l)}}.$$

Therefore,

$$e^{-2\epsilon_i^{(l)}} \leq \frac{\alpha_{i,m}^{(l)} + \Delta\alpha_{i,m}^{(l)}}{\alpha_{i,m}^{(l)}} \leq e^{2\epsilon_i^{(l)}}. \tag{31}$$

Thus

$$\left|\Delta\alpha_{i,m}^{(l)}\right| \leq \left(e^{2\epsilon_i^{(l)}} - 1\right)\alpha_{i,m}^{(l)}.$$

Summing over m gives

$$\left\|\Delta\boldsymbol{\alpha}_i^{(l)}\right\|_1 \leq e^{2\epsilon_i^{(l)}} - 1. \tag{32}$$

Finally, substituting Equations (25), (26), and (32) into Equation (24) yields

$$\left\|\Delta\boldsymbol{h}_i^{(l)}\right\|_2 \leq B_{V,i}^{(l)}\left(e^{2\epsilon_i^{(l)}} - 1\right) + \left\|\Delta\boldsymbol{V}_{\leq i}^{(l)}\right\|_F.$$

The nonnegativity of a norm gives the lower bound $0 \leq \|\Delta\boldsymbol{h}_i^{(l)}\|_2$, completing the proof. □

Proposition B.1. *Under the same assumptions as in Lemma 3, fix iKVC correction interval as $K = 1$. Let $\Delta\boldsymbol{h}_i^{(l),iKVC}, \Delta\boldsymbol{h}_i^{(l)}$ be the attention-output errors of the quantized models with and without iKVC, respectively. Let $U_i^{(l),iKVC} > 0, U_i^{(l)} > 0$ be their upper bounds in L_2 norm, i.e.,*

$$\|\Delta\boldsymbol{h}_i^{(l),iKVC}\|_2 \leq U_i^{(l),iKVC},\ \|\Delta\boldsymbol{h}_i^{(l)}\|_2 \leq U_i^{(l)}.$$

Then we have,

$$\forall K \in \mathbb{N}_{>0}, \quad U_i^{(l),iKVC} \leq U_i^{(l)}. \tag{33}$$

Proof. At decoding step i, for correction interval $K = 1$, all past KV-cache entries $j < i$ have been replaced by their corresponding full-precision KV cache. Therefore, all past KV-cache errors are zero:

$$\forall j < i, \qquad \Delta\boldsymbol{k}_j^{(l),\mathrm{iKVC}} = \mathbf{0}, \qquad \Delta\boldsymbol{v}_j^{(l),\mathrm{iKVC}} = \mathbf{0}.$$

If $K = 1$, the set of uncorrected KV-cache rows at step i reduces to $\mathcal{R}_{i,1} = \{i\}$. Therefore, only the current step i can contain a nonzero KV-cache quantization error.

Consequently, the maximum KV-cache errors under iKVC for $K = 1$ simplify to:

$$B_{\Delta K,i}^{(l),\mathrm{iKVC}} := \max_{j\leq i}\left\|\Delta\boldsymbol{k}_j^{(l),\mathrm{iKVC}}\right\|_2 = \left\|\Delta\boldsymbol{k}_i^{(l)}\right\|_2, \tag{34}$$

$$B_{\Delta V,i}^{(l),\mathrm{iKVC}} := \max_{j\leq i}\left\|\Delta\boldsymbol{v}_j^{(l),\mathrm{iKVC}}\right\|_2 = \left\|\Delta\boldsymbol{v}_i^{(l)}\right\|_2. \tag{35}$$

By Lemma 3, the attention-output error bound under iKVC is

$$\left\|\Delta\boldsymbol{h}_i^{(l),\mathrm{iKVC}}\right\|_2 \leq \max_{j\leq i}\left\|\boldsymbol{v}_j^{(l)}\right\|_2\left(e^{2\epsilon_i^{(l),\mathrm{iKVC}}} - 1\right) + B_{\Delta V,i}^{(l),\mathrm{iKVC}} := U_i^{(l),\mathrm{iKVC}}, \tag{36}$$

where

$$\epsilon_i^{(l),\mathrm{iKVC}} := \frac{1}{\sqrt{d_k}}\left[\left\|\Delta\boldsymbol{q}_i^{(l)}\right\|_2 \max_{j\leq i}\left\|\boldsymbol{k}_j^{(l)}\right\|_2 + \left(\left\|\boldsymbol{q}_i^{(l)}\right\|_2 + \left\|\Delta\boldsymbol{q}_i^{(l)}\right\|_2\right)B_{\Delta K,i}^{(l),\mathrm{iKVC}}\right].$$

Since $i \in \{1, \ldots, i\}$, Equations (34) and (35) satisfy:

$$B_{\Delta K,i}^{(l),\mathrm{iKVC}} = \left\|\Delta\boldsymbol{k}_i^{(l)}\right\|_2 \leq \max_{j\leq i}\left\|\Delta\boldsymbol{k}_j^{(l)}\right\|_2 = B_{\Delta K,i}^{(l)}, \tag{37}$$

$$B_{\Delta V,i}^{(l),\mathrm{iKVC}} = \left\|\Delta\boldsymbol{v}_i^{(l)}\right\|_2 \leq \max_{j\leq i}\left\|\Delta\boldsymbol{v}_j^{(l)}\right\|_2 = B_{\Delta V,i}^{(l)}. \tag{38}$$

Substituting Equation (37) into the expression for $\epsilon_i^{(l),\mathrm{iKVC}}$ yields:

$$\begin{aligned}\epsilon_i^{(l),\mathrm{iKVC}} &\leq \frac{1}{\sqrt{d_k}}\left[\left\|\Delta\boldsymbol{q}_i^{(l)}\right\|_2 \max_{j\leq i}\left\|\boldsymbol{k}_j^{(l)}\right\|_2 + \left(\left\|\boldsymbol{q}_i^{(l)}\right\|_2 + \left\|\Delta\boldsymbol{q}_i^{(l)}\right\|_2\right)\max_{j\leq i}\left\|\Delta\boldsymbol{k}_j^{(l)}\right\|_2\right] \\ &= \epsilon_i^{(l)}.\end{aligned}$$

Since $x \mapsto e^{2x} - 1$ is monotonically increasing for $x \geq 0$, we have $e^{2\epsilon_i^{(l),\mathrm{iKVC}}} - 1 \leq e^{2\epsilon_i^{(l)}} - 1$. Combining this with Equation (38):

$$\begin{aligned} U_i^{(l),\mathrm{iKVC}} &= \max_{j \leq i} \left\| \boldsymbol{v}_j^{(l)} \right\|_2 \left(e^{2\epsilon_i^{(l),\mathrm{iKVC}}} - 1 \right) + B_{\Delta V,i}^{(l),\mathrm{iKVC}} \\ &\leq \max_{j \leq i} \left\| \boldsymbol{v}_j^{(l)} \right\|_2 \left(e^{2\epsilon_i^{(l)}} - 1 \right) + \max_{j \leq i} \left\| \Delta \boldsymbol{v}_j^{(l)} \right\|_2 \\ &= U_i^{(l)}. \end{aligned}$$

Finally, for $K = 1$, iKVC guarantees $U_i^{(l),\mathrm{iKVC}} \leq U_i^{(l)}$. □

C Explicit expression of the KV cache

Let $\mathcal{C}'$ denote the KV cache to which iKVC has been applied (modified), and let $\mathcal{C}$ denote the KV cache to which iKVC has not been applied. The KV cache of the l-th layer in the quantized model $\mathcal{M}_q$ immediately after it has generated the tokens from the $i+1$-th to the $i+K$-th token can be explicitly expressed as follows:

$$\begin{aligned} \mathcal{C}_{\leq i+K-1}^{(l)} &= \left(\mathcal{C}_{\mathcal{M}_f, \leq i-1}^{(l)}, \mathcal{C}_{\mathcal{M}_q, i:i+K-1}^{(l)} \right) \\ &= \left(\begin{bmatrix} \boldsymbol{k}_1^{(l)} \\ \vdots \\ \boldsymbol{k}_{i-1}^{(l)} \\ \boldsymbol{k}_i^{(l)} + \Delta \boldsymbol{k}_i^{(l)} \\ \boldsymbol{k}_{i+1}^{(l)} + \Delta \boldsymbol{k}_{i+1}^{(l)} \\ \vdots \\ \boldsymbol{k}_{i+K-1}^{(l)} + \Delta \boldsymbol{k}_{i+K-1}^{(l)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{v}_1^{(l)} \\ \vdots \\ \boldsymbol{v}_{i-1}^{(l)} \\ \boldsymbol{v}_i^{(l)} + \Delta \boldsymbol{v}_i^{(l)} \\ \boldsymbol{v}_{i+1}^{(l)} + \Delta \boldsymbol{v}_{i+1}^{(l)} \\ \vdots \\ \boldsymbol{v}_{i+K-1}^{(l)} + \Delta \boldsymbol{v}_{i+K-1}^{(l)} \end{bmatrix} \right), \end{aligned}$$

where the cache corresponding to tokens up to the $i-1$-th token does not contain errors $\Delta \boldsymbol{k},\ \Delta \boldsymbol{v}$ because the error-included cache is already replaced with the full-precision cache. The KV cache for the i-th token, a bonus token in the previous cycle, is computed in this current cycle by the quantized model. Note that, due to the characteristic of Transformer attention mechanisms where KV pairs of *previously generated tokens* are stored in a cache, the cache contains KV pairs only up to the $(i+K-1)$-th token once the sequence of tokens from the $(i+1)$-th to the $(i+K)$-th has been generated. The KV-cache after the correction can be expressed as follows:

$$\mathcal{C}'^{(l)}_{\leq i+K-1} \leftarrow \mathcal{C}_{\leq i+K}^{(l)} = \left(\begin{bmatrix} \boldsymbol{k}_1^{(l)} \\ \vdots \\ \boldsymbol{k}_{i-1}^{(l)} \\ \boldsymbol{k}_i^{(l)} \\ \boldsymbol{k}_{i+1}^{(l)} \\ \vdots \\ \boldsymbol{k}_{i+K-1}^{(l)} \\ \boldsymbol{k}_{i+K}^{(l)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{v}_1^{(l)} \\ \vdots \\ \boldsymbol{v}_{i-1}^{(l)} \\ \boldsymbol{v}_i^{(l)} \\ \boldsymbol{v}_{i+1}^{(l)} \\ \vdots \\ \boldsymbol{v}_{i+K-1}^{(l)} \\ \boldsymbol{v}_{i+K}^{(l)} \end{bmatrix} \right). $$

Here, the KV cache for the i-th token is simultaneously corrected along with those for the latest K generated tokens. Therefore, after correction, the height dimension of $\mathcal{M}_q$'s KV cache increases by one.

D Complete algorithm

Algorithm 1 Inference-Time KV-Cache Correction (iKVC)

Require: Quantized model $\mathcal{M}_q$, full-precision model $\mathcal{M}_f$, correction interval K, maximum number of generated tokens $N_{\max}$, and prompt $\boldsymbol{X}_{1:T}$
Ensure: Generated sequence $\boldsymbol{X}_{1:T+n}$

Prompt KV-Cache Initialization

$\{\mathcal{C}^{(l)}_{\mathcal{M}_f,1:T}\}_{l=1}^{L} \leftarrow \mathcal{M}_f(\boldsymbol{X}_{1:T})$. ▷ Compute the full-precision prompt KV cache
$\{\mathcal{C}^{(l)}_{\mathcal{M}_q,1:T}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_f,1:T}\}_{l=1}^{L}$. ▷ Initialize $\mathcal{M}_q$'s prompt KV cache
$n \leftarrow 0$. ▷ Number of generated output tokens
$c \leftarrow T$. ▷ Last token position whose KV cache has been corrected

Iterative Generation and KV-Cache Correction

while $n < N_{\max}$ **do**

K**-token generation by** $\mathcal{M}_q$
  for $j = 1, \ldots, K$ **do**
    if $n \geq N_{\max}$ **then**
      break
    end if
    $x_{T+n+1} \sim p_{\mathcal{M}_q}\left(\cdot \mid \boldsymbol{X}_{1:T+n}, \{\mathcal{C}^{(l)}_{\mathcal{M}_q}\}_{l=1}^{L}\right)$. ▷ Generate the next token using $\mathcal{M}_q$
    if $n > 0$ **then**
      $\{\mathcal{C}^{(l)}_{\mathcal{M}_q,1:T+n}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_q,1:T+n-1}\}_{l=1}^{L} \oplus \{\mathcal{C}^{(l)}_{\mathcal{M}_q,T+n}\}_{l=1}^{L}$. ▷ Append the KV of the token processed in this decoding step
    end if
    $\boldsymbol{X}_{1:T+n+1} \leftarrow \boldsymbol{X}_{1:T+n} \oplus x_{T+n+1}$.
    $n \leftarrow n + 1$.
    if $x_{T+n} = x_{\mathrm{EOS}}$ **then**
      break
    end if
  end for
  if $x_{T+n} = x_{\mathrm{EOS}} \vee n \geq N_{\max}$ **then**
    break
  end if

Full-precision KV acquisition

  $a \leftarrow c + 1$. ▷ First position not yet corrected
  $b \leftarrow T + n$. ▷ Last position after the K-token generation
  $\{\mathcal{C}^{(l)}_{\mathcal{M}_f,a:b}\}_{l=1}^{L} \leftarrow \mathcal{M}_f\left(\boldsymbol{X}_{a:b}\right)$. ▷ Obtain full-precision KV for every uncorrected token

KV-cache replacement

  $\{\mathcal{C}^{(l)}_{\mathcal{M}_q,a:b-1}\}_{l=1}^{L} \leftarrow \{\mathcal{C}^{(l)}_{\mathcal{M}_f,a:b}\}_{l=1}^{L}$. ▷ Correct all KV representations from a through b
  $c \leftarrow b$. ▷ Update the last corrected position

Bonus-token generation

  $x_{b+1} \sim p_{\mathcal{M}_f}\left(\cdot \mid \boldsymbol{X}_{1:b}, \{\mathcal{C}^{(l)}_{\mathcal{M}_f,1:b}\}_{l=1}^{L}\right)$.
  $\boldsymbol{X}_{1:T+n+1} \leftarrow \boldsymbol{X}_{1:T+n} \oplus x_{b+1}$. ▷ Append the bonus token
  $n \leftarrow n + 1$.
  if $x_{T+n} = x_{\mathrm{EOS}}$ **then**
    break
  end if
end while
return $\boldsymbol{X}_{1:T+n} = \{x_1, x_2, \ldots, x_{T+n}\}$.

E EXPERIMENTAL DETAILS

E.1 SETUPS

Hardwares.

- **H100 GPU**: An enterprise data center graphics processing unit based on the Hopper architecture. It features a dedicated Transformer Engine designed to accelerate LLMs' workloads. It has an 80GB memory capacity with HBM2e and supports the following computational precisions: FP16, TF32, FP64, INT8, FP8. We used this GPU to measure accuracy on AIME2025.
- **GeForce RTX 3080 Ti GPU**: A high-end desktop graphics card built on the Ampere architecture (GA102-225-A1 chip). It has a 12GB GDDR6X memory capacity. We used this GPU to measure throughput on AIME2025. To emulate inference under limited memory while avoiding unnecessary disk offloading, we set the memory budgets based on the role of the full-precision model in each configuration, as follows:
  - **PTQ+iKVC**: The 4-bit PTQ-quantized model is kept fully resident on the GPU and serves as the primary decoding model. Therefore, only the remaining GPU capacity is made available to the full-precision model used for periodic KV-cache correction. We restrict its parameter placement to 3 GiB of VRAM and 16 GiB of CPU RAM, so that most GPU memory remains available to the PTQ-quantized model while the full-precision weights can be accommodated through CPU offloading.
  - **Standalone full-precision model**: No PTQ-quantized model needs to coexist on the GPU, so the full-precision model can use most of the available device memory. We accordingly allow up to 11 GiB of VRAM and 12 GiB of CPU RAM, reserving part of the GPU memory for runtime allocations such as the KV cache, activations, and temporary buffers.

Models.

- **Qwen3-4B-Thinking-2507**: A causal LLM developed by the Qwen Team, having a total of 4B parameters, including 3.6B non-embedding parameters across 36 layers. It includes thinking trances.
- **Qwen3-8B**: A causal LLM developed by the Qwen Team, having a total of 8.2B parameters, including 6.95B non-embedding parameters across 36 layers.
- **Qwen3-14B**: A causal LLM developed by the Qwen Team, having a total of 14.8B parameters, including 13.2B non-embedding parameters across 40 layers.

Datasets.

- **AIME2025 (Balunović et al., 2025)**: A dataset consisting of 30 competition math questions from the 2025 American Invitational Mathematics Examination (AIME). It has been widely used to benchmark LLMs' reasoning capability. We reported the percentage of correct answers.
- **MT-Bench** A standardized evaluation framework designed to assess the multi-turn conversational and instruction-following capabilities of LLMs. It utilizes a curated set of 80 two-turn, open-ended questions, scored predominantly via an automated LLM-as-a-Judge pipeline with GPT-4. The task categories are as follows: Writing, Roleplay, Extraction, Reasoning, Math, Coding, Knowledge (STEM), and Knowledge (Humanities).

Generation configuration. We employed the sampling decoding for all evaluated models with the sampling parameters recommended by the official as follows:

- **Temperature**: 0.6,
- **TopP**: 0.95,
- **TopK**: 20,

- **MinP**: 0.

For the throughput measurements (Figure 5), a common setting limiting the maximum number of newly generated tokens to 4,000 was applied across all evaluated configurations.

PTQ techniques.

- **GPTQ (Frantar et al., 2023)**: A post-training quantization technique that uses the inverse Hessian matrix to calculate how individual weight changes affect the final layer output. It sets the weight quantization order in optimal brain quantization (OBQ) to be the same across all rows of the Hessian matrix (i.e., the weights are quantized row-wise, recursively).
- **AutoRound (Cheng et al., 2024)**: A post-training quantization technique that uses SGD-based backpropagation to optimize weight rounding and quantization parameters so that the outputs of each layer after quantization are as close as possible to the outputs of the original full-precision model. It requires a small amount of calibration data.
- **HQQ (Badri & Shaji, 2023)**: A calibration-free post-training quantization technique that minimizes errors in weights rather than layer output. It incorporates a sparsity-promoting loss, such as $l_{p<1}$-norm to model outliers through a hyper-Laplacian distribution.

E.2 Baselines

- **Vanilla (full-precision)**: Throughout this paper, we used the original full-precision model for each PTQ model as the *accuracy baseline*. This is because we assume a situation in which users cannot run billion-scale LLMs on their devices, even though they want to achieve high-accuracy inference. In other words, the target accuracy is that of the billion-scale full-precision model in such a case.
- **PTQ-only**: Throughout this paper, we used the standalone 4-bit PTQ model (to which iKVC is not applied) as the *throughput baseline*. This is because we assume a situation in which users cannot run billion-scale LLMs on their devices, even though they want to achieve high-accuracy inference. In other words, the target throughput is that of the standalone PTQ model in such a case.

E.3 Toy-model parameters for throughput estimation

For the Qwen3-8B throughput comparison in Figure 5, we plot predictions by the toy model in Section 5.2 at correction intervals $K \in \{1, 10, 100, 500\}$. We set $\mathrm{TPS}_{\mathrm{PTQ}}$ to the measured throughput of the corresponding standalone 4-bit, group-size-128 PTQ model: 46.360, 19.418, and 19.533 tokens/s for HQQ, GPTQ, and AutoRound, respectively. We estimate the CPU-resident full-precision weights as $W_{\mathrm{off}} = 12.27$ GiB, using an approximate 15.27-GiB BF16 weight footprint for Qwen3-8B and the 3-GiB VRAM budget assigned to the full-precision model. We use an illustrative effective CPU-to-GPU bandwidth range of $B_{\mathrm{eff}} \in [2.5, 4.0]$ GiB/s and its midpoint, 3.25 GiB/s, for the central prediction. This gives a parameter-transfer term $W_{\mathrm{off}}/B_{\mathrm{eff}}$ ranging from 3.07 to 4.91 seconds per correction. We approximate the combined full-precision computation and KV-cache replacement time as $T_{\mathrm{comp}}(K+1)+T_{\mathrm{KV}}(K+1) = 0.003(K+1)$ seconds. The toy model counts K tokens generated by the PTQ model and one bonus token per correction cycle. The bandwidth range and combined overhead coefficient are illustrative assumptions. They are not fitted to the measured iKVC throughputs.

F Additional results of experiments

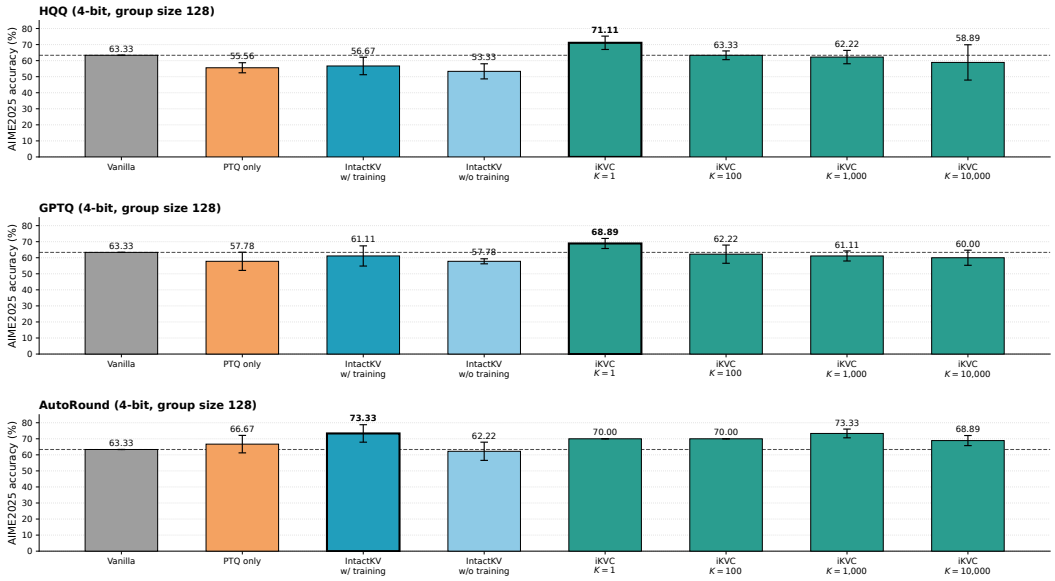

Figure 8: AIME2025 accuracy rates averaged over three runs for models quantized using HQQ for the first stage, AutoRound for the second stage, and GPTQ for the third stage. Vanilla (full-precision) refers to the accuracy baseline of the original full-precision model (Qwen3-4B-Thinking-2507), PTQ only refers to the standalone 4-bit PTQ model to which iKVC is not applied, IntactKV with/without training refers to the PTQ model to which IntactKV is applied with/without training, and iKVC refers to our framework with varying correction interval K from 1 to 10,000 tokens.

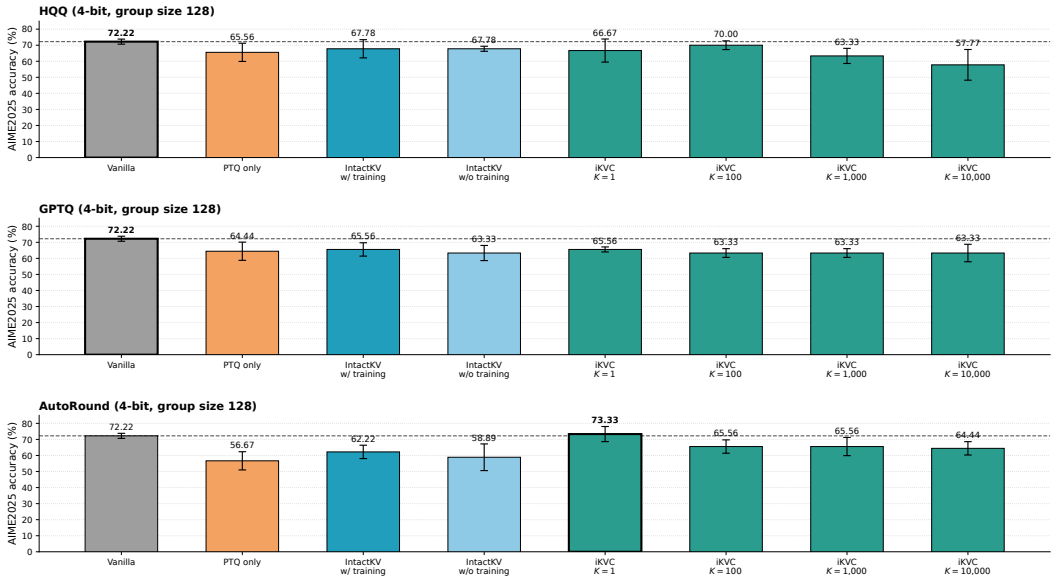

Figure 9: AIME2025 accuracy rates averaged over three runs for models quantized using HQQ for the first stage, AutoRound for the second stage, and GPTQ for the third stage. Vanilla (full-precision) refers to the accuracy baseline of the original full-precision model (Qwen3-14B), PTQ only refers to the standalone 4-bit PTQ model to which iKVC is not applied, IntactKV with/without training refers to the PTQ model to which IntactKV is applied with/without training, and iKVC refers to our framework with varying correction interval K from 1 to 10,000 tokens.