GRACE: GRADED REPRESENTATION AUGMENTATION FOR COST-EFFECTIVE SLM–LLM COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) provide strong generation capabilities but incur substantial inference cost, while small language models (SLMs) offer efficient decoding at the expense of reduced performance on complex tasks. Recent studies have begun to explore ways of combining their complementary strengths, including speculative decoding and model switching. These efforts show the potential of involving stronger LLMs to assist SLM generation, but such assistance may still incur considerable additional computation and decoding latency. We propose **GRACE**, a **G**raded **R**epresentation **A**ugmentation framework for **C**ost-**E**ffective SLM–LLM collaboration, which improves SLM generation quality with minor additional LLM computation. GRACE establishes asynchronous bidirectional latent interaction between the frozen SLM and LLM, enabling LLM assistance without interrupting the SLM's autoregressive decoding. During decoding, a context aligner incrementally compresses the evolving SLM state into compact latent tokens to synchronize with the LLM, while representation augmentation blocks extract latent task-relevant guidance from the asynchronously available LLM representations to augment the SLM. These augmentations are performed at depth-aligned computation grades, allowing subsequent SLM layers to progressively absorb the guidance. Extensive experiments on instruction-following and mathematical benchmarks show that GRACE consistently improves SLM generation quality with favorable performance–efficiency trade-offs across different model scales, architectures, and versions, and remains effective when the collaborating models come from different model families.

1 INTRODUCTION

The rapid progress of large language models (LLMs) has substantially advanced the frontier of language understanding and generation (Guo et al., 2025; Chen et al., 2026; Zhao et al., 2026), yielding increasingly capable models across instruction following (Xu et al., 2026), mathematical reasoning (Yu et al., 2026), code generation (Singh et al., 2025), and a broad range of complex tasks. These advances have been driven by the continued scaling of training data and model capacity (Hoffmann et al., 2022), together with increasingly effective post-training techniques (Guo et al., 2025). However, these growing capabilities of LLMs come at a substantial inference cost. Autoregressive decoding proceeds sequentially, requiring token-by-token model computation. This incurs substantial computational and memory overhead, increasing serving cost, resource consumption, and decoding latency, which makes it difficult to fully exploit LLM capabilities in practical deployment, particularly in settings with constrained compute, memory, or latency budgets.

A natural way to reduce inference cost is to shift decoding toward smaller language models (SLMs). Distilled small language models can inherit part of the knowledge and capabilities of the corresponding LLM while enabling more efficient autoregressive decoding (Agarwal et al., 2024). Nevertheless, their performance often degrades on challenging tasks, motivating early explorations to exploit the complementary strengths of small and large language model (SLM-LLM) collaboration. As illustrated in Figure 1(a), speculative decoding (Li et al., 2024b;c; 2026) establishes a draft-then-verify paradigm, where a lightweight SLM rapidly proposes candidate tokens and a target LLM verifies them in parallel. This parallel verification can effectively accelerate target-LLM decoding, but requires additional computation to verify the drafted tokens, resulting in higher overall computational cost than standard target model decoding.

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.

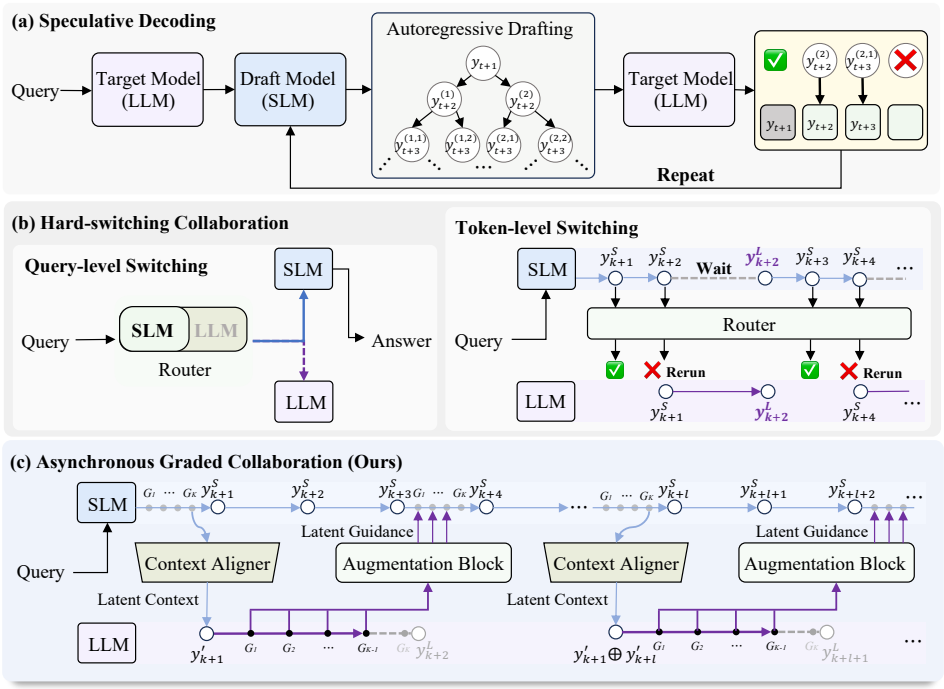


Figure 1: **Comparison of SLM-LLM collaboration paradigms.** Speculative decoding uses a draft–then-verify loop, while hard switching methods route generation to either the SLM or LLM at the query or token level. In contrast, we keep the SLM as the uninterrupted autoregressive decoder and enable asynchronous bidirectional latent interaction with the LLM.

Beyond speculative decoding, recent studies have begun to explore direct collaboration among independently frozen pretrained models of different scales through hard switching, as illustrated in Figure 1(b). Query-level approaches (Ding et al., 2024; Ong et al., 2025) route each input to either the SLM or LLM before generation, after which the selected model decodes the complete response. Token-level approaches (Zheng et al., 2025; Fu et al., 2026a) instead make routing decisions during autoregressive decoding. The SLM generates tokens sequentially, while a router determines whether each decoding step is accepted or rerun by the LLM. These approaches provide flexible ways to allocate stronger LLM computation according to input or decoding difficulty. However, model switching during generation may require additional context synchronization and force the SLM to wait for LLM execution, introducing extra computation and latency. A key challenge therefore lies in improving SLM generation quality with stronger LLM while preserving its native decoding efficiency through non-blocking asynchronous collaboration.

To address this challenge, we propose **GRACE**, a **G**raded **R**epresentation **A**ugmentation framework for **C**ost-**E**ffective SLM–LLM Collaboration, as illustrated in Figure 1(c). GRACE establishes asynchronous bidirectional latent interaction between two frozen models, allowing the LLM to assist the SLM without interrupting its autoregressive decoding. Considering that LLM representations encode rich task-relevant information beyond mere token outputs (Hanna & Ameisen, 2026), GRACE introduces representation augmentation blocks that transform asynchronously available LLM representations into latent guidance for augmenting the SLM. To exploit information encoded at different stages of computation, we partition both models into depth-aligned computation grades and perform representation augmentation at each aligned stage. This graded design allows guidance introduced at earlier stages to be progressively integrated by subsequent SLM layers, while deeper grades provide complementary refinement. Meanwhile, a context aligner incrementally synchronizes the evolving SLM context to the LLM in the latent space by compressing successive windows of SLM representations into LLM-readable latent tokens, effectively aligning their contexts with minor additional LLM computation. Through these designs, the SLM directly benefits from stronger LLM representations without any waiting overhead, achieving a highly favorable performance–efficiency trade-off.

Extensive evaluations on both instruction-following and mathematical reasoning benchmarks demonstrate that GRACE consistently improves SLM generation quality and strengthens its ability to handle complex tasks. It enhances precise interpretation of complex instructions and satisfaction of multifaceted constraints, while promoting coherent mathematical reasoning along more direct solution trajectories. Notably, these improvements are robust across diverse SLM–LLM configurations, spanning model scales, dense and MoE architectures, and intra-family versions. GRACE also remains effective across different model families, indicating that the latent interaction does not rely on closely matched backbone architectures or tokenization spaces. Ultimately, by guiding the SLM toward more effective problem-solving trajectories and reducing redundant generation, GRACE achieves substantial performance gains with only minor additional LLM computation. The key contributions of this work are summarized as follows:

- We introduce an asynchronous SLM–LLM collaboration paradigm in which the SLM remains the uninterrupted autoregressive decoder while receiving assistance from a stronger LLM, improving generation quality with only minor additional LLM computation.
- We propose **GRACE**, establishing bidirectional latent interaction through graded representation augmentation and streaming latent context synchronization. LLM representations provide stage-wise guidance across depth-aligned computation grades, while the evolving SLM context is compactly synchronized back to the LLM through latent tokens.
- Extensive experiments on diverse SLM-LLM configurations and different tasks show that GRACE achieves favorable performance-efficiency trade-offs across model scales, dense and MoE architectures, intra-family versions, and even different model families.

2 Related Works

Speculative Decoding. Speculative decoding has emerged as an effective paradigm for accelerating autoregressive LLM inference, where a lightweight draft model proposes multiple candidate tokens that are subsequently verified in parallel by the target LLM (Leviathan et al., 2023; Chen et al., 2023a). Subsequent studies have explored more efficient and accurate drafting mechanisms. Self-speculative decoding (Zhang et al., 2024) reuses subsets of the target model itself for drafting without introducing an auxiliary model, while Medusa (Cai et al., 2024) augments the target model with multiple decoding heads to predict several future tokens in parallel. EAGLE (Li et al., 2024b) further advances by performing autoregressive drafting over intermediate features rather than directly over tokens. Building on this feature-level drafting paradigm, EAGLE-2 (Li et al., 2024c) introduces context-aware dynamic draft trees to improve candidate construction, while EAGLE-3 (Li et al., 2026) further improves the drafter through direct token prediction and multi-layer feature fusion. Although these approaches substantially reduce the wall-clock latency of LLM generation, verification by the target LLM still incurs considerable additional computation. In contrast, GRACE decouples LLM-side representation computation from the SLM decoding path, allowing the SLM to maintain decoding throughput close to standalone inference while requiring only minor auxiliary LLM computation for graded representation augmentation.

Cross-Model Collaborative Inference. A single model is often difficult to simultaneously optimize for strong capability, low computational cost, and low inference latency (Chen et al., 2023b; Ding et al., 2024). This has motivated cross-model collaborative inference in language model area. At the query level, Hybrid LLM (Ding et al., 2024) dynamically routes requests between SLMs and LLMs according to query difficulty and desired quality, while RouteLLM (Ong et al., 2025) learns routing policies to effectively balance response quality and serving cost. More fine-grained collaboration has subsequently been explored during decoding. CITER (Zheng et al., 2025) introduces token-level routing between SLMs and LLMs, assigning non-critical tokens to the SLM while invoking the LLM for critical ones. R2R (Fu et al., 2026a) identifies path-divergent tokens between SLM and LLM reasoning trajectories and selectively invokes the LLM only at these positions. Beyond model switching, C2C (Fu et al., 2026b) demonstrates that internal representations can directly mediate semantic communication by projecting and fusing their KV caches, avoiding explicit intermediate text generation. These efforts demonstrate the potential of collaborating models with different capabilities to improve generation performance of SLM. However, these methods still require the SLM to wait for LLM assistance, while the additional LLM computation can introduce non-negligible

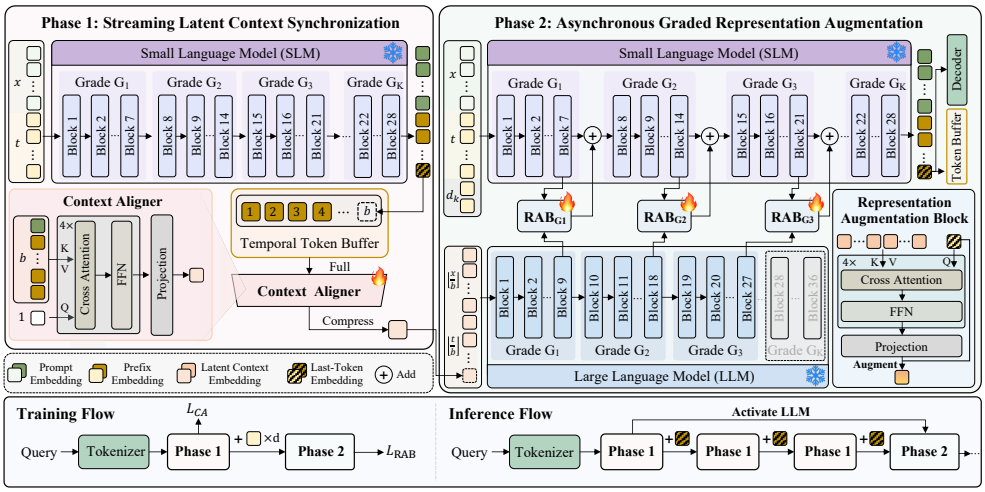


Figure 2: **Overview of GRACE.** Phase 1 performs streaming latent context synchronization, where the context aligner compresses evolving SLM representations into LLM-readable latent tokens. Phase 2 performs asynchronous graded representation augmentation, where task-relevant latent guidance extracted from grade-wise LLM representations progressively augments the corresponding SLM representations. The bottom panel summarizes collaborative training with simulated temporal delays and non-blocking asynchronous inference.

computation cost. GRACE instead enables asynchronous SLM–LLM collaboration through bidirectional latent interaction, achieving a favorable performance-efficiency trade-off.

3 METHOD

We propose **GRACE**, a graded representation augmentation framework for cost-effective SLM–LLM collaboration, as illustrated in Figure 2. GRACE retains the frozen SLM as the primary autoregressive decoder and enables asynchronous bidirectional latent interaction with a frozen LLM. Both models are partitioned into depth-aligned computation grades, where *representation augmentation blocks* (RABs) extract task-relevant latent guidance from the available LLM representations to progressively augment the corresponding SLM representations. Meanwhile, a *context aligner* compresses temporal buffers of evolving SLM representations into LLM-readable latent context tokens, incrementally synchronizing the SLM context to the LLM.

3.1 PROBLEM FORMULATION

Given a frozen SLM S and a frozen LLM L, we study how the two models can collaborate to enhance the generation capability of the SLM. Rather than transferring discrete predictions or using the LLM to directly drive token generation, we organize SLM–LLM collaboration through depth-aligned computation grades. Specifically, we proportionally partition the transformer blocks of each model into K consecutive depth grades. For a language model $M \in \{S, L\}$ with D^M transformer blocks, $\{b_1, b_2, ...b_{D^M}\} \mapsto \{G_1, G_2, ...G_K\}$, where each grade contains approximately $\lfloor D^M/K \rfloor$ consecutive blocks. This partition establishes aligned collaboration levels between the SLM and LLM, allowing the representations to interact at corresponding stages of computation.

Based on these depth-aligned grades, our goal is to learn how the rich representations of the frozen LLM can be transformed into task-relevant latent guidance that effectively augments the frozen SLM. Specifically, for each aligned grade k, we define a cross-model representation mapping $\mathcal{F}_k : \{h^S_{t,k}, \mathcal{H}^L_{r(t),k}\} \mapsto g_{t,k}, r(t) < t$, where $h^S_{t,k}$ denotes the SLM representation at the k-th grade for the current decoding step t, and $\mathcal{H}^L_{t,k}$ denotes the accumulated LLM representations available at the corresponding grade. The resulting $g_{t,k}$ serves as the latent guidance for augmenting the SLM

representation at the corresponding grade. We further consider asynchronous collaboration characterized by $\Delta T^L_{wait} = 0$, in which the SLM continues autoregressive decoding without waiting for LLM-side computation. Consequently, $\mathcal{H}^L_{t,k}$ may be derived from an earlier SLM context state than the current decoding step t. We then seek the set of grade-wise mappings $\mathcal{F} = \{\mathcal{F}_k\}_{k=1}^{K-1}$ that maximizes the generation improvement from LLM latent guidance under asynchronous collaboration with $\Delta T^L_{wait} = 0$. Accordingly, we formulate the objective as:

$$\arg\max_{\mathcal{F}} \mathbb{E}_{(x,y)\sim p_{\text{data}}} \sum_t \left[\log p_S\left(y_t \mid x, y_{<t}, g_{t,1:K-1}\right) - \log p_S\left(y_t \mid x, y_{<t}\right)\right], \tag{1}$$

where x denotes the input prompt, y denotes the corresponding gold response, and $g_{t,1:K-1}$ denotes the grade-wise latent guidance incorporated at decoding step t. The first term denotes the token-level log-likelihood of the frozen SLM augmented by LLM latent guidance, while the second term denotes that of standalone SLM decoding. Their difference measures the generation improvement contributed by asynchronous graded SLM-LLM collaboration. The objective therefore learns grade-wise mappings that transform LLM representations into task-relevant latent guidance for the SLM, improving generation quality while preserving uninterrupted autoregressive decoding.

3.2 Asynchronous Graded Representation Augmentation

A straightforward way to exploit LLM assistance is to invoke the LLM and use its output either for the current token prediction (Fu et al., 2026a) or to hand over the entire generation process to the LLM (Ding et al., 2024; Ong et al., 2025). Such synchronous collaboration forces the SLM to wait for LLM computation whenever assistance is invoked, interrupting autoregressive decoding and increasing generation latency. To avoid this bottleneck, GRACE decouples LLM-side representation computation from the SLM decoding path. Once the grade-wise LLM representations become available, they are incorporated into the SLM representations at the latest eligible decoding step through *representation augmentation blocks* (RAB), without interrupting any ongoing SLM decoding. Although asynchronous collaboration introduces a token-level mismatch, GRACE exploits the higher-level task information encoded in these representations (Hanna & Ameisen, 2026), including LLM formed task interpretation. This information is less tied to decoding positions and can therefore remain useful for later decoding.

To exploit the richer representations formed through the LLM's higher-capacity contextual processing, RAB use all representations $\mathcal{H}^L_{t,k} = \{h^L_{i,k}\}_{i=1}^{m_t}$ from its k-th grade as keys and values, where m_t denotes the number of context tokens available when the LLM-side computation is requested. The complete grade-wise representation sequence preserves task-relevant information distributed across the entire LLM context. Due to asynchronous execution, by the time these representations become available, the SLM may have advanced by d decoding steps. RAB therefore use the current SLM representation $h^S_{t+d,k}$ as the query to retrieve task-relevant information through multi-head cross-attention, followed by a feed-forward network (FFN):

$$g_{t+d,k} = \text{FFN}\left(\text{CrossAttn}\left(Q = \Phi_Q\left(h^S_{t+d,k}\right), K = \Phi_K\left(\mathcal{H}^L_{t,k}\right), V = \Phi_V\left(\mathcal{H}^L_{t,k}\right)\right)\right), \tag{2}$$

where Φ_Q, Φ_K, and Φ_V denote the corresponding query, key, value transformations. The resulting latent guidance g^k_{t+d} augments the SLM representation at the corresponding grade.

Considering that prompt prefill is critical for establishing the model's overall task understanding and solution trajectory (Dong et al., 2025), errors formed at this stage can be difficult to correct through later token-level refinement alone. RAB therefore treat the first interaction differently. When the LLM representations first become available, RAB replay the accumulated SLM context and use its full representations $\mathcal{H}^S_{t+d,k}$ at grade k as queries denoted by $Q = \Phi_Q(\text{SelfAttn}(\mathcal{H}^S_{t+d,k}))$. This augmentation incorporates the initial LLM guidance into the SLM representations and reconstructed KV cache. Subsequent interactions then require only the current decoding-token representation as the query, enabling local refinement while preserving earlier guidance through KV cache.

The extracted latent guidance is incorporated into the corresponding SLM representation through a residual connection. Inspired by the graded organization and collaborative processing of information in the human brain (Quent et al., 2025), GRACE performs RAB at each aligned grade, allowing the subsequent SLM layers to progressively absorb the injected information before receiving additional guidance at the next grade. The guidance from deeper LLM grades can further complement and

refine the representation already shaped by earlier interactions. The resulting SLM decoding process with grade-wise augmentation is formulated as

$$y^S_{t+d+1} = \arg\max_y p_S\left(y \mid x, y_{\le t+d}, \left\{\tilde{h}^S_{t+d,k}\right\}_{k=1}^{K-1}\right), \ \tilde{h}^S_{t+d,k} = h^S_{t+d,k} + g_{t+d,k}, \tag{3}$$

where the augmented representation at each grade is first processed by the subsequent SLM blocks before additional guidance is introduced at the next aligned grade. In this way, GRACE alternates between cross-model augmentation and native SLM computation, enabling guidance from different LLM grades to progressively complement and refine the ongoing SLM decoding state.

3.3 Streaming Latent Context Synchronization

Asynchronous SLM–LLM collaboration complicates synchronization between the LLM context and the continuously evolving SLM context. Synchronizing through discrete text would require repeatedly re-encoding the accumulated context on the LLM side, introducing redundant computation and diminishing the efficiency gains of sparse interaction. Instead, GRACE directly reuses the contextual representations already computed by the SLM, which encode both the accumulated semantics and task-relevant information shaped by the SLM's own processing. To convert these representations into an LLM-readable latent context, we design a *Context Aligner* that operates in a streaming manner. It groups the SLM's final-grade representations over the prompt and subsequently decoded tokens into temporal buffers of capacity b, compresses each full buffer into a single latent context token, and incrementally appends the resulting tokens to the LLM-side context as decoding proceeds. In this way, GRACE maintains LLM-side context synchronization directly in the latent space while reducing both repeated context encoding and the effective sequence length processed by the LLM.

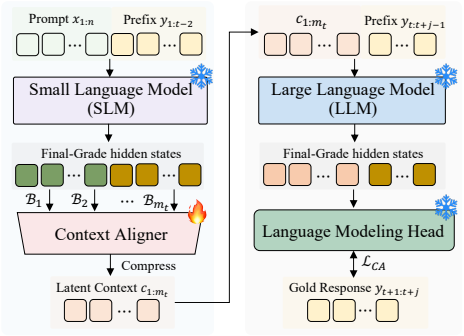


Figure 3: **Context-asymmetry alignment.** The SLM encodes the prompt and a dynamically sampled gold prefix, whose hidden states are compressed by context aligner into latent context tokens. The LLM relies on these tokens to predict gold continuations, enforcing cross-model latent alignment.

To make the latent context interpretable to the frozen LLM, we first warm up the context aligner with an auxiliary context-asymmetry alignment task, as illustrated in Figure 3. At a sampled response boundary t, the SLM is conditioned on the original prompt x and the gold prefix $y_{1:t-2}$. Under causal next-token prediction, the final hidden state of this prefix provides the representation used to predict y_{t-1}. We compress the accumulated final-grade representations into latent context tokens, which serve as the only representation of the preceding context available to the frozen LLM. This information asymmetry encourages the context aligner to encode task-relevant historical information into a form that can be utilized by the frozen LLM. The LLM is then teacher-forced on the subsequent gold response conditioned on the synchronized latent context:

$$\mathcal{L}_{\mathrm{CA}} = -\mathbb{E}_{(x,y)\sim p_{data},t} \sum_{j=t+1} \log p_L(y_j \mid c_{1:m_t}, y_{t:j-1}), c_i = \mathrm{ContextAligner}(\mathcal{B}_i), \tag{4}$$

where c_i denotes the i-th latent context token, $\mathcal{B}_i$ is the corresponding temporal buffer of SLM final-grade representations, and m_t denotes the number of fully populated buffers available at boundary t. Sampling different response boundaries exposes the context aligner to varying amounts of accumulated context, resembling the synchronization states encountered during asynchronous collaboration. Since the prompt and response history preceding t are accessible to the LLM only through the latent contexts, minimizing $\mathcal{L}_{\mathrm{CA}}$ encourages the context aligner to preserve task-relevant contextual information in an LLM-readable latent form. This objective establishes latent compatibility and provides a suitable initialization for subsequent collaborative training.

3.4 Collaborative Training and Inference

After warming up the context aligner, we jointly train the context aligner and RABs while keeping both models frozen. To simulate asynchronous collaboration, we keep the LLM representations

fixed at an earlier request state and append d additional gold response tokens to the SLM context as delay tokens before interaction. The delay d is sampled with a Bernoulli-based scheme, exposing the model to varying temporal mismatches and training the RABs to utilize stale LLM guidance. The overall collaborative training objective is therefore

$$\mathcal{L} = \mathcal{L}_{\mathrm{RAB}} + \lambda_{\mathrm{CA}}\mathcal{L}_{\mathrm{CA}}, \qquad \mathcal{L}_{\mathrm{RAB}} = -\mathbb{E}_{(x,y)\sim p_{\mathrm{data}}} \sum\nolimits_t \log p_S\left(y_t \mid x, y_{<t}; d\right), \tag{5}$$

where λ_{CA} controls the contribution of the auxiliary objective, helping preserve the ability of context aligner to produce LLM-readable latent contexts during collaborative training.

At inference, GRACE starts collaboration after SLM prompt prefill by synchronizing the SLM context to the LLM in latent space. The SLM then continues autoregressive decoding while the LLM computes asynchronously. When the first LLM representations arrive, GRACE replays the accumulated SLM context with graded augmentation and reconstructs the augmented KV cache. Subsequent interactions periodically incorporate the latest available guidance. Throughout inference, the SLM remains the primary autoregressive decoder without waiting for LLM computation.

4 Experiments

We extensively evaluate GRACE on two representative tasks that cover complementary aspects of SLM task understanding. Instruction following requires precise interpretation and satisfaction of multiple constraints, while mathematical reasoning requires formulating coherent and correct problem-solving trajectories. For instruction following, we evaluate on IFEval (Zhou et al., 2023), IFBench (Pyatkin et al., 2026), MulDimIF (Ye et al., 2026), and the instruction-following subset of LiveBench (White et al., 2025), primarily reporting prompt-level accuracy, where a response is correct only when all constraints in the prompt are satisfied. For mathematical reasoning, we report the average accuracy over AIME 2024–2026 (Hugging Face H4, 2025; OpenCompass Community, 2025; Zhang & Math-AI, 2026). We additionally report a modified *Compute-Normalized Gain* (CNG*), adapted from prior work (Bilal et al., 2026; Muennighoff et al., 2023) , which measures the performance gain over the standalone SLM per unit of absolute compute difference. We further evaluate GRACE across diverse SLM–LLM configurations, followed by ablation studies.

4.1 Experimental Results

Collaboration across Qwen models, architectures and versions. We evaluate GRACE with Qwen3-1.7B (Yang et al., 2025) as the SLM and different Qwen-family models as the collaborating LLM. As shown in Table 1, GRACE consistently improves the standalone SLM when pairing with the dense Qwen3-14B, the MoE Qwen3-30B-A3B (Yang et al., 2025), and the newer Qwen3.5-9B (Qwen Team, 2026), demonstrating robust collaboration across model architectures and versions. With Qwen3-14B, GRACE narrows the performance gap between Qwen3-1.7B and the vanilla Qwen3-4B, while surpassing it on most benchmarks. The results indicate that graded representation augmentation can effectively transfer task-relevant guidance from a stronger LLM to improve SLM generation. Notably, GRACE can improve mathematical reasoning while requiring less overall inference computation than standalone SLM decoding. On AIME, collaboration with Qwen3-30B-A3B improves Qwen3-1.7B accuracy from 7.78 to 16.67, yielding a CNG* of 64.82, while Qwen3.5-9B achieves 17.78 accuracy and reduces latency from 74.03 to 42.32 seconds. As further detailed in Table 10, the reduced generation cost offsets the additional LLM computation, suggesting that latent guidance helps the SLM follow clearer and more direct reasoning trajectories, avoiding redundant decoding while improving solution quality.

Compared with existing approaches, GRACE provides a distinct balance between generation quality, computation, and latency. EAGLE-3 (Li et al., 2026) generally achieves lower decoding latency through speculative decoding, but its target-model verification introduces substantial additional computation, resulting in lower CNG* than GRACE in most settings. RouteLLM (Ong et al., 2025) and Jev (Almeida, 2026) are competitive on mathematical reasoning but less consistent on instruction following, likely due to the difficulty of query-level routing for complex instructions. Moreover, routed queries still require full LLM generation, resulting in non-negligible additional computation. C2C (Fu et al., 2026b) also improves several benchmarks through progressively projecting and fusing LLM KV caches into the SLM at the layer level, whereas GRACE yields more stable gains.

Table 1: **Performance–efficiency comparison based on the Qwen family.** CNG* denotes the accuracy gain over the standalone SLM normalized by the absolute difference in inference compute (TFLOPs), with higher values indicating better performance–efficiency trade-off. Latency (Lat.) is measured in seconds per sample.

	IFEval			IFBench			LiveBench-IF			MulDimIF			AIME		
Methods	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**
Vanilla Language Model															
Qwen3-4B	78.37	5.53	8.40	22.33	2.16	9.26	57.50	2.07	9.43	53.92	8.14	5.71	16.66	0.33	110.19
Qwen3-1.7B + Qwen3-14B															
Qwen3-1.7B	68.02	–	6.27	17.67	–	8.34	50.00	–	7.35	32.92	–	4.92	7.78	–	74.03
Qwen3-14B	83.55	1.67	8.81	26.67	0.80	10.34	69.50	1.04	9.65	59.58	1.92	6.75	22.22	0.13	110.60
EAGLE-3‡ (14B)	84.29	0.45	5.35	27.67	0.27	5.41	68.00	0.40	5.38	59.75	0.72	4.55	22.22	0.06	48.30
RouteLLM	69.50	1.37	–	20.00	0.79	–	52.50	1.15	–	51.00	2.24	–	15.58	0.12	–
Jev†	70.98	1.45	6.83	20.00	0.30	9.77	**63.50**	1.44	8.18	37.92	1.04	5.80	14.44	0.07	103.25
C2C	68.02	0.00	6.99	19.67	0.75	**7.87**	50.00	0.00	**5.71**	44.83	1.14	7.04	8.89	0.10	95.66
GRACE (Ours)	**74.12**	**8.97**	**6.76**	**24.67**	**8.97**	7.94	60.50	**2.75**	7.63	**54.42**	**6.78**	**5.71**	**17.78**	**1.50**	**94.40**
Qwen3-1.7B + Qwen3-30B-A3B															
Qwen3-1.7B	68.02	–	6.27	17.67	–	8.34	50.00	–	7.35	32.92	–	4.92	7.78	–	74.03
Qwen3-30B-A3B	82.99	13.73	10.92	28.00	8.40	12.82	68.50	8.64	11.71	58.08	15.73	7.80	21.11	0.69	125.04
EAGLE-3‡ (30B)	81.89	5.12	2.09	28.33	3.71	2.33	68.00	4.63	2.30	57.67	7.48	1.84	18.89	0.41	22.84
RouteLLM	69.69	11.13	–	19.67	6.45	–	50.50	1.67	–	48.50	**15.58**	–	**16.67**	0.73	–
Jev†	69.87	11.56	7.32	18.67	1.19	12.30	56.50	**7.47**	10.05	35.33	4.82	6.87	**16.67**	0.62	118.65
C2C	66.17	-2.20	5.71	20.33	2.44	7.09	47.50	-0.79	5.92	36.42	1.08	6.58	10.00	0.25	76.92
GRACE (Ours)	**73.20**	**17.27**	**4.52**	**29.00**	**34.33**	**5.26**	**59.50**	4.40	**4.90**	**53.08**	10.96	**3.69**	**16.67**	**64.82**	**46.37**
Qwen3-1.7B + Qwen3.5-9B															
Qwen3-1.7B	68.02	–	6.27	17.67	–	8.34	50.00	–	7.35	32.92	–	4.92	7.78	–	74.03
Qwen3.5-9B	82.07	3.23	12.57	33.00	2.73	13.07	65.00	1.66	14.18	59.42	4.36	6.84	58.89	0.65	197.59
EAGLE-3‡ (9B)	81.33	1.81	6.62	33.67	1.00	6.94	65.00	0.77	7.04	59.75	1.64	4.18	55.56	0.28	78.80
RouteLLM	70.79	5.23	–	20.67	2.00	–	50.50	0.47	–	49.00	4.72	–	**42.22**	0.75	–
Jev†	71.90	4.08	7.15	23.33	1.43	12.30	**62.50**	**2.72**	9.52	37.42	2.10	5.78	38.89	0.46	189.38
GRACE (Ours)	**73.20**	**12.63**	**6.54**	**27.33**	**20.55**	**9.01**	56.00	2.13	**6.33**	**52.58**	**8.44**	**4.85**	17.78	**21.76**	**42.32**

† Jev API is used and its own computational cost is excluded. ‡ denotes results obtained with vLLM inference acceleration framework.

With Qwen3-14B, GRACE improves IFEval and LiveBench-IF from 68.02 and 50.00 to 74.12 and 60.50, while C2C remains close to the standalone SLM on both tasks. This difference highlights the benefit of our graded design, where guidance invoked at each grade is progressively absorbed by subsequent SLM layers before complementary refinement at later grades is provided at deeper grades. Together with the higher CNG* achieved in most settings, these results show a favorable performance–efficiency trade-off across Qwen families.

Collaboration across different model families. We further evaluate GRACE using Ministral3-3B-Instruct (Liu et al., 2026) and Qwen3-14B, which differ in model family and tokenizer. As shown in Table 2, GRACE consistently improves the standalone Ministral3-3B-Instruct across the instruction-following benchmarks, with particularly clear gains of 22.5 points on LiveBench-IF, 16.75 points on MulDimIF, and 12.76 points on IFEval. GRACE also achieves the best accuracy on IFBench, LiveBench-IF, and MulDimIF among the collaboration methods. The gains are achieved without increasing decoding latency over the standalone SLM on these benchmarks. On IFEval, the unusually high CNG* is associated with latent guidance mitigating degenerate repetition in SLM, offsetting much of the additional collaboration cost. On AIME, LLM itself outperforms SLM by only 3.33 points, leaving relatively limited additional capability available for transfer, which may partly explain the absence of further improvement from collaboration. Overall, these results indicate that GRACE can extract useful task-relevant guidance from LLM representations even across different model families and tokenization spaces.

4.2 Ablation Studies

We conduct ablation studies using Qwen3-1.7B and Qwen3-14B as the SLM–LLM pair. As shown in Table 3, replacing the context aligner with the direct text re-encoding increases the computation from 1.94 to 2.87 TFLOPs on IFEval and from 22.24 to 27.00 TFLOPs on AIME, while reducing accuracy by 4.06 and 10.00 points. These results show that the context aligner not only reduces LLM-side context-processing cost through latent compression, but also preserves task-relevant con-

Table 2: **Performance–efficiency comparison across heterogeneous SLM–LLM pair.** CNG* denotes the accuracy gain over the standalone SLM normalized by the absolute difference in inference compute (TFLOPs) Latency (Lat.) is measured in seconds per sample.

	IFEval			IFBench			LiveBench-IF			MulDimIF			AIME		
Methods	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**	**Acc.↑**	**CNG*↑**	**Lat.↓**
Ministral3-3B-Instruct	54.71	–	16.95	19.33	–	19.23	23.50	–	15.15	40.50	–	13.38	18.89	–	200.00
Qwen3-14B	83.55	4.10	8.81	26.67	0.84	10.34	69.50	2.92	9.65	59.58	1.72	6.75	22.22	0.05	110.60
RouteLLM	57.86	3.62	–	21.33	0.87	–	29.00	2.96	–	53.00	1.93	–	18.89	0.02	–
Jev†	56.19	0.93	16.39	22.67	0.57	**14.93**	39.50	2.39	13.51	43.50	0.90	12.50	**21.11**	**0.06**	**142.86**
C2C	**70.43**	9.53	**13.51**	22.33	1.14	18.87	38.00	1.09	14.71	44.25	0.37	12.05	7.78	-2.40	166.67
GRACE (Ours)	67.47	**1276.00**	**13.51**	**24.33**	**7.35**	17.16	**46.00**	**4.14**	**12.66**	**57.25**	**4.10**	**10.60**	16.67	-0.62	222.77

† Jev API is used and its own computational cost is excluded.

Table 3: **Component ablation of GRACE.** *w/o Context Aligner* replaces latent synchronization with text retokenization, while *w/o KV Cache Reconstruction* removes cache rebuilding during the initial augmentation. Latency (Lat.) is measured in seconds per sample.

	IFEval			AIME		
Variant	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**
GRACE (Full Model)	74.12	1.94	6.76	17.78	22.24	94.40
w/o Context-Asymmetry Task	70.24	1.86	7.04	7.78	16.45	71.43
w/o Context Aligner	70.06	2.87	6.49	7.78	27.00	111.11
w/o KV Cache Reconstruction	66.36	1.64	6.37	7.78	18.09	83.33
w/o Subsequent Augmentation	73.20	1.86	6.54	14.44	23.72	100.00

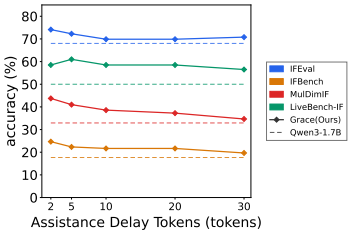


Figure 4: Robustness to asynchronous assistance in the collaboration between Qwen3-1.7B and Qwen3-14B.

textual information more effectively for collaboration. Besides, removing the auxiliary context-asymmetry task consistently degrades performance, highlighting the importance of establishing an LLM-readable latent context space before collaborative training. The ablations also reveal the importance of the initial replay after prompt prefill. Removing KV cache reconstruction reduces accuracy from 74.12 to 66.36 on IFEval and from 17.78 to 7.78 on AIME, showing that augmenting only the current decoding state is insufficient to revise the task representation already encoded in the SLM cache. In contrast, removing subsequent augmentation causes a smaller performance drop, indicating that the initial replay accounts for much of the benefit by globally updating the decoding state, while later interactions mainly provide complementary refinement. The additional 3.34-point drop on AIME nevertheless shows that continued guidance remains useful during extended reasoning. Finally, we evaluate robustness to asynchronous execution by varying the arrival delay of LLM assistance, as shown in Figure 4. GRACE maintains stable performance across different simulated delays, demonstrating that the latent guidance remains effective despite the temporal mismatch between LLM computation and the SLM decoding process.

5 Conclusion

In this paper, we presented GRACE, a framework for cost-effective SLM–LLM collaboration that enhances SLM generation quality with minor additional computation. Unlike traditional speculative decoding or model routing, GRACE maintains the SLM as the uninterrupted primary autoregressive decoder while establishing an asynchronous, bidirectional latent interaction with a frozen LLM. This interaction is facilitated through representation augmentation blocks and a context aligner, operating across depth-aligned computation grades to allow the SLM to progressively absorb LLM guidance. Our analysis reveals that correcting the global task representation early in the generation process is crucial, while subsequent interactions act as complementary refinements. Extensive experiments on instruction-following and mathematical reasoning benchmarks demonstrate that GRACE strengthens the SLM's ability to handle complex tasks while maintaining favorable performance–efficiency trade-offs across different model scales, architectures, versions, and families. Extending GRACE to broader tasks and more diverse SLM-LLM pairs as well as adaptively invoking LLM assistance based on the evolving SLM decoding state, remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with and polish the writing, as well as to support the development of portions of the code. All AI-assisted text was reviewed and revised by the authors, and AI-assisted code was checked and tested for correctness before use. We take full responsibility for the final content of this work, including all text, claims, analyses, and code produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Diogo Almeida. Introducing system one models & jev, September 2026. URL https://typesafe.ai/blog/introducing-system-one-models-and-jev. TypeSafe AI Blog.

Ahsan Bilal, Muhammad Ahmed Mohsin, Muhammad Umer, Lena Trigg, Ali Subhan, Muhammad Ali, and Dean F Hougen. Refining over resampling: Test-time self-correction for llm reasoning. *arXiv preprint arXiv:2608.05643*, 2026.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023b.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks Lakshmanan, and Ahmed H Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, volume 2024, pp. 41348–41366, 2024.

Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. Emergent response planning in LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 14301–14320. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dong25p.html.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. *Advances in Neural Information Processing Systems*, 38:124108–124145, 2026a.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations*, volume 2026, pp. 43130–43158, 2026b.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Michael Hanna and Emmanuel Ameisen. Latent planning emerges with scale. *arXiv preprint arXiv:2604.12493*, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hugging Face H4. Aime 2024. Hugging Face Dataset, 2025. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. https://huggingface.co/datasets/AI-MO/NuminaMath-1.5, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024c.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

OpenCompass Community. Aime 2025 dataset. Hugging Face Dataset, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

Jörn A Quent, Liangyue Song, Xinyu Liang, Yueting Su, Wenwen Yu, He Wang, and Deniz Vatansever. Graded encoding of spatial novelty scales in the human brain. *Nature Communications*, 17 (1):303, 2025.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, volume 2025, pp. 91595–91631, 2025.

Tianze Xu, Yanzhao Zheng, Pengrui Lu, Lyumanshan Ye, Yong Wu, Zhentao Zhang, Yuanqiang Yu, Chao Ma, Jihuai Zhu, Pengfei Liu, et al. Rubrics to tokens: Bridging response-level rubrics and token-level rewards in instruction following tasks. *arXiv preprint arXiv:2604.02795*, 2026.

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.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, et al. Muldimif: A multi-dimensional constraint framework for evaluating and improving instruction following in large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2078–2104, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

APPENDICES

A ADDITIONAL EXPERIMENTAL RESULTS

A.1 EFFECTIVENESS OF LATENT CONTEXT TOKENS

Context aligner architecture. As described in Sec. 3.3, the context aligner incrementally transfers the evolving SLM context into the LLM latent space by compressing each full temporal token buffer into a single LLM-readable context token. To preserve informative content, we design a learnable latent query that selectively aggregates information from the buffered SLM representations, rather than directly pooling or projecting them. For the i-th temporal buffer $\mathcal{B}_i \in \mathbb{R}^{b\times \mathrm{dim}_S}$, which contains b consecutive final-grade SLM representations, a shared learnable query $q^{(0)} \in \mathbb{R}^{\mathrm{dim}_S}$ is used to extract the information relevant for context synchronization. Each interaction block is formulated as

$$\hat{q}_i^{(\ell)} = q_i^{(\ell-1)} + \mathrm{CrossAttn}\left(Q = q_i^{(\ell-1)}, K = \mathcal{B}_i, V = \mathcal{B}_i\right), q_i^{(\ell)} = \hat{q}_i^{(\ell)} + \mathrm{FFN}\left(\mathrm{LN}\left(\hat{q}_i^{(\ell)}\right)\right), \quad (6)$$

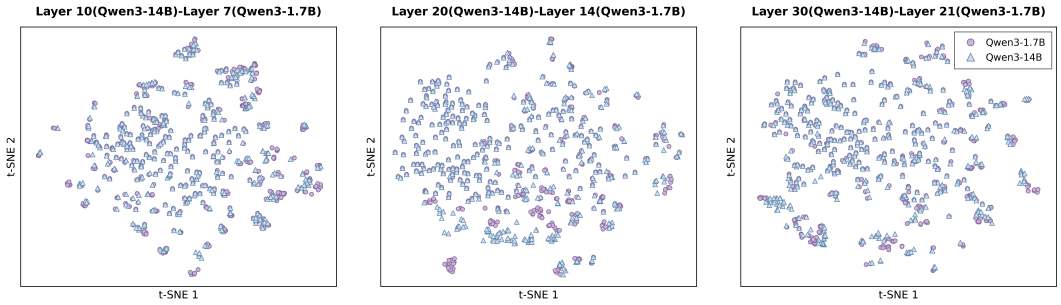
where $q_i^{(\ell)}$ denotes the latent query representation after the ℓ-th interaction block ($\ell = 4$), and the same learnable query $q^{(0)}$ is used to initialize each buffer, i.e., $q_i^{(0)} = q^{(0)}$. The stacked interaction blocks progressively aggregate the buffered SLM context into a single compact latent representation.

After four interaction blocks, the resulting latent representation is normalized and projected from the SLM hidden dimension dim_S to the LLM hidden dimension dim_L. This query-based compression allows the aligner to jointly perform temporal context aggregation and cross-model latent mapping while keeping the synchronized LLM context compact.

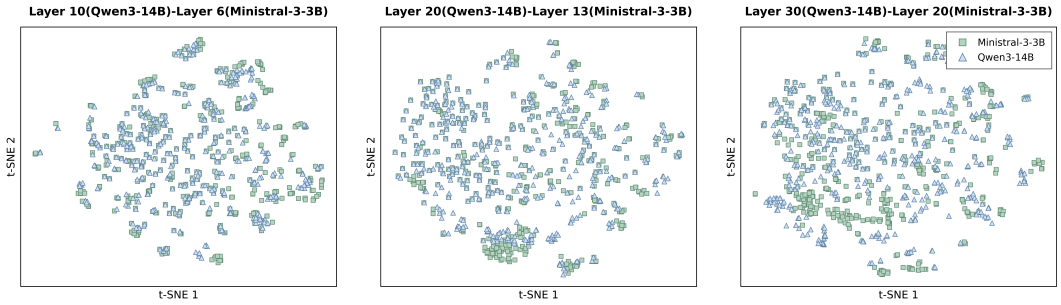
Quantitative analysis. To further evaluate whether the latent context token preserves information that can be effectively interpreted by the LLM, we directly condition the frozen Qwen3-14B on the latent context tokens produced by the context aligner, without providing the original prompt text. We consider two source SLMs: Qwen3-1.7B, which shares the model family and tokenizer with the LLM, and Ministral3-3B, which provides a cross-family setting with a different tokenizer. We then compare direct LLM generation from these latent contexts against standard text-based prompting in terms of instruction-following accuracy and context-processing cost. As shown in Table 4, latent context tokens derived from both SLMs retain substantial task-relevant information and allow the frozen LLM to perform instruction following without access to the original textual context. The stronger results obtained with Qwen3-1.7B indicate that representation compatibility within the same model family is beneficial, while the Ministral3-3B results further show that the learned latent context remains interpretable even across different model families and tokenization spaces. This demonstrates that the context aligner learns an LLM-readable latent interface rather than merely relying on shared discrete token representations. The computational benefit becomes more pronounced as the prompt length increases. On LiveBench-IF and MulDimIF, whose prompts contain on average 429.03 and 329.45 tokens, respectively, latent context compression substantially reduces both TFLOPs and LLM prefill time. In contrast, the reductions on IFEval and IFBench are more

Table 4: **Effectiveness of latent context compression.** We directly condition the frozen Qwen3-14B on either the original text context or the latent context tokens compressed by the Context Aligner.

Dataset	Prompt Token Num	Context	P-Acc.↑	C-Acc.↑	TFLOPs↓	Prefill Time (ms)↓
LiveBench-IF	429.03	Text	69.50	88.49	21.34	72.18
		Latent (Qwen3-1.7B)	56.00	82.74	**14.00**	**32.14**
		Latent (Ministral3-3B-Instruct)	53.50	80.54	**14.86**	**32.05**
MulDimIF	329.45	Text	59.58	84.67	15.70	61.35
		Latent (Qwen3-1.7B)	54.25	82.18	**10.03**	**31.78**
		Latent (Ministral3-3B-Instruct)	50.08	79.02	**11.26**	**31.97**
IFEval	58.39	Text	83.55	89.09	10.55	28.40
		Latent (Qwen3-1.7B)	69.50	78.42	**9.43**	**26.83**
		Latent (Ministral3-3B-Instruct)	59.52	70.74	**9.92**	**26.73**
IFBench	85.77	Text	26.67	29.65	12.91	31.10
		Latent (Qwen3-1.7B)	23.33	25.29	**11.29**	**27.01**
		Latent (Ministral3-3B-Instruct)	20.33	21.80	**12.18**	**26.88**



(a) Qwen3-1.7B and Qwen3-14B.



(b) Ministral3-3B and Qwen3-14B.

Figure 5: **T-SNE visualization of representations from corresponding computation depths of SLM–LLM pairs.** We randomly sample 500 examples from the MulDimIF test set and collect representations at aligned relative depths, illustrating cross-model correspondence across different stages of computation.

modest because their prompts are shorter, with only 58.39 and 85.77 tokens on average. These results demonstrate that the context aligner can compactly preserve task-relevant contextual information while reducing the cost of processing long contexts on the LLM side.

A.2 Analysis of graded SLM-LLM collaboration

Motivation for grade partition. To partition depth-aligned grade of different language model, we examine whether models with different depths exhibit comparable representation structures at corresponding stages of computation. Figure 5 visualizes representations collected from proportionally aligned layers of both same-family and cross-family model pairs. For Qwen3-14B and Qwen3-1.7B, representations at corresponding relative depths exhibit broadly overlapping structures across different stages. A similar correspondence can also be observed between Qwen3-14B and Ministral3-3B-Instruct, despite their differences in model family and architecture. These observations suggest that representations at similar relative depths retain a certain degree of cross-model correspondence, motivating our proportional partition of different models into depth-aligned grades for subsequent collaboration.

Progressive integration of grade-wise latent guidance. We further examine how representation augmentation at different computation grades contributes to SLM–LLM collaboration. The graded design distributes LLM guidance across the SLM computation process rather than concentrating it at a single depth. After each augmentation, subsequent SLM layers can further process the received information, while guidance introduced at later grades can complement the evolving SLM representation. As shown in Table 5, each individual grade provides useful guidance, while combining multiple grades generally yields stronger performance. Among the single-grade settings, G_2 achieves the strongest results on most benchmarks, suggesting that an intermediate grade provides a favorable balance between informative LLM representations and sufficient remaining SLM depth for integration. The effectiveness of different grade combinations is also task dependent. For example, $G_1 + G_2$ performs particularly well on the instruction-following benchmarks, whereas $G_2 + G_3$

Table 5: **Ablation of graded representation augmentation.** Different combinations of G_1, G_2, and G_3 indicate the grades at which SLM representations are augmented with LLM guidance.

			IFEval		IFBench		LiveBench-IF		MulDimIF		AIME
G_1	G_2	G_3	P-Acc.↑	C-Acc.↑	P-Acc.↑	C-Acc.↑	P-Acc.↑	C-Acc.↑	P-Acc.↑	C-Acc.↑	Acc.↑
✓			64.14	72.90	17.67	18.31	46.00	77.33	36.67	68.44	8.89
	✓		67.47	76.38	15.67	17.15	54.50	80.37	37.67	69.55	12.22
		✓	65.80	75.18	18.00	19.48	48.00	78.17	34.42	66.65	11.11
✓	✓		71.72	79.50	21.33	23.26	59.50	83.93	47.50	78.76	7.78
✓		✓	63.40	72.30	19.00	20.06	48.50	76.65	37.75	69.78	12.22
	✓	✓	70.06	78.30	16.67	18.60	48.50	76.65	38.00	70.00	15.56
✓	✓	✓	**74.12**	**81.65**	**24.67**	**25.58**	**60.50**	**84.26**	**54.42**	**81.72**	**17.78**

Table 6: **Sensitivity analysis of Grade Partitioning.** K denotes the number of partitioned grades. Latency (Lat.) is measured in seconds per sample.

	IFEval			IFBench			LiveBench-IF			MulDimIF		
Methods	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**
Qwen3-1.7B	68.02	1.26	6.27	17.67	1.65	8.34	50.00	2.54	7.35	32.92	1.78	4.92
Qwen3-14B	83.55	10.55	8.81	26.67	12.91	10.34	69.50	21.34	9.65	59.58	15.70	6.75
GRACE ($K = 4$)	**74.12**	**1.94**	**6.76**	**24.67**	**2.43**	**7.94**	**60.50**	**6.36**	**7.63**	**54.42**	**4.95**	**5.71**
GRACE ($K = 6$)	**73.75**	**1.92**	**6.37**	**24.33**	**2.43**	**8.00**	**58.00**	**6.83**	**6.80**	**55.25**	**5.14**	**4.74**

achieves the strongest two-grade result on AIME. This indicates that guidance from different computation stages provides complementary information whose utility can vary across tasks.

Using all three grades achieves the strongest overall performance, improving over the best single-grade configuration by 6.65, 6.67, 6.00, 16.75, and 5.56 points on IFEval, IFBench, LiveBench-IF, MulDimIF, and AIME. Although adding a later grade does not monotonically improve every individual metric, the full graded configuration provides the most consistent gains across both instruction-following and mathematical reasoning tasks. These results support our grade-wise design, where successive augmentation allows LLM guidance from different depths to complement one another while being progressively integrated through the SLM's own computation.

Sensitivity analysis of Grade Partitioning. To examine whether GRACE is sensitive to the specific grade partition, we further refine the default $K = 4$ configuration to $K = 6$. The original augmentation positions are retained, while two intermediate stages are added. Specifically, the LLM representations are extracted at layers $\{10, 15, 20, 25, 30\}$ of Qwen3-14B, and the corresponding augmentation is applied at layers $\{7, 10, 14, 18, 21\}$ of Qwen3-1.7B. As shown in Table 6, the finer partition achieves performance close to the default setting across all four benchmarks: accuracy changes from 74.12 to 73.75 on IFEval, from 24.67 to 24.33 on IFBench, from 60.50 to 58.00 on LiveBench-IF, and from 54.42 to 55.25 on MulDimIF. The computational cost also remains comparable, with only modest increases on the longer benchmarks due to the additional augmentation stages. These results indicate that GRACE is not highly sensitive to the exact grade granularity, while the default $K = 4$ partition provides a practical balance between collaboration effectiveness and computational overhead.

A.3 Computational Cost Analysis

We further analyze the inference computation of GRACE using Qwen3-1.7B as the SLM and Qwen3-14B as the LLM, and compare its cost with standalone models and existing collaboration methods. For most experiments, the SLM and LLM are concurrently deployed on a single NVIDIA RTX 5090 GPU. When the models cannot fit on a single GPU, we extend inference to two GPUs. All results use greedy non-thinking decoding. We report the mean TFLOPs per query, and the AIME results are averaged over AIME 2024–2026. For GRACE, the auxiliary computation consists of the context aligner (CA) and representation augmentation blocks (RABs), whose individual costs are reported in parentheses in Table 7.

As shown in Table 7, the compared collaboration methods exhibit different sources of computational overhead. RouteLLM provides four routing variants. We use its BERT-based router as the main im-

Table 7: **Inference performance and computation breakdown across evaluation benchmarks.** Computation is reported as mean TFLOPs per query.

Dataset	Method	Accuracy↑	SLM↓	LLM↓	Auxiliary↓	Total↓
IFEval	SLM	68.02	1.26	–	–	1.26
	LLM	83.55	–	10.55	–	10.55
	RouteLLM	69.50	**1.12**	1.21	0.01	2.34
	C2C	68.02	1.61	1.61	0.06	3.28
	GRACE (Ours)	**74.12**	1.52	**0.34**	0.09 (0.01 CA + 0.08 RAB)	**1.94**
IFBench	SLM	17.67	1.65	–	–	1.65
	LLM	26.67	–	12.91	–	12.91
	RouteLLM	20.00	**1.19**	3.40	0.01	4.61
	C2C	19.67	1.87	2.38	0.09	4.33
	GRACE (Ours)	**24.67**	1.89	**0.43**	0.11 (0.01 CA + 0.10 RAB)	**2.43**
MulDimIF	SLM	32.92	1.78	–	–	1.78
	LLM	59.58	–	15.70	–	15.70
	RouteLLM	51.00	**0.78**	9.03	0.04	9.85
	C2C	44.83	2.63	9.29	0.35	12.27
	GRACE (Ours)	**54.42**	2.86	**1.64**	0.44 (0.05 CA + 0.39 RAB)	**4.95**
LiveBench-IF	SLM	50.00	2.54	–	–	2.54
	LLM	69.50	–	21.34	–	21.34
	RouteLLM	52.50	**2.25**	2.39	0.08	**4.72**
	C2C	50.00	2.71	12.06	0.45	15.22
	GRACE (Ours)	**60.50**	3.58	**2.18**	0.60 (0.07 CA + 0.53 RAB)	6.36
AIME	SLM	7.78	15.70	–	–	15.70
	LLM	22.22	–	132.88	–	132.88
	RouteLLM	15.58	**6.33**	85.43	0.02	91.79
	C2C	8.89	25.11	3.92	0.15	29.18
	GRACE (Ours)	**17.78**	21.24	**0.81**	0.19 (0.03 CA + 0.16 RAB)	**22.24**

plementation, since the other variants rely on precomputed features that are less suitable for our online inference setting. Although query-level routing avoids unnecessary LLM calls, queries assigned to the LLM still require full LLM generation, leaving substantial LLM-side computation, such as 9.03 TFLOPs on MulDimIF and 85.43 TFLOPs on AIME. C2C relies on a series of lightweight projection and fusion modules, resulting in relatively small auxiliary overhead. However, it still requires substantial LLM-side representation computation, particularly for long-context tasks, reaching 9.29 and 12.06 TFLOPs on MulDimIF and LiveBench-IF, while providing limited performance gains.

GRACE follows a different computational trade-off. Its context aligner and RABs introduce slightly higher auxiliary computation than the lightweight modules used by C2C, but this additional overhead enables a much larger reduction in LLM-side computation through latent context compression and graded representation computation. On MulDimIF and LiveBench-IF, GRACE reduces LLM computation to 1.64 and 2.18 TFLOPs, while achieving 54.42 and 60.50 accuracy, respectively. As a result, the total computation remains substantially below C2C and competitive with routing-based approaches while providing stronger generation quality. These results show that GRACE trades a modest amount of auxiliary computation for a substantial reduction in expensive LLM computation, leading to a favorable performance–efficiency trade-off.

A.4 Collaboration with a SLM without post-training

We further examine whether GRACE can benefit an SLM that has not undergone post-training. Specifically, we use the pretrained-only Gemma4-E2B (Team et al., 2026) as the SLM and Qwen3-14B as the collaborating LLM. Gemma4-E2B adopts a dense transformer architecture with interleaved local and global attention, and we place representation augmentation immediately before its three global-attention layers to provide LLM assistance at representative stages of the SLM.

As shown in Table 8, the pretrained-only Gemma4-E2B exhibits limited instruction-following ability on its own, achieving 18.67, 4.33, 1.00, and 6.83 accuracy on IFEval, IFBench, LiveBench-IF, and MulDimIF, respectively. With GRACE, the corresponding accuracy improves to 35.49, 11.33, 9.00,

Table 8: **Performance and efficiency comparison for Gemma4-E2B–Qwen3-14B collaboration.** Latency (Lat.) is measured in seconds per sample.

	IFEval			IFBench			LiveBench-IF			MulDimIF		
Methods	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**	**Acc.↑**	**TFLOPs↓**	**Lat.↓**
Gemma4-E2B	18.67	2.94	20.41	4.33	2.59	17.24	1.00	2.77	8.77	6.83	4.68	25.64
Qwen3-14B	83.55	10.55	8.81	26.67	12.91	10.34	69.50	21.34	9.65	59.58	15.70	6.75
RouteLLM	24.95	3.88	-	9.33	5.50	-	8.00	4.84	-	34.25	11.29	-
GRACE (Ours)	**35.49**	**1.76**	**8.62**	**11.33**	**2.12**	**9.71**	**9.00**	**5.03**	**3.60**	**36.50**	**4.85**	**4.98**

and 36.50. GRACE also consistently outperforms RouteLLM under this setting, suggesting that hard switching becomes less effective when the SLM itself has limited task capability, since generation segments assigned to the SLM remain constrained by its pretrained-only behavior. These results suggest that representation-level collaboration can provide a pretrained-only SLM with useful task-specific guidance from a post-trained LLM, allowing it to exhibit a degree of instruction-following capability without directly post-training the SLM backbone. This further indicates that GRACE can transfer task-relevant capabilities through collaboration rather than relying entirely on capabilities already acquired by the SLM during post-training.

A.5 Detailed Experimental Results

The detailed results in Tables 9–12 provide a finer-grained view of how GRACE benefits from different collaborating LLMs and where its efficiency gains arise.

Recovering substantial LLM capability with limited computation. GRACE can recover a substantial portion of the performance gap between the standalone SLM and LLM without incurring the cost of full LLM decoding. On AIME, Qwen3-1.7B achieves an average accuracy of 7.78, while Qwen3-14B reaches 22.22. GRACE improves the SLM to 17.78, recovering approximately 69% of this performance gap while requiring only 22.24 TFLOPs, compared with 132.88 TFLOPs for standalone Qwen3-14B decoding. A similar pattern is observed with Qwen3-30B-A3B. GRACE reaches 16.67 accuracy, recovering about two-thirds of the gap between the SLM and LLM, while requiring only 15.05 TFLOPs. These results suggest that GRACE does not need to reproduce the full computation of the collaborating LLM to benefit from its stronger representations. Instead, latent interaction can selectively transfer useful task-relevant information to improve SLM generation.

MoE models provide particularly compute-efficient collaboration. The results with Qwen3-30B-A3B further highlight the benefit of combining GRACE with sparse MoE architectures. Despite its larger nominal model size, Qwen3-30B-A3B requires substantially less computation than the dense Qwen3-14B and Qwen3.5-9B models due to sparse expert activation. This advantage carries over to collaboration. On instruction-following tasks, GRACE with Qwen3-30B-A3B requires only 1.56, 1.98, 4.70, and 3.62 TFLOPs on IFEval, IFBench, LiveBench-IF, and MulDimIF, while consistently improving the standalone SLM. The effect is even more pronounced on mathematical reasoning, where the average AIME accuracy increases from 7.78 to 16.67 while total computation decreases slightly from 15.70 to 15.05 TFLOPs. This suggests that sparse MoE models are particularly suitable collaborators for GRACE, providing strong latent guidance while keeping the additional LLM computation small.

Detailed task-level results reveal how GRACE improves generation. On instruction-following benchmarks, GRACE consistently improves both prompt-level and constraint-level accuracy, indicating that its benefits extend beyond satisfying isolated constraints. For example, with Qwen3-14B, GRACE improves MulDimIF P-Acc. from 32.92 to 54.42 and C-Acc. from 65.67 to 81.72, while LiveBench-IF improves from 50.00 to 60.50 in P-Acc. and from 78.68 to 84.26 in C-Acc. The simultaneous gains under both metrics suggest that latent guidance helps the SLM more reliably coordinate multiple constraints and satisfy complex instructions as a whole. On mathematical reasoning, the detailed AIME results provide complementary evidence. With Qwen3-30B-A3B, GRACE improves accuracy on AIME 2024, 2025, and 2026 while keeping the computation at or below standalone SLM decoding on all three subsets. In particular, the corresponding TFLOPs change from 18.33 to 18.26, 13.57 to 11.90, and 15.21 to 15.00. This consistent pattern supports the observation that LLM guidance can steer the SLM toward more effective reasoning trajectories, reducing computation spent on redundant generation while improving solution quality.

Table 9: **Detailed performance and efficiency comparison on instruction-following benchmarks based on the Qwen family.** P-Acc. and C-Acc. denote prompt-level and constraint-level accuracy, respectively. Computational cost is measured in TFLOPs.

	IFEval			IFBench			LiveBench-IF			MulDimIF		
Methods	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**
Vanilla Language Model												
Qwen3-4B	78.37	84.53	3.13	22.33	23.84	3.81	57.50	81.90	6.16	53.92	80.06	4.36
Qwen3-1.7B + Qwen3-14B												
Qwen3-1.7B	68.02	75.78	1.26	17.67	19.77	1.65	50.00	78.68	2.54	32.92	65.67	1.78
Qwen3-14B	83.55	89.09	10.55	26.67	29.65	12.91	69.50	88.49	21.34	59.58	84.67	15.70
EAGLE-3* (14B)	84.29	89.45	37.10	27.67	29.94	38.59	68.00	88.16	47.69	59.75	84.64	39.11
RouteLLM	69.50	78.06	2.34	20.00	22.38	4.61	52.50	79.53	**4.72**	51.00	80.12	9.85
Jev†	70.98	79.50	3.30	20.00	23.55	9.29	**63.50**	**86.13**	11.91	37.92	71.53	6.57
C2C	68.02	76.26	3.28	19.67	22.09	4.33	50.00	79.70	15.22	44.83	76.10	12.27
GRACE (Ours)	**74.12**	**81.65**	**1.94**	**24.67**	**25.58**	**2.43**	60.50	84.26	6.36	**54.42**	**81.72**	**4.95**
Qwen3-1.7B + Qwen3-30B-A3B												
Qwen3-1.7B	68.02	75.78	1.26	17.67	19.77	1.65	50.00	78.68	2.54	32.92	65.67	1.78
Qwen3-30B-A3B	82.99	88.37	2.35	28.00	30.81	2.88	68.50	87.65	4.68	58.08	83.40	3.38
EAGLE-3* (30B)	81.89	87.53	3.97	28.33	30.81	4.52	68.00	86.63	6.43	57.67	83.01	5.09
RouteLLM	69.69	77.46	**1.41**	19.67	22.38	**1.96**	50.50	78.85	**2.84**	48.50	78.54	2.78
Jev†	69.87	78.42	1.42	18.67	23.55	2.49	56.50	82.23	3.41	35.33	69.50	**2.28**
C2C	66.17	74.58	2.10	20.33	22.67	2.74	47.50	74.62	5.70	36.42	69.62	5.02
GRACE (Ours)	**73.20**	**80.46**	1.56	**29.00**	**31.40**	1.98	**59.50**	**82.57**	4.70	**53.08**	**81.00**	3.62
Qwen3-1.7B + Qwen3.5-9B												
Qwen3-1.7B	68.02	75.78	1.26	17.67	19.77	1.65	50.00	78.68	2.54	32.92	65.67	1.78
Qwen3.5-9B	82.07	87.41	5.61	33.00	35.47	7.26	65.00	87.65	11.55	59.42	84.04	7.86
EAGLE-3* (9B)	81.33	86.93	8.63	33.67	36.34	17.64	65.00	87.31	22.01	59.75	83.95	18.17
RouteLLM	70.79	78.54	1.79	20.67	23.26	3.15	50.50	79.36	**3.61**	49.00	78.95	5.19
Jev†	71.90	**79.74**	2.21	23.33	**26.74**	5.62	**62.50**	**86.46**	7.13	37.42	71.12	**3.92**
GRACE (Ours)	**73.20**	79.74	**1.67**	**27.33**	25.29	**2.12**	56.00	81.56	5.36	**52.58**	**81.51**	4.11

† Jev API is used and its own computational cost is excluded. ∗ denotes results obtained with vLLM inference acceleration framework.

B Dataset Information and Construction Details

B.1 Dataset Information

Training data construction. We primarily use IF-RLVR (Pyatkin et al., 2026) ,RLVR-IFEval (Lambert et al., 2024), and the training split of MulDimIF (Ye et al., 2026) three instruction-following datasets consisting of prompts with verifiable constraints. Since these datasets provide prompts rather than decoding trajectories, we construct pair-specific supervision for collaborative training. To further adapt GRACE to mathematical ability, we sample 20,000 examples from NuminaMath-1.5 (Li et al., 2024a) while strictly excluding all AIME problems from 2024 to 2026 to avoid evaluation contamination.

Evaluation metrics. We evaluate instruction-following performance using prompt accuracy (P-Acc.) and constraint or instruction accuracy (C-Acc.). Prompt accuracy considers a response correct only when all constraints specified by the corresponding prompt are satisfied, providing a strict measure of complete instruction compliance. Constraint accuracy instead measures the fraction of individual constraints that are successfully satisfied, offering a finer-grained view of instruction-following performance. To further characterize performance relative to inference computation, we report a modified *Compute-Normalized Gain* (CNG*), defined as the accuracy improvement over the standalone SLM normalized by the absolute difference in inference compute:

$$\mathrm{CNG}^* = \frac{\mathrm{Acc.}_M - \mathrm{Acc.}_S}{|\mathrm{TFLOPs}_M - \mathrm{TFLOPs}_S|}, \tag{7}$$

where M denotes the evaluated method and S denotes the corresponding standalone SLM. A higher CNG* indicates a larger performance improvement per unit of absolute inference-compute difference.

Table 10: **Detailed performance and efficiency comparison on mathematical reasoning benchmarks based on the Qwen family.** Acc. denotes problem-level accuracy, and computational cost is measured in TFLOPs.

	AIME 2024		AIME 2025		AIME 2026	
Methods	**Acc.↑**	**TFLOPs↓**	**Acc.↑**	**TFLOPs↓**	**Acc.↑**	**TFLOPs↓**
Vanilla Language Model						
Qwen3-4B	23.33	43.22	13.33	47.14	13.33	45.21
Qwen3-1.7B + Qwen3-14B						
Qwen3-1.7B	6.67	18.33	10.00	13.57	6.67	15.21
Qwen3-14B	26.67	155.11	23.33	101.30	16.67	142.23
EAGLE-3* (14B)	30.00	279.93	20.00	230.41	16.67	255.44
RouteLLM	16.77	85.01	**23.30**	76.45	6.67	113.92
Jev†	10.00	122.76	13.33	87.93	**20.00**	116.29
C2C	13.33	30.48	6.67	27.74	6.67	29.26
GRACE (Ours)	**20.00**	**26.18**	16.67	**18.87**	16.67	**21.68**
Qwen3-1.7B + Qwen3-30B-A3B						
Qwen3-1.7B	6.67	18.33	10.00	13.57	6.67	15.21
Qwen3-30B-A3B	26.67	39.89	16.67	28.08	20.00	34.78
EAGLE-3‡ (30B-A3B)	23.33	51.36	16.67	41.05	16.67	35.39
RouteLLM	**20.00**	30.60	13.33	23.43	16.67	28.28
Jev†	13.33	31.55	**16.67**	29.05	**20.00**	29.84
C2C	6.67	30.56	10.00	23.46	13.33	24.18
GRACE (Ours)	16.67	**18.26**	**16.67**	**11.90**	16.67	**15.00**
Qwen3-1.7B + Qwen3.5-9B						
Qwen3-1.7B	6.67	18.33	10.00	13.57	6.67	15.21
Qwen3.5-9B	76.67	92.22	43.33	99.57	56.67	97.38
EAGLE-3‡ (9B)	63.33	184.40	46.67	191.19	56.67	193.09
RouteLLM	**43.33**	50.61	**36.67**	70.42	**46.67**	78.49
Jev†	**43.33**	78.78	33.33	96.52	40.00	84.76
GRACE (Ours)	23.33	**16.93**	16.67	**13.41**	13.33	**14.64**

† Jev API is used and its own computational cost is excluded.
∗ denotes results obtained with vLLM inference acceleration framework.

Table 11: **Detailed performance and efficiency comparison on instruction-following benchmarks across a heterogeneous SLM–LLM pair.** P-Acc. and C-Acc. denote prompt-level and constraint-level accuracy, respectively. Computational cost is measured in TFLOPs. † denotes experiments using the Jev API, excluding Jev's own computational cost.

	IFEval			IFBench			LiveBench-IF			MulDimIF		
Methods	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**	**P-Acc.↑**	**C-Acc.↑**	**TFLOPs↓**
Ministral3-3B	54.71	67.27	3.51	19.33	20.93	4.16	23.50	61.08	5.56	40.50	72.20	4.59
Qwen3-14B	83.55	89.09	10.55	26.67	29.65	12.91	69.50	88.49	21.34	59.58	84.67	15.70
RouteLLM	57.86	69.66	4.38	21.33	25.00	6.47	29.00	65.14	**7.42**	53.00	81.70	11.08
Jev†	56.19	69.30	5.11	22.67	**26.16**	10.02	39.50	74.96	12.26	43.50	75.24	**7.94**
C2C	**70.43**	**78.54**	5.16	22.33	24.71	6.80	38.00	68.70	18.87	44.25	75.07	14.75
GRACE (Ours)	67.47	76.38	**3.52**	**24.33**	**26.16**	**4.84**	**46.00**	**76.14**	10.99	**57.25**	**82.70**	8.68

† Jev API is used and its own computational cost is excluded.

B.2 SLM-LLM CONFIGURATIONS

We select language models spanning different scales, architectures, tokenization spaces, modalities, and inference precisions, as summarized in Table 13. The Qwen series provides controlled within-family configurations across model scales and architectures, including dense models, the Qwen3-30B-A3B MoE variant, and the newer Qwen3.5-9B for evaluating collaboration across model versions. We further include Ministral3-3B and Gemma4-E2B to construct cross-family settings with different architectures and tokenizers. Together, these configurations introduce substantial diversity in model depth, hidden dimension, vocabulary space, and runtime precision, allowing us to evaluate whether GRACE remains effective across heterogeneous SLM–LLM configurations rather than relying on closely matched model pairs.

Table 12: **Detailed performance and efficiency comparison on mathematical reasoning benchmarks across a heterogeneous SLM–LLM pair.** Acc. denotes problem-level accuracy, and computational cost is measured in TFLOPs.

	AIME 2024		AIME 2025		AIME 2026	
Methods	**Acc.↑**	**TFLOPs↓**	**Acc.↑**	**TFLOPs↓**	**Acc.↑**	**TFLOPs↓**
Ministral3-3B-Instruct	26.67	46.89	16.67	47.01	13.33	45.93
Qwen3-14B	26.67	155.11	23.33	101.30	16.67	142.23
RouteLLM	23.33	101.02	**23.33**	89.32	10.00	121.69
Jev†	**26.67**	94.34	20.00	70.84	**16.67**	114.16
C2C	6.67	57.38	6.67	**43.53**	10.00	**47.30**
GRACE (Ours)	20.00	**53.10**	13.33	48.65	**16.67**	48.61

† Jev API is used and its own computational cost is excluded.

Table 13: **Configurations of backbone models used in our experiments.** For MoE models, the parameter count is reported as total/active parameters. For Gemma 4 E2B, total/effective parameters are reported. For multimodal models, the number of layers and hidden dimension refer to the language-model backbone.

	Model Architecture					Tokenization		Runtime
Model	**Modality**	**Arch.**	**Params.**	**Layers**	**Hidden Dim.**	**Tokenizer**	**Vocab Size**	**Precision**
Qwen3-1.7B	Text	Dense	1.7B	28	2048	Qwen3	151,936	BF16
Qwen3-4B	Text	Dense	4.0B	36	2560	Qwen3	151,936	BF16
Qwen3-14B	Text	Dense	14.8B	40	5120	Qwen3	151,936	BF16
Qwen3-30B-A3B	Text	MoE	30.5B / 3.3B	48	2048	Qwen3	151,936	BF16
Ministral3-3B-Instruct	Multimodal	Dense	3.4B LM + 0.4B V	26	3072	Tekken	131,072	FP8
Gemma4-E2B	Multimodal	Dense	5.1B (2.3B eff.)	35	1536	Gemma	262,144	BF16
Qwen3.5-9B	Multimodal	Dense	9.0B	32	4096	Qwen3.5	248,320	BF16

B.3 Implementation Details

Throughout training, both the SLM and LLM backbones remain frozen, and only the context aligner and representation augmentation blocks are optimized. We set the temporal buffer size to $b = 4$ and partition both models into $K = 4$ depth-aligned computation grades, with representation augmentation performed at the first three grades.

We first warm up the context aligner for 15 epochs using AdamW with a global batch size of 32 and a weight decay of 0.1. The learning rate is linearly increased from 5×10^{-6} to 5×10^{-5} and then cosine-decayed to 5×10^{-7}. We subsequently perform collaborative training using the same global batch size and learning-rate schedule, while retaining the auxiliary context aligner objective with $\lambda_{\mathrm{CA}} = 0.1$. To simulate the temporal mismatch introduced by asynchronous execution, we use a base delay of 2 tokens and vary the delay according to a Bernoulli distribution with probability $p = 0.5$. For the mathematical reasoning experiments, we initialize from the instruction-following checkpoint and further fine-tune GRACE for 10 epochs on the constructed mathematical training data. The learning rate is linearly increased from 1×10^{-6} to 1×10^{-5} and subsequently cosine-decayed to 1×10^{-7}. Training is conducted on 8 NVIDIA RTX PRO 6000 GPUs. For reproducibility and consistent comparison, we use greedy decoding for the main experiments. For models that support explicit thinking modes, we use the non-thinking setting throughout evaluation.