MIX-QUANT: QUANTIZED PREFILLING, PRECISE DECODING FOR AGENTIC LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents have recently emerged as a powerful paradigm for solving complex tasks through planning, tool use, memory retrieval, and multi-step interaction. However, these agentic workflows often introduce substantial input-side overhead, making the compute-intensive prefilling stage a key bottleneck in long-context, multi-turn inference. In this work, we propose Mix-Quant, a simple and effective phase-aware quantization framework for fast agentic inference. We first investigate FP4 quantization in agentic LLM workflows and observe that, despite being the dominant source of computation, prefilling exhibits substantially higher quantization redundancy than autoregressive decoding. Based on this insight, we apply high-throughput NVFP4 quantization to the prefilling phase while preserving BF16 precision for decoding. By decoupling prefilling acceleration from decoding quality, Mix-Quant combines phase-aware algorithmic quantization with hardware-efficient NVFP4 execution to alleviate the inference bottleneck in LLM agents. Extensive experiments across long-context and agentic benchmarks demonstrate that Mix-Quant largely preserves task performance while delivering significant improving efficiency, achieving up to a 3.74× speedup during prefilling.

1 INTRODUCTION

Large language model (LLM) agents have emerged as a powerful paradigm for solving complex real-world tasks involving tool use, memory retrieval, code generation, and multi-step interaction (Yao et al., 2022; Schick et al., 2023; Yang et al., 2024a; Xu et al., 2025). They have shown strong potential across coding agents, personal assistants, web agents, and general-purpose autonomous systems Luo et al. (2025). However, agentic workflows typically require repeated inference steps and multi-call interaction loops, leading to substantial context-processing overhead. In many cases, the input context can be tens to hundreds of times longer than the generated output, making the compute-intensive prefilling phase a major efficiency bottleneck in terms of both latency and throughput.

To alleviate the inference overhead, prior work has explored various model-efficiency strategies (Frantar et al., 2022a; Ma et al., 2023; Gu et al., 2024). While these methods are promising for improving deployment efficiency, aggressive compression may introduce non-negligible performance degradation. For example, post-training quantization (PTQ) is widely used due to its practicality. Weight-only PTQ (Frantar et al., 2022a; Lin et al., 2024) lowers memory footprint by representing model weights in low-bit formats such as INT4, improving throughput in memory-bound autoregressive decoding. However, because activations remain in high precision, it provides limited acceleration for the compute-bound prefill phase. In contrast, weight-and-activation quantization enables Xiao et al. (2023) low-bit matrix multiplications, directly reducing computational cost, but it can degrade performance, especially on complex long-trajectory tasks, due to error accumulation accumulates at each decoding step (Li et al., 2025).

Applying a uniform quantization strategy to both prefilling and decoding often leads to an unfavorable efficiency-performance trade-off. Prefilling and decoding serve different roles in LLM inference and exhibit distinct efficiency bottlenecks Zhong et al. (2024); Patel et al. (2024). More specically, prefilling processes a fixed input context and is compute-intensive, while decoding generates tokens autoregressively and is more sensitive to accumulated numerical errors. This distinction raises an important question: *Can we decouple prefilling and decoding, and optimize the model for each phase according to its distinct characteristics?*

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.

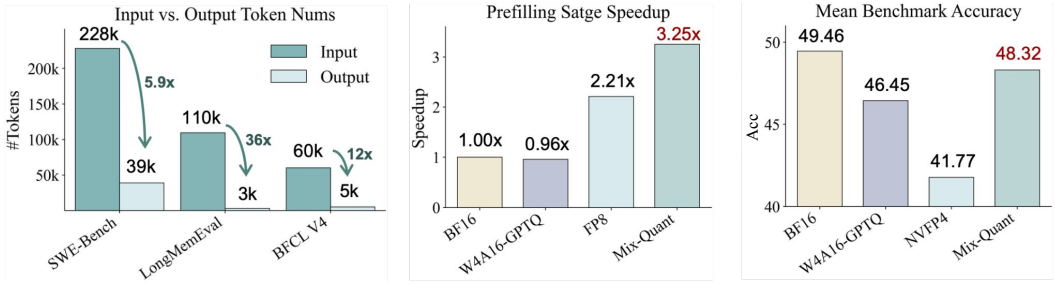


Figure 1: Agentic workflows are highly input-heavy, introducing substantial prefilling overhead. NVFP4 quantization can greatly accelerates computaion, but applying it to both prefilling and decoding causes notable accuracy degradation. Mix-Quant instead uses NVFP4 for prefilling and precise BF16 for decoding, achieving substantial speedup while largely preserving agentic performance.

Motivated by this, we propose Mix-Quant, a phase-aware quantization framework for efficient long-context LLM agentic inference. Mix-Quant applies high-throughput NVFP4 weight-and-activation quantization to the compute-intensive prefilling phase, while preserving BF16 precision for autoregressive decoding. NVFP4 precision (NVIDIA, 2025) is a microscaling FP4 format introduced with NVIDIA Blackwell, which uses fine-grained scaling to improve numerical accuracy at ultra-low bit-widths and provides native hardware support for efficient low-precision computation. The design of our method rests on two key observations: (1) Long-context, multi-turn agentic workflows introduce substantial context-processing overhead, making the compute-intensive prefilling phase a major efficiency bottleneck. Therefore, optimizing the prefilling phase is critical for efficient agentic inference; (2) Prefilling and decoding exhibit distinct computational bottlenecks and quantization redundancy behaviors. Prefilling processes a fixed input sequence in parallel and is suited to aggressive quantization: quantization errors do not recursively affect future inputs within the same prefill pass, and long agentic contexts often contain substantial redundancy. In contrast, decoding is much more error-sensitive, as each sampled token affect the generation process. Quantization errors can thus propagate and accumulate over long trajectories, ultimately degrading final task performance. By integrating high-throughput NVFP4 computation into prefilling while keeping decoding in precise BF16, Mix-Quant combines algorithm-level phase awareness with hardware-level acceleration, addressing the long-context processing bottleneck in agentic inference while preserving overall task performance.

We evaluate Mix-Quant on a comprehensive suite of long-context and agentic benchmarks, including two widely used long-context benchmarks (Bai et al., 2024; Artificial Analysis Team, 2025) and four multi-turn agentic benchmarks (Jimenez et al., 2023; Patil et al., 2025; Wu et al., 2025a; Barres et al., 2025), with state-of-the-art agentic base models (Google DeepMind, 2026; Qwen Team, 2026a). The results show that Mix-Quant consistently delivers substantial prefilling throughput gains, achieving up to 3.74× speedup, while largely preserving task performance across diverse long-context and agentic scenarios. These findings demonstrate that phase-aware quantization provides a favorable efficiency-performance trade-off for input-heavy LLM agentic inference.

Contributions. To summarize, our main contributions include: (1) We reveal that LLM agentic workflows are highly input-heavy due to multi-step interactions with environments, making the compute-intensive prefilling phase a major efficiency bottleneck in long-context agentic inference. Meanwhile, naive model-efficiency methods can hurt task performance, highlighting the need for phase-aware optimization. (2) We propose Mix-Quant, a phase-aware quantization framework that applies FP4 quantization to prefilling while retaining BF16 precision for autoregressive decoding, thereby improving efficiency without introducing severe error accumulation during generation. (3) We empirically show that Mix-Quant largely preserves agentic task performance while significantly improving inference efficiency, achieving up to a 3.74× prefill speedup over BF16 and demonstrating the potential of phase-aware model quantization for efficient and reliable LLM agents.

2 Related Work

Long-Context Agentic Workflows. LLM agents extend language models with action interfaces, external tools, memory, and feedback from interactive environments. ReAct introduced the pattern of

interleaving reasoning traces with environment actions (Yao et al., 2023), while Toolformer showed that language models can learn to invoke external APIs and condition on tool outputs (Schick et al., 2023). WebGPT demonstrated browser-assisted question answering (Nakano et al., 2021), MemGPT explored memory management for long-lived interactions (Packer et al., 2023), and SWE-agent showed the importance of agent-computer interfaces for software engineering (Yang et al., 2024b). These systems repeatedly call an LLM with prompts that include instructions, tool schemas, retrieved evidence, execution traces, and memory states. Recent work of agentic inference further emphasize substantial input-token overhead, repeated-context redundancy, and high serving cost (Wadlom et al., 2026; Wu et al., 2025b). Mix-Quant is motivated by this workload shift: for long-context agents, accelerating context processing is often as important as improving token generation throughput.

Prefill-Decode Disaggregation. Prefill and decode have different computational profiles. Prefill processes all prompt tokens in parallel and is dominated by large matrix multiplications, whereas decode advances autoregressively and repeatedly streams model weights and KV-cache entries. Serving systems have exploited this distinction by separating prompt processing from token generation. Splitwise maps prefill and decode to different machine configurations (Patel et al., 2023); DistServe disaggregates the phases across GPU pools to reduce interference between time-to-first-token and time-per-output-token objectives (Zhong et al., 2024); and other algorithmic approaches optimize long-context prompt processing through model transformations (Qiao et al., 2025) and dynamic sparse attention (Jiang et al., 2024a) for faster prefill . These works show that prefill should be treated as a distinct system-level workload. Mix-Quant is complementary: it makes the same distinction at the model-efficiency level by applying low-bit quantization only to the prefill path.

LLM Quantization. Quantization is widely used to reduce LLM inference cost (Zhou et al., 2024). Weight-only methods such as GPTQ (Frantar et al., 2022b) and AWQ (Lin et al., 2024) lower memory traffic and are effective for bandwidth-bound decoding, but provide limited speedup for long-context prefill because activations remain high precision and computation is not fully executed in low bit-widths. Weight-and-activation quantization can accelerate compute-bound prefilling, but applying aggressive W4A4 quantization to the full autoregressive process is brittle, as activation errors may perturb token choices and accumulate over generation (Dettmers et al., 2022; Xiao et al., 2023; Zhao et al., 2024). Mix-Quant therefore quantizes only context encoding while keeping decoding on the original high-precision path.

3 METHOD

LLM agentic workflows can effectively solve various complex real-world tasks through multi-round interactions with external environments, tools, and memory. However, such interaction-intensive workflows substantially increase the input context that must be processed at each inference step, leading to heavy inference costs. A naive application of model-efficiency techniques to accelerate inference often compromises overall task quality and destabilizes the generation process, especially in long agentic trajectories. To address this dilemma, we introduce a decoupled model-efficiency framework that applies FP4 quantization exclusively to high-throughput context prefilling while preserving high-precision decoding for stable and effective agentic generation.

In the remainder of this section, we first characterize the behaviour of long-context agentic workflows and identify their key efficiency bottlenecks. Next, we investigate FP4-quantized inference for both prefilling and decoding, with a particular focus on the error accumulation risks for long agentic trajectories. Finally, we detail our phase-aware quantization framework, which enables efficient and effective long-context agentic inference.

3.1 EFFICIENCY BOTTLENECKS IN LONG-CONTEXT AGENTIC WORKFLOWS.

LLM-based agentic workflows solve tasks through repeated model calls, tool invocations, environment observations, and memory updates. At each round, the prompt combines task instructions and tool descriptions with retrieved documents, previous actions, and execution results. As these records accumulate, the model repeatedly processes a growing context, often to generate only a short response, tool call, or action. As shown in fig. 1, input token counts can be tens of times larger than output token counts. This imbalance makes repeated context prefilling a major source of inference cost in long-context agentic workloads.

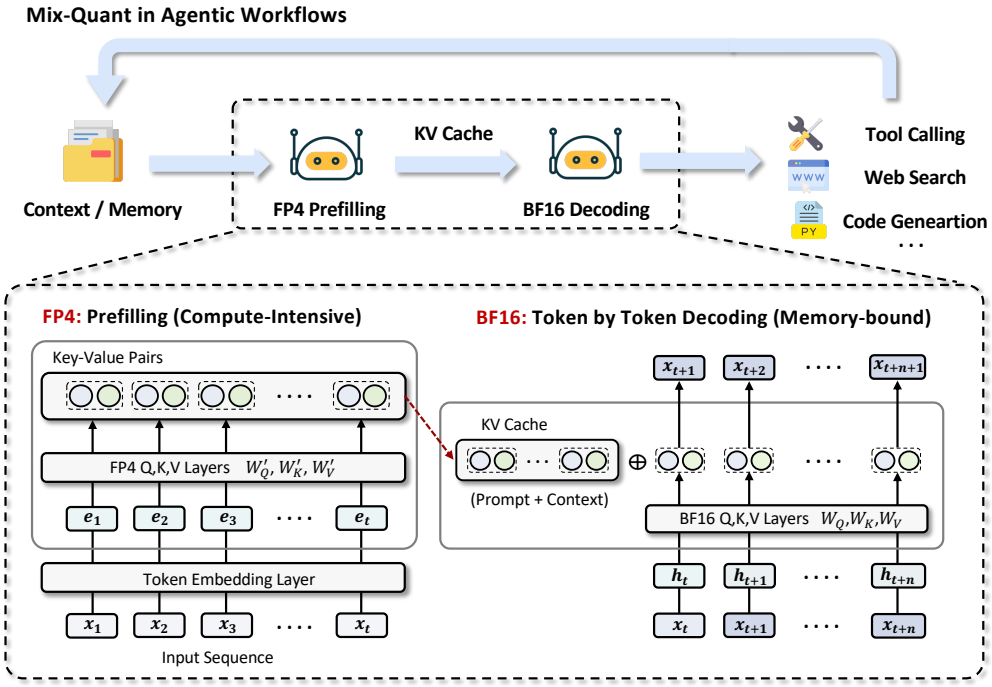


Figure 2: **Overview of Mix-Quant for efficient agentic LLM inference.** Agentic workflows repeatedly incorporate tool outputs, memory retrievals, and intermediate results into the input context, making the prefilling stage increasingly compute-intensive. Mix-Quant adopts a phase-aware quantization strategy: the context prefilling phase is accelerated with high-throughput NVFP4 computation, while autoregressive token-by-token decoding remains in BF16 to avoid error accumulation and preserve generation quality.

Prefilling and decoding impose distinct computational demands, as illustrated in fig. 2. Prefilling processes a fixed input context in parallel to produce hidden states and construct the KV cache. Its large-scale matrix multiplications typically make this stage compute-bound. Decoding generates tokens sequentially and is often constrained by memory traffic for model weights and the KV cache. Weight-only quantization methods (Frantar et al., 2022a; Lin et al., 2024) reduce weight storage and bandwidth requirements, primarily benefiting memory-bound decoding. However, prefilling still incurs substantial dense computation and dequantization overhead, limiting the speedups observed in our evaluation (fig. 1). Consequently, reducing weight precision alone does not fully address the prefill bottleneck in input-heavy agentic inference.

These observations motivate a phase-aware design that treats prefilling and decoding as distinct optimization targets. We study quantization as a concrete approach to accelerating compute-intensive context processing while preserving decoding quality. By allocating precision according to each stage's computational demands and accuracy requirements, this design targets the specific efficiency bottlenecks of long-context agentic workloads.

3.2 Error Accumulation Risks of Quantized Generation.

Model quantization is attractive for accelerating LLM inference because it can reduce memory usage and enable low-bit computation. In particular, applying weight and activation FP4 quantization to prefilling can reduce the cost of processing long input contexts, since the prefill phase is dominated by large matrix multiplications. However, naively applying FP4 quantization to the entire inference pipeline, including autoregressive decoding, can introduce significant quality degradation.

The key issue is that prefilling and decoding propagate quantization errors in different ways. During prefilling, the input context is fixed. Quantization errors may affect the hidden states and the

constructed KV cache, but they do not change the input tokens being processed. Therefore, the error introduced in prefilling is mainly a representation-level perturbation on a fixed context.

Moreover, long-context inputs often contain substantial redundancy (Jiang et al., 2024b). As shown in fig. 3, only a small set of heavy-hitter tokens dominates the attention mass at each decoding step. In our 128K-context setting, the top-4096 tokens, corresponding to only 3.125% of the full context, retain 95.8% of the total attention mass on average across layers and heads. This suggests that subsequent decoding is mainly influenced by a small subset of context tokens, while most tokens receive negligible attention and have limited impact on the next-token representation. The attention mass concentration further implies that prefill-stage KV errors are not simply accumulated over all context tokens. Since the attention output is a normalized weighted aggregation over cached values, quantization errors on low-attention tokens are attenuated by their small attention weights. As discussed in section A, the impact of such perturbations is governed by the attention-weighted effective context rather than the raw context length. Therefore, prefill quantization errors do not grow linearly or explosively with the context prefilling length, which helps explain the robustness of aggressive quantization during prefilling.

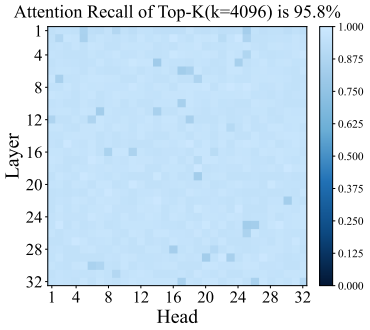


Figure 3: Attention mass concentration in a 128K-token context. The top 4,096 tokens, representing only 3.125% of the full 128K-token context, account for an average of 95.8% of the total attention mass. This suggests that long-context attention is highly concentrated on a small subset of tokens.

In contrast, decoding is a sequential decision process. At each step, the model predicts the next token based on all previous tokens:

$$y_t \sim p(y_t \mid x_{1:L}, y_{<t}). \tag{1}$$

When decoding is performed under quantized model, numerical perturbations can change the output distribution. Even a small change in the token distribution may lead to a different sampled or selected token. Once this happens, the generated sequence diverges from the high-precision trajectory, and all future predictions are conditioned on a different history. As a result, decoding errors can accumulate over time rather than remaining local. Previous work (Zhao et al., 2025; Li et al., 2025) also observe that token prediction changes can trigger snowball effect.

This risk is amplified in long agentic trajectories. A single erroneous token may produce an invalid tool call, select a wrong action, corrupt a code edit, or introduce an incorrect intermediate state. Such mistakes can then affect later observations and decisions, causing the agent to move further away from the correct solution path. Therefore, while aggresive FP4 quantization is well suited for accelerating the compute-intensive prefill phase, applying it to decoding can destabilize agentic generation.

These observations reveal a clear efficiency-quality trade-off. Quantizing prefilling can provide substantial efficiency gains with limited error propagation, whereas quantizing decoding may repeatedly inject errors into the autoregressive process. This motivates our phase-aware framework: use FP4 quantization for high-throughput context prefilling, but preserve high-precision computation for decoding.

3.3 Mix-Quant: Flash Prefilling, Precise Decoding

NVFP4 Microscaling Quantization for Prefilling. We adopt NVFP4 weight-and-activation quantization as the default configuration of Mix-Quant. To assess the generality of our phase-aware design, we additionally evaluate INT4 and FP8 (Table 4) and MXFP4 (Table 7). NVFP4 is a 4-bit microscaling floating-point format (NVIDIA, 2025) introduced for Blackwell-generation low-precision tensor-core execution. Each numerical value is represented by an E2M1 FP4 value, while groups of consecutive elements share a local scale. Unlike coarser microscaling formats such as MXFP4, which typically use larger groups and power-of-two scales, NVFP4 adopts smaller groups of 16 elements with FP8 E4M3 block scales, together with an additional tensor-level scale that controls the global dynamic range (Egiazarian et al., 2025). This two-level scaling is crucial: the tensor-level scale prevents global saturation, while the local block scale adapts to fine-grained variations within

the tensor. Moreover, due to its small group size and fine-grained scaling design, NVFP4 already achieves strong quantization performance with simple round-to-nearest (RTN) quantization, while more complex quantization techniques such as rotation provide little additional benefit and introduce extra runtime overhead (Egiazarian et al., 2025). Therefore, we directly adopt RTN quantization in our implementation.

Let $x \in \mathbb{R}^n$ be a vectorized activation or weight tensor, and let $\mathcal{B}$ be a partition of its elements into blocks of size $g = 16$. For an element x_i in block $b(i)$, NVFP4 quantization can be written as

$$q_i = \Pi_{\mathrm{FP4}}\left(\frac{x_i}{\alpha_x\, \sigma_{b(i)}}\right), \qquad \hat{x}_i = \alpha_x\, \sigma_{b(i)} q_i, \tag{2}$$

where $\Pi_{\mathrm{FP4}}(\cdot)$ projects a scaled value to the nearest representable FP4 value with clipping, σ_b is the FP8 block scale for block b, and α_x is the tensor-level scale. A standard amax-based choice sets the block scale according to the largest magnitude in the block,

$$\sigma_b = \Pi_{\mathrm{E4M3}}\left(\frac{\max_{i \in b} |x_i|}{\alpha_x\, q_{\max}}\right), \tag{3}$$

where $q_{\max}$ is the largest finite FP4 magnitude and $\Pi_{\mathrm{E4M3}}(\cdot)$ rounds the scale to the FP8 E4M3 grid. In practice, activations and weights use layouts aligned with the GEMM dimension so that quantization, dequantization, and matrix multiplication can be fused efficiently by the backend.

Prefill-Decode Disaggregation Deployment. At inference time, Mix-Quant maintains two execution paths for the same base model: an NVFP4 W4A4 prefill path and the original high-precision decode path. We deploy these two paths using a prefill-decode disaggregation framework, where prefill workers process the input prompt and transfer the resulting KV cache to decode workers through a NIXL-based KV-cache transfer mechanism (Kwon et al., 2023a). Given a prompt, the quantized prefill path processes all input tokens and writes the initial KV cache in the precision expected by the high-precision decode engine. The decode path then consumes this cache and generates output tokens autoregressively, while KV entries for newly generated tokens are produced by the high-precision decode path as usual, as shown in fig. 2. This system-level design avoids the extra low-level kernel switching, conversion overhead, and KV-cache misalignment that can arise in mixed-precision quantization pipelines, while preserving the deployment benefits of prefill-decode disaggregation (Zhong et al., 2024; Patel et al., 2023).

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate Mix-Quant on a diverse suite of input-intensive benchmarks covering long-context reasoning and agentic inference. For long-context evaluation, we use LongBench-V2 (Bai et al., 2024) and AA-LCR (Artificial Analysis Team, 2025), which test understanding, synthesis, and reasoning over long documents. For agentic evaluation, we consider BFCL v4 (Patil et al., 2025) for tool use and function calling, LongMemEval (Wu et al., 2025a) for long-term interactive memory, τ^2-bench (Barres et al., 2025) for stateful interactive conversations as general agents. In addition, we conduct a minimal complementary evaluation on SWE-Bench Verified (Jimenez et al., 2023; OpenAI, 2024), using mini-SWE-agent (Yang et al., 2024b) for software-engineering agents. Since SWE-Bench Verified involves long agent trajectories and is particularly time-consuming to evaluate, we select one representative model setting for this experiment in section B.1 to validate whether Mix-Quant preserves agentic software-engineering capability. To provide a more comprehensive assessment, we further evaluate Mix-Quant on challenging reasoning benchmarks, including Math500 (Lightman et al., 2023), AIME24 and AIME25 (Dekoninck et al., 2026).

Models. We evaluate recent open-weight agentic base or instruction-tuned models spanning multiple model families and scales: Qwen3-8B (Yang et al., 2025), Gemma-4-26B-A4B-it, Gemma-4-31B-it (Google DeepMind, 2026), and Qwen3.5-9B (Qwen Team, 2026a). These models are selected because they support long-context or agent-oriented use cases and cover both compact and larger-capacity deployment regimes. We also evaluate Qwen3.8-27B (Qwen Team, 2026b) to further assess generalization, with results reported in Appendix B.2.

Table 1: Agentic benchmark performance of Mix-Quant across general, long-term memory, and stateful interaction benchmarks.

Model	BFCL v4	LongMemEval	τ^2-bench	Avg.
Qwen3-8B	40.50	**57.00**	**31.06**	**42.85**
Qwen3-8B-NVFP4	38.77	49.82	27.34	38.64
Mix-Quant	**40.63**	54.85	28.86	41.45
Qwen3.5-9B	**72.68**	**86.27**	**86.69**	**81.88**
Qwen3.5-9B-NVFP4	69.90	78.00	76.26	74.72
Mix-Quant	71.13	84.27	81.89	79.10
Gemma-4-26B-A4B-it	**62.42**	**80.50**	**64.63**	**69.18**
Gemma-4-26B-A4B-it-NVFP4	60.18	62.42	57.31	59.97
Mix-Quant	61.72	72.45	60.62	64.93
Gemma-4-31B-it	**76.92**	**90.80**	**73.50**	**80.41**
Gemma-4-31B-it-NVFP4	76.33	89.20	70.74	78.76
Mix-Quant	76.19	90.40	72.84	79.81

Evaluation Setup. We serve FP4 models on RTX 5090 and B200 GPUs to use Blackwell-generation FP4 hardware acceleration. The serving stack is based on vLLM (Kwon et al., 2023b). For disaggregated execution, we use NIXL-based KV-cache transfer between prefill and decode workers, following the standard disaggregated-prefill serving pattern in which compute-intensive prefill and memory-bandwidth-intensive decode can be placed on separate workers (NVIDIA, 2026). Each benchmark is run independently three times and we report the mean score. We use the default maximum context length exposed by the model configuration for Gemma-4 and Qwen3.5 models, namely 256K and 262K tokens, respectively. We extend Qwen3-8B's native 32K context window to 131K tokens using YaRN (Peng et al., 2024) with a scaling factor of 4.

4.2 Main Results

Long-Context Agentic Benchmark Results. Table 1 reports the results on long-context agentic benchmarks. Overall, almost all quantized baselines suffer noticeable performance degradation, indicating that maintaining accuracy under quantization is particularly challenging for complex long-context agentic workloads. In particular, uniform NVFP4 leads to a large average performance drop, showing that directly applying low-precision quantization to all inference phases can significantly harm agentic reasoning and long-context decision making.

In contrast, Mix-Quant largely preserves the performance of the high-precision model and achieves higher average agentic benchmark scores than uniform NVFP4 for all four evaluated models, highlighting the importance of keeping the decoding phase in high precision. Interestingly, we also observe that quantizing only the prefill phase can achieve better performance than the W4A16 baseline in section 4.3, suggesting that preserving high-precision autoregressive decoding is more critical than uniformly increasing activation precision across both phases. This trend is consistent with the phase-aware motivation of Mix-Quant: long-memory and tool-use workloads require efficient processing of large input contexts, but their autoregressive decisions remain highly sensitive to quantization-induced perturbations during decoding.

Reasoning and Long-Context Benchmark Results. To complement the agentic evaluation, we further evaluate MixQuant on reasoning and long-context benchmarks, with the results summarized in Table 2. These benchmarks assess whether the benefits of MixQuant generalize beyond agentic workloads to tasks involving multi-step reasoning and long-context understanding. As shown in Table 2, MixQuant consistently recovers a large portion of the accuracy loss introduced by uniform NVFP4 quantization across these tasks. This shows that phase-aware quantization is not only beneficial for agentic serving, but also effective for workloads involving multi-step reasoning and long-context understanding.

Table 2: Reasoning and long-context performance of Mix-Quant on mathematical reasoning and long-context benchmarks.

Model	Reasoning			Long Context		Avg.
	MATH500	AIME24	AIME25	LongBench-V2	AA-LCR	
Qwen3-8B	93.73	75.54	**67.77**	**39.86**	**33.67**	**62.11**
Qwen3-8B-NVFP4	94.12	66.53	55.33	35.46	24.67	55.22
Mix-Quant	**94.40**	**76.66**	66.66	38.60	28.67	61.00
Qwen3.5-9B	**94.85**	68.89	**60.00**	**55.47**	**81.00**	**72.04**
Qwen3.5-9B-NVFP4	93.46	54.44	40.00	50.40	78.00	63.26
Mix-Quant	94.32	**70.33**	56.67	52.29	79.33	70.59
Gemma-4-26B-A4B-it	**95.86**	77.67	65.33	**53.83**	**67.00**	**71.94**
Gemma-4-26B-A4B-it-NVFP4	95.20	75.33	62.22	48.15	50.67	66.31
Mix-Quant	95.40	**78.67**	**69.67**	51.57	64.33	71.93
Gemma-4-31B-it	97.33	92.22	**82.22**	**63.35**	**76.67**	**82.36**
Gemma-4-31B-it-NVFP4	**97.75**	83.33	80.00	58.55	71.67	78.26
Mix-Quant	97.20	**93.33**	81.11	61.64	73.67	81.39

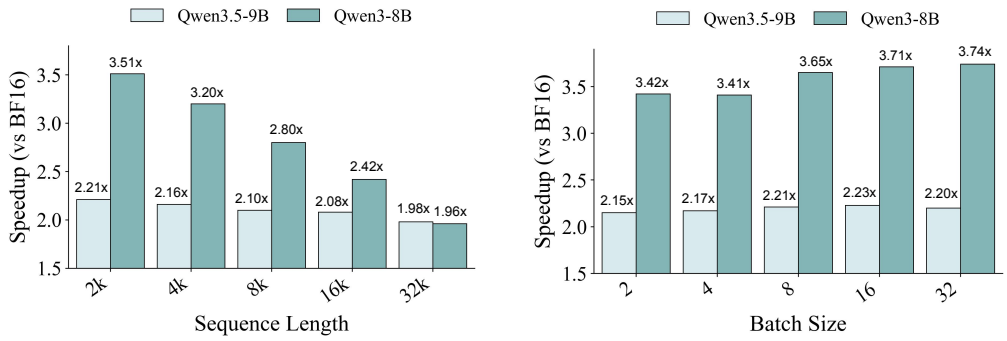


Figure 4: End-to-end prefill latency speedup of MixQuant over the BF16 baseline on NVIDIA RTX 5090 GPUs. Left: speedup across different sequence lengths with batch size fixed to 1. Right: speedup across different batch sizes with sequence length fixed to 2K.

Prefilling Stage Speedup. We evaluate the end-to-end prefill latency speedup of MixQuant over the BF16 baseline on NVIDIA RTX 5090 GPUs. All latency measurements are performed in vLLM, using FlashInfer for attention computation and Blackwell NVFP4 W4A4 GEMM kernels for linear layers. As shown in fig. 4, MixQuant consistently accelerates prefill across both Qwen3.5-9B and Qwen3-8B under varying sequence lengths and batch sizes, achieving speedups ranging from 1.96× to 3.74× across the evaluated settings. These results show that aggressive NVFP4 weight-and-activation quantization can substantially reduce the computation cost of input-intensive prefill workloads, highlighting MixQuant's potential for efficient agentic LLM serving.

4.3 Ablations and Analysis

Phase-wise Quantization Ablation. Table 3 compares FP4-prefill/BF16-decode with BF16-prefill/FP4-decode to examine stage-specific quantization sensitivity. Despite the input-heavy workloads, quantizing prefill achieves higher average scores than quantizing decoding on both models. This finding is consistent with their different error-propagation mechanisms: prefill encodes a fixed context, whereas decoding perturbations can alter token decisions and affect subsequent generation. These results reinforce the phase-aware design of Mix-Quant, which aligns precision allocation with the distinct computational bottlenecks and quantization sensitivities of prefilling and decoding.

Table 3: Phase-wise quantization ablation comparing the impact of quantizing different inference stages on long-context and multi-turn agentic benchmarks.

Model	Long Context		Agentic			Avg.
	LongBench-V2	AA-LCR	BFCL v4	LongMemEval	τ^2-bench	
Qwen3-8B	**39.86**	**33.67**	40.50	**57.00**	**31.06**	**40.42**
Qwen3-8B-NVFP4	35.46	24.67	38.77	49.82	27.34	35.21
Qwen3-8B-P16D4	37.38	29.33	36.40	52.47	28.13	36.74
Mix-Quant	38.60	28.67	**40.63**	54.85	28.86	38.32
Gemma-4-26B-A4B-it	**53.83**	**67.00**	**62.42**	**80.50**	**64.63**	**65.68**
Gemma-4-26B-A4B-it-NVFP4	48.15	50.67	60.18	62.42	57.31	55.75
Gemma-4-26B-A4B-it-P16D4	50.72	63.33	62.22	71.62	61.27	61.83
Mix-Quant	51.57	64.33	61.72	72.45	60.62	62.14

Table 4: Performance of Mix-Quant with FP8 and INT4 quantization. Mix-Quant applies the indicated format during prefilling and retains BF16 precision during decoding.

Method	BFCL v4	LongMemEval	LongBench-V2	AA-LCR	Avg.
Qwen3-8B					
BF16	**40.50**	**57.00**	**39.86**	33.67	**42.76**
FP8	34.56	55.40	38.97	31.67	40.15
Mix-Quant (FP8)	36.70	55.53	38.30	35.67	41.55
INT4	35.64	53.13	37.30	31.33	39.35
Mix-Quant (INT4)	37.85	55.13	38.36	**36.33**	41.92
Gemma-4-26B-A4B-it					
BF16	62.42	80.50	53.83	67.00	65.94
FP8	62.32	**80.80**	52.75	69.67	66.39
Mix-Quant (FP8)	**63.42**	80.20	**54.08**	**70.33**	**67.01**
INT4	57.96	69.40	49.10	62.00	59.62
Mix-Quant (INT4)	61.62	77.00	52.02	66.33	64.24

Other Quantization Formats. To evaluate the generality of Mix-Quant beyond NVFP4, we extend its phase-aware quantization strategy to FP8 and INT4. For each format, we quantize prefilling while retaining BF16 decoding and compare this configuration with quantization applied to both stages. Table 4 reports the results on Qwen3-8B and Gemma-4-26B-A4B-it across agentic and long-context benchmarks. This comparison complements the phase-wise ablation in Table 3, where NVFP4 prefilling with BF16 decoding achieves higher average scores than the reverse allocation on both models. Across both formats, Mix-Quant achieves higher average scores than the corresponding baselines quantized in both stages. These results support the generality of phase-aware precision allocation beyond NVFP4 and reinforce the benefit of treating prefilling and decoding separately. We further evaluate Mix-Quant with MXFP4 quantization in Appendix B.3, extending the evaluation to another microscaling FP4 format.

5 Conclusion

We presented Mix-Quant, a phase-aware quantization framework for long-context agentic LLM inference. Motivated by the distinct computational demands and quantization sensitivities of prefilling and decoding, Mix-Quant applies FP4 weight-and-activation quantization to prefilling while retaining BF16 decoding. This design targets the cost of repeatedly processing long contexts while limiting quantization-induced disruption of autoregressive generation. Experiments on agentic and long-context benchmarks show that Mix-Quant largely preserves task performance while substantially accelerating prefilling. These findings support phase-aware precision allocation as a practical approach to efficient agentic inference.

AI USE STATEMENT

We used generative AI tools to assist with manuscript preparation, including translating author-provided notes, drafting and polishing text, and improving presentation. Additional assistance included formatting LaTeX tables, calculating arithmetic means from author-provided benchmark scores, and checking textual claims against the reported results. The authors are responsible for verifying all AI-assisted content and for the accuracy, originality, and integrity of the final manuscript. The authors reviewed and revised the retained AI-assisted text and independently checked its supporting references and calculations.

ETHICS STATEMENT

This work studies LLM inference efficiency using existing models and established benchmarks. Our evaluation focuses on task performance and inference efficiency; it does not establish the safety of the underlying models or their downstream applications. Lower inference costs may facilitate both beneficial and harmful uses of LLM agents, while quantization can alter model behavior beyond the metrics evaluated here. Deployment should therefore include application-specific safety evaluation, appropriate controls on tool access, and human oversight where needed.

REPRODUCIBILITY STATEMENT

Section 3.3 describes the quantization scheme and prefill–decode execution pipeline, including scale computation and KV-cache transfer. Section 4.1 identifies the evaluated models and benchmarks and specifies the hardware, serving stack, context-window configuration, and repeated-run protocol. The prefill latency measurement settings are described in Section 4.2. Appendix A provides the theoretical analysis of prefill perturbations, while Appendix B reports additional benchmark results.

REFERENCES

Artificial Analysis Team. Artificial Analysis Long Context Reasoning Benchmark (LCR). https://artificialanalysis.ai/, 2025. Accessed: 2026-05-06.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvalddson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling fp4 quantization, 2025. URL https://arxiv.org/abs/2509.23202.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022a.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022b.

Google DeepMind. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4, 2026. Accessed: 2026-05-06.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The twelfth international conference on learning representations*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024a.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024b.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023a.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023b.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring llm low-bit quantization degradation for mathematical reasoning. *arXiv preprint arXiv:2501.03035*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

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. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

NVIDIA. Introducing nvfp4 for efficient and accurate low-precision inference. `https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/`, 2025.

NVIDIA. Enhancing distributed inference performance with the nvidia inference transfer library, 2026. Accessed 2026-05-03.

OpenAI. Introducing swe-bench verified. `https://openai.com/index/introducing-swe-bench-verified/`, 2024. Accessed 2026-05-03.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. *arXiv preprint arXiv:2311.18677*, 2023.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024.

Aurick Qiao, Zhewei Yao, Samyam Rajbhandari, and Yuxiong He. Swiftkv: Fast prefill-optimized inference with knowledge-preserving model transformation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25745–25764, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL `https://qwen.ai/blog?id=qwen3.5`.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026b. URL `https://qwen.ai/blog?id=qwen3.8`.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Noppanat Wadlom, Junyi Shen, and Yao Lu. Efficient llm serving for agentic workflows: A data systems perspective. *arXiv preprint arXiv:2603.16104*, 2026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025a.

Haoran Wu, Can Xiao, Jiayi Nie, Xuan Guo, Binglei Lou, Jeffrey TH Wong, Zhiwen Mo, Cheng Zhang, Przemyslaw Forys, Chengyang Ai, et al. Combating the memory walls: Optimization pathways for long-context agentic llm inference. *arXiv preprint arXiv:2509.09505*, 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. In *International Conference on Machine Learning*, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024a.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. Qspec: Speculative decoding with complementary quantization schemes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4779–4795, 2025.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *arXiv preprint arXiv:2310.19102*, 2024.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation*, 2024.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

A Theoretical Justification

Attention concentration mitigates prefill KV errors accumulation. The concentration of attention mass provides a theoretical explanation for why prefill-stage KV perturbations are often tolerable. Consider one attention head during decoding. Given the query q_t and the prefetched keys and values $\{k_i, v_i\}_{i=1}^{L}$, the attention output is

$$o_t = \sum_{i=1}^{L} a_{t,i} v_i, \quad a_{t,i} = \mathrm{softmax}\left(\frac{q_t k_i^\top}{\sqrt{d}}\right). \tag{4}$$

After prefill quantization, the cached values become $\hat{v}_i = v_i + \Delta v_i$. If we first consider the value perturbation under fixed attention weights, the induced output error is

$$\Delta o_t^{(v)} = \sum_{i=1}^{L} a_{t,i} \Delta v_i. \tag{5}$$

Thus, the contribution of each KV error is weighted by its attention probability. For a top-K attention set

$$\mathcal{S}_K = \mathrm{TopK}_i(a_{t,i}), \tag{6}$$

the perturbation from the remaining low-attention tokens is bounded by

$$\begin{aligned} \left\| \sum_{i \notin \mathcal{S}_K} a_{t,i} \Delta v_i \right\| &\le \sum_{i \notin \mathcal{S}_K} a_{t,i} \left\| \Delta v_i \right\| \\ &\le (1 - R_K)\, \delta_v, \end{aligned} \tag{7}$$

where $R_K = \sum_{i \in \mathcal{S}_K} a_{t,i}$ is the top-K attention recall and $\delta_v = \max_i \|\Delta v_i\|$. Therefore, the error from the low-attention part is controlled by its total attention mass, rather than by the number of tokens it contains.

In our 128K-context setting, the top-4096 tokens account for only 3.125% of all context positions but retain $R_K = 95.8\%$ of the attention mass. Hence, the remaining 96.875% of tokens collectively receive only $1 - R_K = 4.2\%$ attention mass. Under bounded KV perturbations, their aggregate contribution to the attention output is therefore suppressed by a factor of 0.042. This shows that prefill KV errors do not accumulate linearly with the raw context length L; instead, they are filtered through the attention distribution.

A similar intuition also applies to key perturbations. Let $\hat{k}_i = k_i + \Delta k_i$ and denote the resulting attention weights by $\hat{a}_{t,i}$. The quantized attention output can be written as

$$\hat{o}_t = \sum_{i=1}^{L} \hat{a}_{t,i} \hat{v}_i. \tag{8}$$

The total output perturbation can be decomposed as

$$\begin{aligned} \hat{o}_t - o_t = &\sum_{i=1}^{L} a_{t,i} \Delta v_i + \sum_{i=1}^{L} \left(\hat{a}_{t,i} - a_{t,i}\right) v_i \\ &+ \sum_{i=1}^{L} \left(\hat{a}_{t,i} - a_{t,i}\right) \Delta v_i. \end{aligned} \tag{9}$$

The first term is the attention-weighted value error, while the second term captures the change in attention weights caused by key perturbations. For small key perturbations, the first-order change of the attention weights satisfies

$$\Delta a_{t,i} \approx a_{t,i} \left(\Delta z_i - \sum_{j=1}^{L} a_{t,j} \Delta z_j \right), \quad \Delta z_i = \frac{q_t \Delta k_i^\top}{\sqrt{d}}. \tag{10}$$

This expression again shows that the effect of key perturbations is scaled by $a_{t,i}$. Perturbations on low-attention tokens have limited influence on the attention distribution, while errors on high-attention tokens dominate the output change.

Therefore, long-context redundancy reduces the effective sensitivity of the prefill stage to quantization noise. Although the KV cache contains L entries, the attention computation uses them through a normalized weighted aggregation. When the attention mass is concentrated on a small subset of tokens, quantization errors on the majority of low-attention KV entries are naturally attenuated. This contrasts with decoding-stage errors, where a small perturbation can alter a discrete token decision and affect all subsequent predictions.

B Additional Results

B.1 Results on SWE-Bench-Verified

As a complementary evaluation, we further report results on SWE-Bench-Verified (Jimenez et al., 2023) using mini-SWE-agent (Yang et al., 2024a). Since this benchmark is large-scale and involves particularly long agent trajectories, running the full suite across all models is computationally expensive. Therefore, we select one representative model setting for this validation. The results provide additional evidence that Mix-Quant preserves agentic software-engineering capability under challenging long-horizon execution settings.

B.2 Results on Qwen3.8-27B

To further assess the applicability of Mix-Quant to recent agentic models, we evaluate Qwen3.8-27B, an open-weight model designed for complex, multi-step tasks (Qwen Team, 2026b). We compare the original BF16 model, uniform NVFP4 quantization, and Mix-Quant on AA-LCR, LongMemEval, and LongBench-V2. Mix-Quant uses NVFP4 during prefilling and retains BF16 decoding, following the phase-aware design in the main experiments. As shown in Table 6, Mix-Quant achieves a higher average score than uniform NVFP4 quantization. This result extends the empirical support for phase-aware precision allocation to Qwen3.8-27B.

Table 5: Additional results on SWE-Bench-Verified evaluated with mini-SWE-agent for software-engineering agents.

Model	SWE-Bench-Verified
Qwen3-8B	**10.00**
Qwen3-8B-INT4	8.00
Qwen3-8B-NVFP4	6.00
Mix-Quant	**10.00**

Table 6: Performance of Qwen3.8-27B on long-context and long-term memory benchmarks. NVFP4 denotes quantization applied to both prefilling and decoding; Mix-Quant combines NVFP4 prefilling with BF16 decoding. Results are averaged over three independent runs. Avg. denotes the arithmetic mean across the three benchmarks.

Method	AA-LCR	LongMemEval	LongBench-V2	Avg.
Qwen3.8-27B				
BF16	82.33	**91.67**	**67.41**	**80.47**
NVFP4	79.67	89.43	63.76	77.62
Mix-Quant	**83.00**	90.56	65.97	79.84

B.3 Results with MXFP4 Quantization

We further evaluate Mix-Quant with MXFP4, a 4-bit floating-point format defined by the Open Compute Project (OCP) microscaling specification (Rouhani et al., 2023). Like NVFP4, MXFP4 represents individual values in E2M1, but shares an E8M0 power-of-two scale across blocks of 32 elements. NVFP4 instead uses 16-element blocks with E4M3 scales and an additional tensor-level scale (NVIDIA, 2025). By sharing each scale across more elements, MXFP4 reduces block-scale storage overhead, while its power-of-two scales allow rescaling through exponent adjustment. However, its larger blocks and coarser scale representation provide less flexibility for adapting to local tensor distributions, potentially increasing quantization error.

We evaluate MXFP4 on Qwen3-8B and Qwen3.8-27B using AA-LCR, LongMemEval, and LongBench-V2. For each model, we compare BF16, uniform MXFP4 quantization, and Mix-Quant with MXFP4 prefilling and BF16 decoding. As shown in Table 7, Mix-Quant achieves higher average scores than uniform MXFP4 quantization on both models. These results support the applicability of phase-aware precision allocation to MXFP4, extending the evidence beyond NVFP4.

Table 7: Performance of Mix-Quant with MXFP4 quantization on long-context and long-term memory benchmarks. MXFP4 denotes quantization applied to both prefilling and decoding; Mix-Quant combines MXFP4 prefilling with BF16 decoding. Results are averaged over three independent runs. Avg. denotes the arithmetic mean across the three benchmarks.

Method	AA-LCR	LongMemEval	LongBench-V2	Avg.
Qwen3-8B				
BF16	**33.67**	**57.00**	**39.86**	**43.51**
MXFP4	19.67	40.24	33.97	31.29
Mix-Quant	29.67	51.13	39.30	40.03
Qwen3.8-27B				
BF16	82.33	**91.67**	**67.41**	**80.47**
MXFP4	81.67	88.42	62.33	77.47
Mix-Quant	**82.67**	90.13	65.24	79.35

C LIMITATIONS AND FUTURE WORK

This work focuses on quantization as a first exploration of phase-aware model efficiency for agentic LLM inference. Mix-Quant shows that prefilling and decoding have different efficiency bottlenecks and quantization redundancy, motivating separate optimization strategies for each phase. However, our current method is limited to low-precision quantization, while the same principle may be extended to other model efficiency techniques, such as sparsity, pruning, or speculative decoding.

Future work could combine prefill-oriented acceleration with more robust decoding-side techniques represents a promising direction for future work, as it may further improve end-to-end efficiency for long-trajectory LLM agents.