SADAS: Enabling Efficient On-Device Reasoning via State-Aware Dynamic Attention Scheduling

Anonymous ACL submission

Abstract

The deployment of reasoning-intensive LLMs on edge devices is severely hampered by the memory bandwidth bottlenecks inherent in processing long Chain-of-Thought sequences. To address this, we propose State-Aware Dynamic Attention Scheduling (SADAS), a novel framework that aligns computational complexity with cognitive phases. SADAS leverages control tokens to autonomously toggle between bandwidth-efficient Sliding Window Attention for intermediate reasoning steps and high-fidelity Full Attention for final answer integration. This mechanism reduces memory pressure without sacrificing global context. Experimental results on commodity CPUs demonstrate that SADAS transforms unresponsive long-chain reasoning into a real-time capability, achieving up to 3.88× end-to-end speedups. Crucially, SADAS-1.7B enjoys superior performance that matches or exceeds the results of leading 8B-scale reasoning models on challenging reasoning benchmarks like AIME24, while maintaining robust agentic instruction-following capabilities. To the best of our knowledge, SADAS is the first architecture to implement token-level dynamic attention mixing, establishing a new paradigm for responsive on-device intelligence.

1 Introduction

The paradigm of Natural Language Processing (NLP) is being profoundly reshaped by Large Language Models (LLMs). Currently, a major research direction clearly points towards endowing models with complex reasoning capabilities, a cutting-edge field exemplified by works such as OpenAI's o1 (Jaech et al., 2024), Alibaba's Qwen3 (Yang et al., 2025), and DeepSeek's DeepSeek-R1 (Guo et al., 2025). The core mechanism of these models involves simulating and executing complex cognitive tasks by generating explicit Chain-of-Thought (CoT) sequences (Wei et al., 2022), leading to breakthrough progress in various logic-intensive benchmarks. Crucially, this deep reasoning capability serves as the foundational engine for a wide range of downstream applications, from solving intricate mathematical problems to empowering autonomous agents on edge devices that must plan and execute complex user instructions (Xi et al., 2025).

However, this transition to generative reasoning fundamentally shifts the computational bottleneck. The demand for deep reasoning renders LLMs increasingly decoding-oriented. Before generating a final answer, models must produce extensive internal thought processes token-by-token. For a CoT of length N, the decoding process necessitates N forward passes, each involving self-attention with $O(N^2)$ complexity (Chou et al., 2024). While manageable on cloud GPUs, this quadratic cost becomes prohibitive when deploying reasoning models on resource-constrained edge devices (e.g., smartphones, laptops). On commodity CPUs with limited memory bandwidth, generating long reasoning chains results in unacceptable latency, making on-device reasoning capabilities practically unresponsive. Analyzing its output structure more deeply, this process functionally bifurcates the model's generation task into two implicit stages: an exploratory intermediate thought stage, which focuses on rapid generation and path exploration; and an integrative final answer stage, which emphasizes reviewing and precisely consolidating global information. This raises a core architectural design question: Is a single homogeneous attention architecture optimal for these two functionally distinct computational stages?

The Linear Attention mechanism (Shen et al., 2021; Han et al., 2024; Ma et al., 2021) offers a highly promising direction to address this challenge, with its $O(N)$ computational complexity significantly superior to the $O(N^2)$ of standard attention mechanisms. However, linear attention

models exhibit a crucial bottleneck: their limited ability to recall contextual information. Performance significantly decreases, particularly in tasks that require precise retrieval of specific facts or maintenance of long-term dependencies (Fan et al., 2025). This inherent forgetfulness characteristic makes them unsuitable for independently undertaking complex reasoning tasks that require a global perspective.

To address this inherent architectural contradiction, we propose a novel framework, SADAS, which enables state-aware dynamic attention mechanism switching at the token granularity. SADAS can dynamically reconfigure its core computational mode during its generation process. Utilizing predefined control tokens, the model can seamlessly transition between efficiency-prioritizing Linear Attention and fidelity-prioritizing Full Attention (Zhang et al., 2019). Specifically, during the exploratory phase of generating intermediate thought steps, the model employs Sliding Window Attention (SWA) (Fu et al., 2025) to maximize computational efficiency; while in the integrative phase, when global information needs to be consolidated to produce the final answer, it switches back to the Full Attention mechanism to ensure maximum accuracy and coherence. This design parallels human cognitive processes, where stepwise reasoning relies on a transient working memory to focus on immediate logical antecedents rather than re-evaluating the entire context history. Furthermore, this approach is intrinsically tailored for edge CPUs (Zhang et al., 2025a). By restricting attention to a local window, SADAS significantly alleviates the memory bandwidth bottleneck, which constitutes the primary constraint for fast decoding on consumer-grade devices. Compared to the severe latency faced by standard Full Attention models during long sequence decoding and the performance degradation of approximate attention models due to information loss, SADAS achieves a superior balance that renders complex reasoning practically viable on mobile and personal computing hardware. Our main contributions are summarized as follows:

- We propose a hardware-aware dynamic inference framework that adapts the attention mechanism to the intrinsic phase of generation. This effectively decouples the computational cost of the exploratory phase from the integrative phase to suit bandwidth-limited edge devices.

- We validate the framework on commodity CPUs using llama.cpp. Experimental results demonstrate that SADAS achieves 2x–3x end-to-end speedups compared to full-attention baselines, transforming long-chain reasoning from an unresponsive feature into a real-time capability for on-device agents.

- We conduct comprehensive ablation studies demonstrating that SADAS maintains robust performance not only on mathematical benchmarks but also on instruction following tasks. This proves its potential as a general-purpose engine for next-generation on-device intelligence.

2 Related Work

2.1 Reasoning Models and On-Device Agents

Integrating reasoning capabilities into LLMs has evolved from instruction tuning (Zhao et al., 2023; Zhang et al., 2023, 2025b) to CoT prompting (Wei et al., 2022). CoT guides models to generate explicit reasoning paths, serving as the cognitive foundation for on-device agents to perform autonomous planning. However, CoT training faces challenges like inefficient gradient estimation (Yao et al., 2025), and its necessity for all tasks remains debated (Ma et al., 2025).

State-of-the-art small reasoning models suitable for edge deployment, such as Qwen3 (Yang et al., 2025) and DeepSeek-R1-Distill (Guo et al., 2025), integrate CoT with self-reflection. Despite their performance, the computational overhead of long reasoning chains creates severe bottlenecks for edge devices. Unlike cloud infrastructure, memory-constrained edge hardware lacks the bandwidth to sustain the quadratic complexity of self-attention during lengthy decoding, causing thinking latency that renders local models unresponsive (Jiang et al., 2025). While architectures like Hunyuan-TurboS (Liu et al., 2025) attempt mode switching, they do not fundamentally resolve the redundancy of long chains on local devices. Our work mitigates this by dynamically adapting attention to the reasoning stage.

2.2 Efficient Attention Mechanisms

The $\mathcal{O}(N^2)$ complexity of standard self-attention (Ho et al., 2024) consumes excessive memory band-

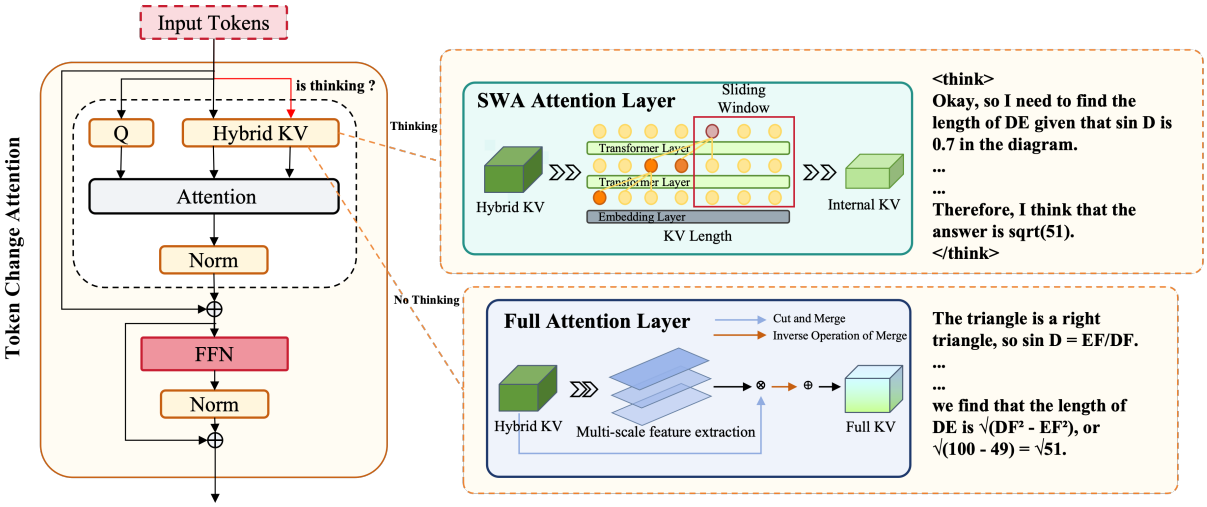

Figure 1: Architectural Overview of SADAS

width on edge devices, dominating decoding latency. Various mechanisms aim to alleviate this.

Linear Attention (e.g., RWKV (Peng et al., 2023), Mamba (Gu and Dao, 2023), Lightning Attention-2 (Qin et al., 2024)) reduces complexity to $\mathcal{O}(N)$. While efficient for mobile deployment, these methods often suffer from forgetfulness in high-fidelity recall tasks (Sun et al., 2025), posing risks for agents executing complex instructions. Similarly, Sparse Attention variants like SWA (Fu et al., 2025) restrict attention to local windows. Although cache-friendly for edge CPUs, SWA's limited receptive field degrades performance in global reasoning tasks (Fan et al., 2025).

Hybrid approaches such as MoBA (Lu et al., 2025) and MoM (Du et al., 2025) combine sparse and dense attention. However, these typically employ spatial mixing and do not take advantage of the stage-specific structure of reasoning. They do not dynamically minimize resources during the transient thinking phase. SADAS addresses this via temporal mixing, adapting attention modes to the model's real-time cognitive state to maximize edge hardware potential.

3 Method

To achieve an optimal balance between the exploratory and integrative stages of reasoning on resource-constrained devices, we propose a State-Aware Dynamic Attention Mechanism. The core idea of this framework is to enable the model to reconfigure its core attention computation paradigm in real-time at the token level during autoregressive generation, based on its current cognitive task.

3.1 Architectural Overview

The overall architectural design of SADAS is based on a key insight: complex reasoning processes are naturally divided into two cognitive stages, exploratory thinking akin to agentic planning and integrative answering corresponding to execution. To map this cognitive model onto a computational architecture, we designed a dynamic switching framework, as depicted in Fig. 1. This framework seamlessly integrates two computation modes: bandwidth-efficient SWA for the exploratory phase of quickly generating a Chain-of-Thought, and information-fidelity preserving Full Attention for the final answer stage, which requires integrating global information.

To enable autonomous switching between modes, we introduce a set of predefined control tokens: `<think>` and `</think>`. These tokens serve as explicit signals for the model's internal state. When the model generates `<think>`, it automatically enters the hardware-friendly SWA mode; upon generating `</think>`, it switches back to the high-fidelity Full Attention mode. This design empowers the model with autonomous control over its computational flow.

At the implementation level, SADAS utilizes a unified and append-only KV cache (Zhou et al., 2024). In SWA mode, attention computation only accesses the most recent portion of the cache; whereas in Full Attention mode, it can access the entire historical data within the cache. This design is particularly advantageous for CPU-based inference as it eliminates the costly memory copy operations typically associated with cache eviction

in standard sliding window implementations. It significantly simplifies architectural complexity, ensuring a high degree of continuity and parallelism in the computational process.

The complete workflow of SADAS is as follows: when the model needs to perform complex reasoning, it first generates the <think> token, which triggers the framework to switch its computational core to the SWA Attention Layer. In this mode, the model efficiently generates a series of intermediate thought steps, minimizing memory bandwidth consumption. When the thinking process concludes, the model generates the </think> token, marking the completion of the exploratory phase. At this point, the framework seamlessly switches the computational core back to the Full Attention Layer, and the model begins to integrate all contextual information, including the thought chain, to finally generate an accurate answer.

3.2 State-Aware Dynamic Attention Mechanism

To realize the architecture described, we designed a dynamic scheduling mechanism that relies on precise tracking of each sequence's cognitive state and is formalized by dynamic attention computation graph reconfiguration. For the i-th sequence in a batch, at each generation timestep t, we define its cognitive state $S_t^{(i)}$ and a termination state latch $F_t^{(i)}$, where $S_t^{(i)}, F_t^{(i)} \in \{0, 1\}$. $S_t^{(i)}$ indicates whether the model is currently in thinking mode and $F_t^{(i)}$ marks whether the entire thinking process has concluded.

3.2.1 State Transition Dynamics

The model's state evolution is driven by the previously generated token $y_{t-1}^{(i)}$. Let y_{think} and $y_{\text{end\_think}}$ denote the IDs of the control tokens, respectively. The cognitive state $S_t^{(i)}$ represents whether the model is in the thinking mode. Its transition logic is defined as follows:

$$S_t^{(i)} = \begin{cases} 1 & \text{if } y_{t-1}^{(i)} = y_{\text{think}} \\ 0 & \text{if } y_{t-1}^{(i)} = y_{\text{end\_think}} \\ S_{t-1}^{(i)} & \text{otherwise} \end{cases} \quad (1)$$

where $S_t^{(i)} = 1$ corresponds to the thinking mode, and $S_t^{(i)} = 0$ to the integrative mode. To ensure that once reasoning is complete, the model stably enters the high-fidelity integrative mode, we introduce the termination state latch $F_t^{(i)}$. This is a unidirectional trigger that, once activated, remains persistent. Its update rule is as follows:

$$F_t^{(i)} = F_{t-1}^{(i)} \vee (y_{t-1}^{(i)} = y_{\text{end\_think}}) \quad (2)$$

where $\vee$ denotes the logical OR operation, with initial state $F_0^{(i)} = 0$. This latch ensures that after a complete reasoning chain, the model will be locked into the integrative mode and will not switch back to SWA.

3.2.2 State-Dependent Attention Selection

The computation of the standard self-attention mechanism can be abstractly represented as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (3)$$

The core innovation of our mechanism lies in transforming the key (K) and value (V) matrices provided to this function from static, undifferentiated historical records into a dynamically configured context set based on the cognitive state $F_t^{(i)}$. This process can be viewed as a real-time reconfiguration of the attention computation graph at each decoding step.

For the complete key-value history of sequence i at timestep t, we define it as $\mathcal{H}_t^{(i)} = \{(k_j^{(i)}, v_j^{(i)})\}_{j=1}^t$. When the model enters the high-fidelity integrative mode, $F_t^{(i)} = 1$, it must have access to the global context for accurate recall and integration. In this computational path, the attention mechanism is granted unrestricted access to the complete history $\mathcal{H}_t^{(i)}$. At this point, the effective key-value pairs $(\mathbf{K}_{\text{Full}}^{(i)}, \mathbf{V}_{\text{Full}}^{(i)})$ are formed by concatenating all key and value tensors from $\mathcal{H}_t^{(i)}$ along the sequence dimension.

Conversely, when the model is in the computationally more efficient exploratory thinking stage, $F_t^{(i)} = 0$, to accelerate thinking, its context access is actively restricted to a local window of a predefined size W_c. In this path, the effective key-value pairs $(\mathbf{K}_{\text{SWA}}^{(i)}, \mathbf{V}_{\text{SWA}}^{(i)})$ consist only of the most recent subset of the history $\mathcal{H}_t^{(i)}$. This dynamic constraint on historical records is crucial for achieving a trade-off between computational efficiency and model performance.

Therefore, the final key-value set $(\mathbf{K}_{\text{eff}}^{(i)}, \mathbf{V}_{\text{eff}}^{(i)})$ used by the effective attention computation $\text{Attn}_{\text{eff}}^{(l,i,t)}$ of the l decoding layer can be formally

described as a conditional selection function controlled by the state $F_t^{(i)}$:

$$(\mathbf{K}_{\text{eff}}^{(i)}, \mathbf{V}_{\text{eff}}^{(i)}) = \begin{cases} (\mathbf{K}_{\text{SWA}}^{(i)}, \mathbf{V}_{\text{SWA}}^{(i)}) & \text{if } F_t^{(i)} = 0 \\ (\mathbf{K}_{\text{Full}}^{(i)}, \mathbf{V}_{\text{Full}}^{(i)}) & \text{if } F_t^{(i)} = 1 \end{cases} \tag{4}$$

Finally, the output of this layer $h_t^{(l,i)}$ is computed based on this dynamically selected context set:

$$h_t^{(l,i)} = \text{Attention}(q_t^{(l,i)}, \mathbf{K}_{\text{eff}}^{(i)}, \mathbf{V}_{\text{eff}}^{(i)}) \tag{5}$$

In terms of implementation, this high-level stateful logic is elegantly mapped onto low-level computational optimizations. The state variable F_t is propagated across model layers via a boolean flag, ultimately passing a concrete window size to the underlying attention implementation to execute the logic of Equation 4. This achieves a direct mapping from abstract semantic states to concrete hardware-accelerated computational paths, thereby enabling dynamic control over the model's computation mode without introducing significant architectural complexity.

4 Experiments

To empirically evaluate the effectiveness and efficiency of the SADAS framework, this section presents a series of experiments. We first detail the experimental setup, including model initialization, fine-tuning scheme, and evaluation benchmarks. Subsequently, we present a performance comparison of SADAS with current mainstream reasoning models and architectural variants. Finally, through a series of ablation studies, we deeply analyze the impact of the framework's key components and hyperparameters.

4.1 Experimental Setup

4.1.1 Model Initialization and Baselines

SADAS was initialized by distilling and mapping weights from the pre-trained Qwen3 model (Yang et al., 2025). For comprehensive assessment, we compared SADAS against a representative set of baseline models, including top-tier closed-source models like GPT-4o (Hurst et al., 2024) and Claude-3.5-Sonnet. Open-source baselines included advanced reasoning models such as DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), Qwen3 (Yang et al., 2025), its pure Sliding Window Attention variant SWAQwen, and the latest hybrid architecture reasoning model M1-3B (Wang et al., 2025).

4.1.2 Experimental Setup

Lightweight Adaptation Fine-tuning. To enable the SADAS framework, we performed a lightweight fine-tuning stage (Team, 2025). We emphasize that this process is designed strictly for mechanism adaptation rather than knowledge injection. Its core objectives are twofold: (1) Cognitive Rhythm Alignment: enabling the model to learn the timing for generating control tokens; and (2) Computation-Mode Adaptation: allowing the model weights to adapt to the receptive field shifts between Sliding Window Attention and Full Attention. Detailed training parameters and settings are provided in the Appendix B.

4.1.3 Benchmark

Benchmarks and Metrics. To provide a holistic assessment of SADAS as a foundation for on-device agents, we evaluated the model across two distinct capability dimensions:

Deep Reasoning: We utilized the challenging AIME24 and AIME25 datasets, employing the pass@k metric (with $k = 8$) (Chen et al., 2021; Brown et al., 2024) to measure robust mathematical problem-solving. We also included Math-500 (Hendrycks et al., 2021), a curated collection of 500 diverse mathematical problems, to assess breadth.

General and Agentic Capabilities: To verify that the efficiency gains of SADAS do not compromise the general knowledge and strict instruction adherence required for agents, we incorporated MMLU (Hendrycks et al., 2020) for general world knowledge, BBH (Suzgun et al., 2023) for complex symbolic reasoning, and IFEval (Zhou et al., 2023) to specifically evaluate prompt compliance and instruction following fidelity.

All evaluations were conducted using the EvalScope framework, scoring mathematical problems based on strict equivalence. For consistency across accuracy benchmarks, the SWA layer window size was fixed at 4096 tokens, using standard decoding parameters (temperature 0.7, top-p 0.8) (Guo et al., 2025; Luo et al., 2025)

On-Device Inference Setup. To evaluate practical deployment efficiency, we conducted end-to-end latency tests on two representative consumer platforms: an Intel Core i5-12600KF desktop CPU and a Qualcomm Snapdragon 8 Gen 3 mobile SoC, utilizing the llama.cpp engine. All inference speed tests were conducted using Q4_0 quantization precision. To simulate realistic edge memory con-

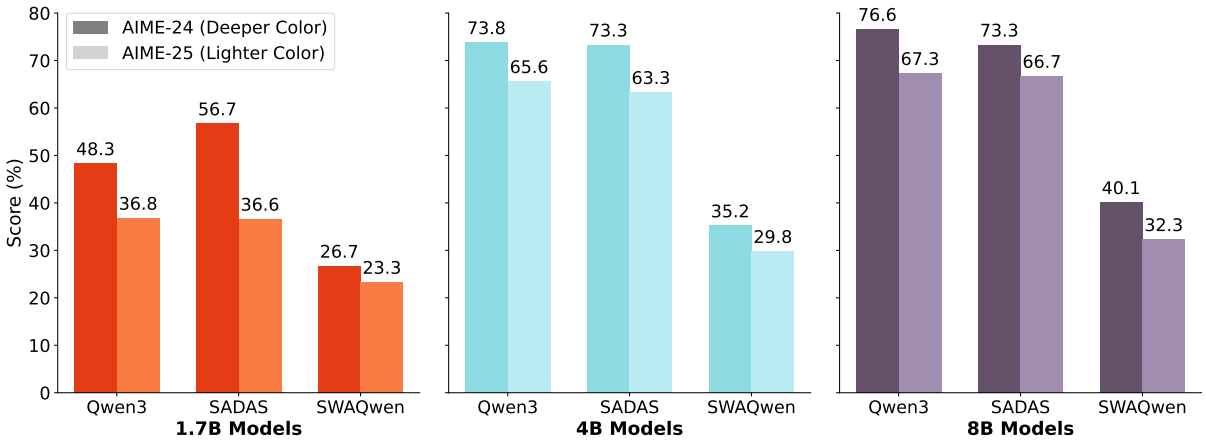

Figure 2: Comparison of SADAS and baseline Qwen3 variants on AIME24 and AIME25 datasets.

Table 1: Reasoning and General Evaluation Results for SADAS and Baseline Models

Model	Math500	AIME-24	AIME-25	MMLU	BBH	IFeval
GPT-4o-0513	74.6	9.3	11.6	87.2	–	84.3
Claude-3.5-Sonnet-1022	78.3	16.0	7.4	88.3	–	86.5
M1-3B	81.7	23.0	22.0	–	–	–
QWEN3-1.7b	89.0	48.3	**36.8**	55.7	48.2	33.6
DeepSeek-R1-Distill-Llama-8B	89.1	50.4	32.9	54.2	60.7	43.9
DeepSeek-R1-Distill-Qwen-1.5B	83.9	28.9	23.5	35.2	41.9	29.6
SADAS-1.7B	**89.4**	**56.7**	36.6	55.7	47.3	41.0
SADAS-4B	**91.8**	**73.3**	**63.3**	68.4	73.5	45.4

straints, the sliding window size was set to 1024 tokens. Tests were performed with a batch size of 1 and a fixed prompt length of 2048 tokens across varying output lengths from 1024 to 32768 tokens.

4.2 Main Results Comparing with Baselines

As shown in Table 1, SADAS shows excellent performance on all benchmarks. On relatively foundational benchmarks like Math500, SADAS performs comparably with other high-performance models, showing a slight improvement. However, SADAS's advantage becomes particularly prominent when evaluated on benchmarks requiring deeper reasoning capabilities, such as AIME24 and AIME25. SADAS-4B achieved 73.3 on AIME24 and 63.3 on AIME25, significantly outperforming all comparable and even much larger models. Even at the 1.7B parameter scale, SADAS-1.7B's performance surpassed that of other larger models, demonstrating improvements across all three datasets compared to the 8B-parameter DeepSeek-R1-Distill-Llama. Beyond mathematical reasoning, our results further validate the framework's robustness in general and agentic capabilities. On MMLU, SADAS-1.7B achieved 55.7, effectively matching the base model Qwen3-1.7B at 55.7, confirming that the efficiency optimization does not compromise general knowledge. Most notably, on the IFEval benchmark, SADAS-1.7B scored 41.0, marking a substantial improvement over the Qwen3-1.7B baseline of 33.6. This significant gain suggests that the dedicated thinking phase serves as an effective cognitive planning step, thereby enhancing the model's ability to strictly adhere to complex instructions essential for on-device agents.

Table 2: Completion Token Lengths on AIME24

Model	Ave	Max	Min
Qwen3-1.7B	16737	28248	4771
SADAS-1.7B	18883	32768	5544

To further clarify the specific contributions of the SADAS framework, we conducted a direct comparison with the Full Attention model Qwen3 and its pure SWA variant, SWAQwen. As illustrated in Fig. 2, the performance of SADAS consistently positions it between Qwen3 and SWAQwen. No-

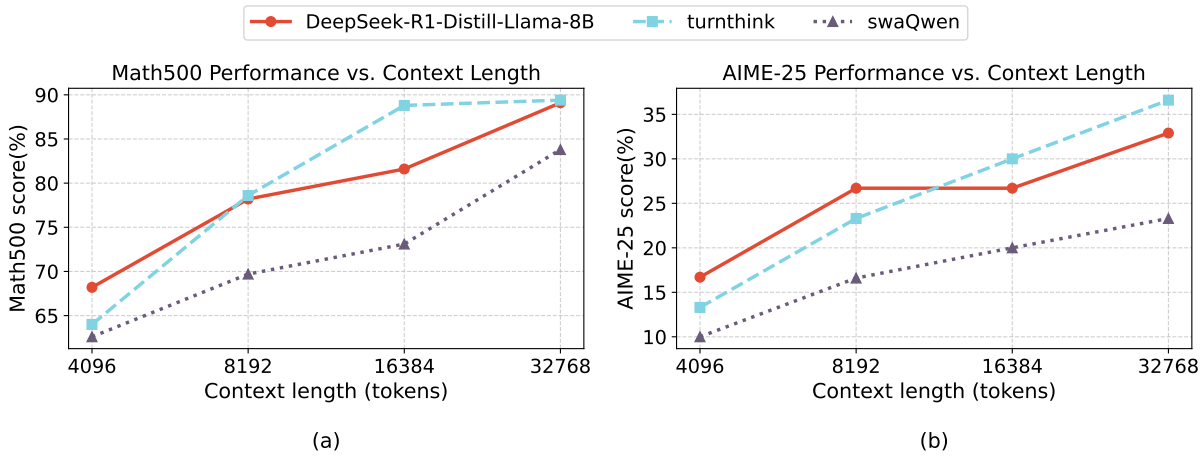

Figure 3: Relationship between Generation Length and Score Accuracy

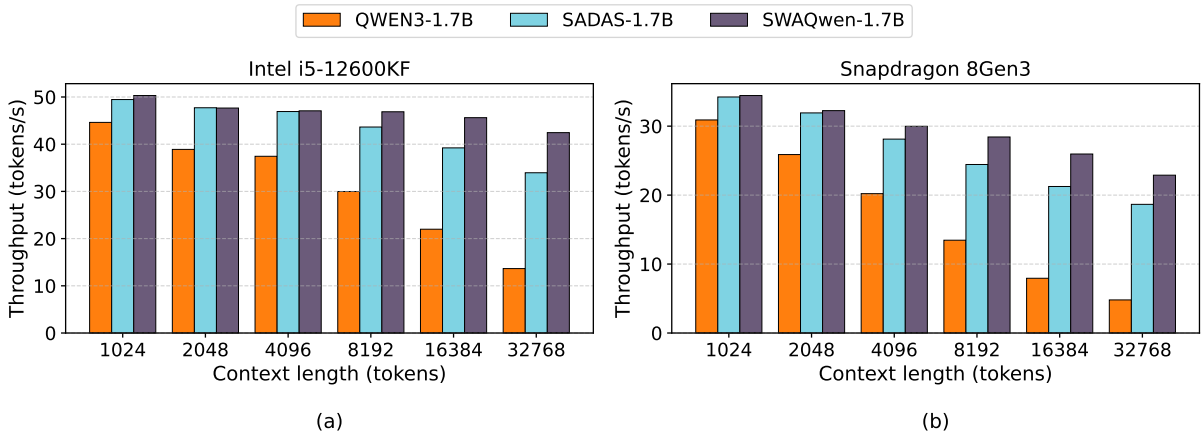

Figure 4: End-to-end inference latency of SADAS and Qwen3 variants under different output lengths (seconds)

tably, at the 1.7B scale, SADAS achieved a nearly 16% lead over Qwen3 on AIME24, representing a substantial improvement. As shown in Table 2, under the same 32k maximum output length constraint, SADAS demonstrated an approximate 12% improvement in the average length of generated answers compared to Qwen3, with maximum and minimum lengths increasing by nearly 16%. This confirms that SADAS successfully achieves an effective trade-off between computational efficiency and model fidelity, and to some extent, unlocks stronger reasoning potential. This is achieved by employing SWA during the thinking phase to accelerate reasoning and utilizing Full Attention for global information integration during the synthesis phase, resulting in only minor performance loss, far superior to pure SWA architectures that completely sacrifice global context.

Table 4 presents a detailed comparison of generation speeds for 1.7B-scale models on representative edge hardware (comprehensive results for PC CPUs are provided in Appendix A). Under the severe memory bandwidth constraints of mobile devices, the speed of the standard Qwen3-1.7B model degrades drastically as the sequence length increases, plummeting to just 4.8 tokens/s at 32,768 tokens. By leveraging hardware-friendly SWA during the thinking phase, SADAS achieves faster end-to-end inference speeds than Qwen3 and its variants on both the Snapdragon 8 Gen 3 and Intel PC CPUs.

Specifically, on the Snapdragon 8 Gen 3 platform, when the output length reaches 32,768 tokens, SADAS maintains a speed of 18.66 tokens/s, achieving an approximate 3.88× speedup compared to Qwen3-1.7B. This effectively transforms long-chain reasoning from an unusable state to a real-time capability. On the Intel PC CPU, at 32,768 tokens, SADAS sustains 33.95 tokens/s, representing an approximate 2.5× speedup over Qwen3's 13.65 tokens/s. Although SWAQwen is marginally faster than SADAS, its significant degra-

dation in accuracy renders it inferior to SADAS in the balance between usability and speed. This underscores the architectural superiority of SADAS: it utilizes an append-only mechanism to avoid the severe accuracy loss characteristic of pure SWA models, confirming its critical role in efficiently processing long-sequence reasoning on edge devices.

4.3 Ablation Study

4.3.1 Impact of Output Length

Figure 3 illustrates the impact of increased output length on complex reasoning. While all models benefit from extended reasoning depth, SADAS demonstrates the most significant efficiency gains. Quantitatively, extending the maximum output length from 4096 to 32768 tokens, SADAS's performance on Math500 improved by 39.7%, outperforming DeepSeek-R1-Distill-Llama-8B (30.6%) and SWAQwen (33.9%). On the more challenging AIME25, SADAS showed a remarkable 175.2% increase, significantly exceeding DeepSeek-R1-Distill-Llama-8B's 97.0% and SWAQwen's 133.0%.

This superior performance extension stems from SADAS's dynamic switching, which efficiently generates long reasoning paths with SWA during the thinking phase and then leverages Full Attention for lossless global recall and integration in the final synthesis. This allows SADAS to optimally translate reasoning depth into problem-solving capabilities, particularly for complex tasks requiring a global perspective, where its performance curve exhibits the steepest growth.

4.3.2 Impact of Window Size

Table 3: Performance under Different Window Sizes

Model	Window Size	Math500	AIME24	AIME25
SADAS	1024	53.7	16.7	13.3
	2048	76.2	26.7	23.3
	4096	89.4	56.7	36.6
SWAQwen	1024	32.4	6.6	3.3
	2048	62.8	10.0	16.7
	4096	83.8	26.7	23.3

Table 3 presents performance results across different window sizes, investigating how the width of local context in SADAS's thinking phase affects model performance. The maximum output length was set to 32,768 tokens. Analysis reveals that increasing the window size significantly benefits both SADAS and pure SWA architectures, underscoring the importance of a wider local receptive field during exploratory thinking. For instance, on AIME24, SADAS's score surged by 239.5% (from 16.7 to 56.7) when the window expanded from 1024 to 4096.

Crucially, SADAS utilizes this expanded context much more efficiently than pure SWA. The performance gap between SADAS and SWAQwen systematically widens with increasing window size. On AIME24, SADAS's lead over SWAQwen grew from 10.1 points (1024 window size) to 30.0 points (4096 window size), demonstrating SADAS's performance more than doubling SWAQwen's at the largest window size. This widening gap confirms SADAS's hybrid mechanism maximizes the value of the SWA window. The SWA efficiently generates high-quality thought content, which the subsequent Full Attention layer, with its global access, meticulously processes for the final answer. This paradigm, which allocates computational resources to a stage that can be fully leveraged by a subsequent high-fidelity module, is more effective than universally employing a single approximate attention mechanism.

5 Conclusion

We propose SADAS, a dynamic inference framework designed to resolve the memory bandwidth bottleneck of long-chain reasoning on edge devices. By autonomously toggling between bandwidth-efficient sliding window attention and high-fidelity full attention, SADAS aligns computational modes with the model's cognitive phases. Experimental results confirm that SADAS transforms unresponsive reasoning into a real-time mobile capability, achieving accuracy comparable to leading 8B-scale reasoning models. As the first architecture to implement token-level dynamic attention mixing, our work establishes a robust path for next-generation on-device agents that demand both speed and intelligence.

Limitations

While our proposed SADAS framework demonstrates significant improvements in balancing inference efficiency and reasoning accuracy, we acknowledge several limitations that offer avenues for future research.

1. **Static Window Size Configuration:** Currently, SADAS employs a pre-defined fixed

window size (e.g., 1024 or 4096 tokens) for the sliding window attention phase. While experimental results confirm this setting covers the majority of local context required for reasoning steps, a static window may not be theoretically optimal for every specific instance. Future work could explore adaptive mechanisms that dynamically adjust the window size based on real-time token entropy or semantic density, potentially further optimizing the trade-off between memory bandwidth usage and information recall.

2. **Memory Consumption of the KV Cache:** Our implementation of SADAS prioritizes inference speed and architectural simplicity by utilizing a unified, append-only KV cache (DynamicCache). While this approach avoids the latency overhead associated with cache eviction in standard sliding window attention, it comes at the cost of increased memory consumption. The KV cache grows linearly with the length of the entire generated sequence. For extremely long reasoning chains (e.g., far exceeding 32k tokens), this could become a practical bottleneck on hardware with limited VRAM, representing a direct trade-off between speed and memory footprint.

3. **Rigidity of the Bipartite Cognitive Model:** The current SADAS framework is built upon a binary model of cognition, bifurcating the generation process into a single exploratory thinking phase (SWA) and a final integrative answering phase (Full Attention). However, complex reasoning may not always follow such a linear path. It could be iterative, requiring the model to switch back and forth between exploration and integration multiple times. Our current design, with its unidirectional termination latch ($F^{(i)}$), does not support such complex, multi-turn cognitive state transitions, potentially limiting its effectiveness on tasks that require more sophisticated reasoning patterns.

4. **Generalizability to Other Long-Context Tasks:** Our experiments have primarily focused on mathematical and logical reasoning benchmarks, where the Chain-of-Thought paradigm is well-established. The applicability and effectiveness of the SADAS framework for other long-context tasks, such as long-form document summarization, narrative generation, or complex question-answering over large texts, remain to be thoroughly investigated. The think-then-answer structure may not be as naturally suited or as beneficial for these domains, and adapting the framework might require designing new task-specific control mechanisms.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Rui-Jie Zhu, Jibin Wu, Yiran Zhong, Yu Qiao, Bo Xu, and Guoqi Li. 2024. Metala: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37:71034–71067.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. 2025. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*.

Qihang Fan, Huaibo Huang, and Ran He. 2025. Breaking the low-rank dilemma of linear attention. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 25271–25280.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. 2025. Sliding window attention training for efficient large language models. *arXiv preprint arXiv:2502.18845*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. 2024. Demystify mamba in vision: A linear attention perspective. *arXiv preprint arXiv:2405.16605*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Namgyu Ho, Sangmin Bae, Taehyeon Kim, Hyunjik Jo, Yireun Kim, Tal Schuster, Adam Fisch, James Thorne, and Se-Young Yun. 2024. Block transformer: Global-to-local language modeling for fast inference. *Advances in Neural Information Processing Systems*, 37:48740–48783.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. 2025. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*.

Ao Liu, Botong Zhou, Can Xu, Chayse Zhou, ChenChen Zhang, Chengcheng Xu, Chenhao Wang, Decheng Wu, Dengpeng Wu, Dian Jiao, and 1 others. 2025. Hunyuan-turbos: Advancing large language models through mamba-transformer synergy and adaptive chain-of-thought. *arXiv preprint arXiv:2505.15431*.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, and 1 others. 2025. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, and 1 others. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. 2024. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

Weigao Sun, Disen Lan, Yiran Zhong, Xiaoye Qu, and Yu Cheng. 2025. Lasp-2: Rethinking sequence parallelism for linear attention and its hybrid. *arXiv preprint arXiv:2502.07563*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Ling Team. 2025. Ring-lite: Scalable reasoning via c3po-stabilized reinforcement learning for llms. *Preprint*, arXiv:2506.14731.

Junxiong Wang, Wen-Ding Li, Daniele Paliotta, Daniel Ritter, Alexander M Rush, and Tri Dao. 2025. M1: Towards scalable test-time compute with mamba reasoning models. *arXiv preprint arXiv:2504.10449*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. 2025. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR.

Libo Zhang, Zhaoning Zhang, Rui Li, Zhiliang Tian, Songzhu Mei, Dongsheng Li, and 1 others. 2025a. Dovetail: A cpu/gpu heterogeneous speculative decoding for llm inference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17393–17406.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Yu Liu, Kai Chen, and Ping Luo. 2025b. Gpt4roi: Instruction tuning large language model on region-of-interest. In *European Conference on Computer Vision*, pages 52–70. Springer.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

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

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. 2024. Dynamickv: Task-aware adaptive kv cache compression for long context llms. *arXiv preprint arXiv:2412.14838*.

A PC-CPU Inference Performance

Table 4 details the inference speed (tokens/s) across 1.7B, 4B, and 8B parameter scales on the Intel Core i5-12600KF CPU. All inference speed tests were conducted using Q4_0 quantization precision. The results demonstrate a consistent trend: as the sequence length extends to 32,768 tokens, the standard Qwen3 models suffer severe performance degradation due to memory bandwidth saturation, with the 4B and 8B models dropping to single-digit speeds (6.84 and 5.13 tokens/s, respectively). In contrast, SADAS maintains robust generation rates across all scales. Notably, for the 4B model at 32k length, SADAS achieves 18.15 tokens/s, representing a nearly 2.6× speedup over the baseline, ensuring that even larger models remain responsive on consumer-grade CPUs.

B Training Details and Hyperparameters

We utilized the inclusionAI/Ring-lite-distill-preview-sft-data. To demonstrate the data efficiency of our approach and ensure that performance gains are attributable to the architecture rather than extensive training, we restricted the training data to only the first 5,000 samples. Fine-tuning the SADAS-1.7B model required only 3.5 hours on a single NVIDIA A800 GPU. This minimal training overhead confirms that the model's reasoning capabilities are preserved from the base model, while the efficiency gains are derived from the SADAS architecture.

Training was conducted for one epoch. To accommodate the lengthy Chain-of-Thought sequences typical in reasoning tasks, we set the maximum sequence length to 8192 tokens. We employed a conservative initial learning rate of 7×10^{-6} with a cosine decay schedule to minimize the disruption to the model's pre-trained knowledge representations.

C Analysis of the Necessity for Fine-tuning

In our research, to activate and stabilize SADAS's dynamic switching capability, we performed continuous full-parameter fine-tuning on the base model. We did not adopt parameter-efficient fine-tuning (PEFT) methods such as LoRA. Our primary considerations were to provide an unconstrained upper bound for evaluating the potential of the SADAS architecture and to ensure all model parameters adapted to the new framework, thereby avoiding potential performance bottlenecks that PEFT methods might introduce. To validate the necessity of fine-tuning, we designed a set of comparative experiments, directly testing performance under an initial setup where only model weights were imported and no fine-tuning was performed. The model parameters were set to 1.7B, and training settings referred to Appendix B.

From the data in Table 5, it is clearly evident that fine-tuning is a prerequisite for the successful operation of the SADAS framework. The non-fine-tuned SADAS model exhibited significant performance degradation across all key metrics, which can be attributed to the following reasons:

- **Cognitive Mismatch:** The internal weights of the non-fine-tuned model have not learned the "think-answer" rhythm. Forcing attention

Table 4: End-to-end inference throughput (tokens/s) of turnthink and Qwen3 under different output lengths.

Model	1024	2048	4096	8192	16384	32768
QWEN3-1.7B	44.62	38.90	37.44	29.98	21.99	13.65
turnthink	49.46	47.72	46.92	43.64	39.22	33.95
swaQwen	50.31	47.66	47.07	48.87	46.62	42.45
QWEN3-4B	22.16	20.70	18.49	14.97	10.73	6.84
turnthink	24.02	23.11	21.64	21.14	20.94	18.15
swaQwen	24.12	23.59	23.38	23.10	22.35	21.92
QWEN3-8B	13.16	12.72	11.80	10.30	8.26	5.13
turnthink	13.25	12.93	12.67	12.33	11.68	10.56
swaQwen	13.31	13.12	12.98	12.78	12.11	11.53

Table 5: Performance Comparison of SADAS under Fine-tuned and Non-Fine-tuned Settings

Scheme	AIME-24	AIME-25	COT Stability
Fine-tuned	56.7	36.6	High
Non-Fine-tuned	12.3	8.7	Low

mode switching disrupts its inherent generation logic, leading to incoherent model outputs and even complete disorientation in complex reasoning tasks, as evidenced by the substantial drop in AIME-24 scores.

- **Generation Instability:** As the model does not understand the semantics of control tokens and has not adapted to the dynamic changes in the attention calculation range, its generation process becomes highly unstable. We observed numerous instances of repetition, logical interruptions, or premature generation termination, which explains its "low" rating in CoT generation stability.

- **Reduced Computational Efficiency:** Although SWA is theoretically faster, the actual inference speed of the non-fine-tuned model was lower than that of the end-to-end optimized fine-tuned model. This is due to uncoordinated internal states during mode switching, which leads to frequent interruptions in the GPU computation flow.

In summary, a targeted fine-tuning phase is crucial for SADAS. It not only teaches the model how to autonomously utilize control tokens but, more importantly, reshapes the model's internal computation flow, enabling it to transition smoothly and efficiently between the two attention modes, thereby truly converting SADAS's architectural advantages into practical performance gains.

D Exploration and Trade-offs of Alternative Cache Management Schemes

The core of the SADAS framework lies in its dynamic nature, and KV cache management is the technical cornerstone for achieving this dynamism. The globally unified DynamicCache scheme adopted in the main text is the optimal solution we derived after comprehensive consideration of speed, accuracy, and implementation complexity. To more comprehensively illustrate our design decision process, this section will detail two other alternative schemes we explored, analyzing their advantages and limitations with experimental data.

Similarly, all experiments in this section were conducted with 1.7B model parameters, and all throughput tests used a batch size of 16, performing inference on a single A100 GPU.

D.1 Scheme 1: Hybrid Cache

This scheme aims to minimize GPU memory footprint. Its core mechanism involves using a fixed-size SWA Cache during the thinking phase. Upon transitioning to the answering phase, the contents of the SWA Cache are migrated to a new, infinitely growing Dynamic Cache via a one-time cache copy operation.

As shown in Table 6, the Hybrid Cache scheme demonstrates a significant advantage in peak GPU memory usage. For example, at an output length of 32768, it reduced memory demand by approximately 94.2% compared to SADAS. However, this advantage comes at the cost of sacrificing critical

Table 6: Performance Evaluation of the Hybrid Cache Scheme

Scheme		Output Length						
		512	1024	2048	4096	8192	16384	32768
Hybrid Cache	Speed (token/s)	17.09	33.65	71.93	169.41	453.71	1421	4853
	Peak Memory (GB)	0.889	1.763	1.763	1.764	1.766	1.768	3.221
SADAS	Speed (token/s)	18.9	37.84	75.31	163.97	417.68	1271.93	4316.15
	Peak Memory (GB)	0.889	1.765	3.515	7.016	14.019	28.02	55.808

Table 7: Performance Evaluation of the Truncated Dynamic Cache Scheme

Scheme		Output Length						
		512	1024	2048	4096	8192	16384	32768
Truncated Cache	Speed (token/s)	17.08	33.34	71.54	168.37	451.59	1418	4833
	Peak Memory (GB)	0.889	1.763	1.763	1.764	1.766	1.768	3.221
SADAS	Speed (token/s)	18.9	37.84	75.31	163.97	417.68	1271.93	4316.15
	Peak Memory (GB)	0.889	1.765	3.515	7.016	14.019	28.02	55.808

Table 8: Performance Evaluation of Alternative Schemes Integrating Sink Attention

Scheme		Output Length						
		512	1024	2048	4096	8192	16384	32768
Truncated Cache	Speed (token/s)	16.76	33.22	71.20	168.97	452.74	1416.22	4814
	Peak Memory (GB)	0.889	1.763	1.763	1.764	1.766	1.768	3.221
Hybrid Cache	Speed (token/s)	17.02	34.08	72.13	167.74	535.11	1418.22	4821
	Peak Memory (GB)	0.889	1.763	1.763	1.764	1.766	1.768	3.221
Truncated Cache + Sink	Speed (token/s)	17.08	33.34	71.54	168.37	451.59	1418	4833
	Peak Memory (GB)	0.889	1.764	1.764	1.765	1.766	1.768	3.223
Hybrid Cache + Sink	Speed (token/s)	17.09	33.65	71.93	169.41	453.71	1421	4853
	Peak Memory (GB)	0.889	1.764	1.764	1.765	1.766	1.768	3.223

Table 9: Reasoning Accuracy Evaluation of the Hybrid Cache Scheme

Scheme	AIME-24	AIME-25
Hybrid Cache	36.6	26.7
SADAS	56.7	36.6

performance. Its speed significantly decreased, primarily due to the substantial latency introduced by the cache copying operation, which represents a difficult-to-optimize serial bottleneck on the GPU. More critically, as shown in Table 9, its accuracy also suffered significantly, because the context for the final answering phase was limited to only the latter part of the thinking process, leading the model to lose a large amount of crucial early information and fail to complete complex reasoning tasks requiring long-range recall.

D.2 Scheme 2: Truncated Dynamic Cache

To address the speed issues of the Hybrid Cache scheme, we designed the Truncated Dynamic Cache. This scheme uses a single global cache, but during the thinking phase, it logically truncates access to older cache entries outside the window through attention masks.

As can be seen from the data in Table 7, this scheme avoids data copying, and its speed performance is close to that of our final adopted

Table 10: Reasoning Accuracy Evaluation of the Truncated Dynamic Cache Scheme

Scheme	AIME-24	AIME-25
Truncated Cache	36.6	26.7
SADAS	56.7	36.6

scheme. Its memory footprint is the same as the final scheme, as it still retains all KV pairs at the underlying level. However, as shown in Table 10, during the thinking phase, the model is similarly unable to recall early thinking steps outside the window, resulting in its accuracy still being lower than the final scheme. While this involves less information loss than the Hybrid Cache scheme, this limitation remains critical in complex reasoning chains that require repeated backtracking and verification.

Through a rigorous evaluation of the three schemes discussed, we concluded that while the alternative schemes offer attractive benefits in terms of memory savings, they all compromise the model's peak reasoning ability by introducing some form of permanent information loss during the thinking phase. The globally unified DynamicCache scheme we ultimately adopted has its core advantage in ensuring information completeness and flexible access. It allows the model to recall the entire history at any time by modifying attention masks, a capability that proved crucial for achieving efficient inference without sacrificing accuracy. Although it demands higher GPU memory, it provides the best overall performance in terms of both speed and accuracy, which is fully consistent with the original design philosophy of SADAS.

E GPU Inference Performance Analysis

E.1 Performance Observation

To quantify computational efficiency on server-grade hardware, we conducted end-to-end latency tests on a single NVIDIA A800 GPU. With a fixed prompt length of 2048 tokens and a batch size of 16, models generated output sequences of varying lengths (from 512 to 32,768 tokens). As detailed in Table 11, while SADAS still outperforms baselines at extreme lengths, the acceleration is less pronounced compared to edge CPUs.

Specifically, at 32,768 tokens, SADAS-1.7B reduces total generation time from 4937.93s (Qwen3) to 4316.15s, a 12.6% speedup. However, at intermediate lengths (e.g., 4096 tokens), the pure SWA variant actually exhibits higher latency (192.53s) compared to the full-attention baseline (183.34s). This counter-intuitive phenomenon—where a theoretically lighter mechanism runs slower—provides critical insight into the hardware-specific bottlenecks of linear attention.

E.2 Mechanism Analysis

Why does a pure SWA mechanism sometimes result in slower inference speeds? Our investigation revealed that its core bottleneck lies in the SlidingWindowCache's management. To maintain a fixed window size, it necessitates frequent eviction of the oldest KV pairs at each generation step. This non-trivial, often unparallelizable overhead—especially against highly optimized attention computations on GPUs—can negate the theoretical gains from reduced attention scope, leading to a theoretically fast, practically slow outcome.

In contrast, SADAS bypasses this issue by utilizing a globally unified DynamicCache. This cache is append-only, meaning KV pairs are simply added without costly eviction or rolling operations. This design simplifies data flow, eliminates cache management overhead, and ensures high computational continuity and parallelism. Consequently, the inherent speed benefits of linear attention methods like SWA are fully realized in SADAS, leading to robust acceleration (e.g., outperforming pure SWA by avoiding cache rolling) even on powerful accelerators, despite potentially higher memory consumption for very long sequences compared to fixed-window caches.

F Future Work: Exploration of Integrating Sink Attention

Building upon the success of the SADAS framework, our future work will focus on exploring token-level dynamic inference models with improved performance, higher efficiency, and lower memory footprint. A promising direction is to combine the cache optimization schemes we explored in Appendix B with cutting-edge long-sequence inference techniques, particularly the Attention Sink concept proposed in StreamingLLM.

The Attention Sink mechanism posits that in autoregressive models, the initial few tokens are crucial for maintaining attention stability and integrating global information, even when they fall outside the attention window. Retaining these ini-

Table 11: End-to-end inference latency of SADAS and Qwen3 variants under different output lengths (seconds)

Model	512	1024	2048	4096	8192	16384	32768
Qwen3-1.7B	17.16	34.60	72.24	183.34	595.11	1427.36	4937.93
SADAS-1.7B	18.90	37.84	75.31	163.97	417.68	1271.93	4316.15
SWAQwen-1.7B	16.87	33.65	71.73	192.53	508.92	1419.33	4617.54
Qwen3-4B	22.14	43.70	94.77	224.73	604.05	1862.90	–
SADAS-4B	25.86	47.99	97.64	214.26	547.98	1644.21	–
SWAQwen-4B	27.52	55.44	107.70	216.82	567.08	1800.73	–
Qwen3-8B	23.76	47.47	100.73	237.90	628.27	2097.96	–
SADAS-8B	26.28	52.30	106.53	248.72	616.95	1813.29	–
SWAQwen-8B	30.30	60.10	125.63	281.58	699.16	1963.14	–

tial tokens can effectively mitigate performance degradation caused by window sliding in long sequences. Inspired by this, we improved the two alternative schemes discussed in Appendix B:

1. **Hybrid Cache + Sink:** Building upon the original Hybrid Cache scheme, we permanently retained the initial few sink tokens within the SWA Cache. During cache copying, these sink tokens, along with the tokens within the sliding window, were copied to the new dynamic cache.

2. **Truncated Dynamic Cache + Sink:** In the Truncated Dynamic Cache scheme, we modified the attention mask to allow it to permanently attend to the initial sink tokens during the thinking phase, in addition to the tokens within the sliding window.

We conducted preliminary experiments on these two improved schemes, also using 1.7B model parameters, and all models underwent the same fine-tuning procedure.

Table 12: Reasoning Accuracy Evaluation of Alternative Schemes Integrating Sink Attention

Scheme	AIME-24	AIME-25
SADAS	56.7	36.6
Hybrid Cache	36.6	26.7
Hybrid Cache+Sink	46.6	32.9
Truncated Cache	36.6	26.7
Truncated Cache+Sink	46.6	32.9

From the results in Table 12, we can observe that by introducing the Attention Sink mechanism, the accuracy of both alternative schemes significantly improved, showing an increase of approximately 27.3% (relative to their non-Sink counterparts) on AIME-24. This demonstrates the importance of retaining initial global information for maintaining long-range reasoning capabilities, even under cache-constrained conditions. Despite the improved accuracy, these enhanced schemes did not show an advantage in inference speed. As shown in Table 8, versions integrating Sink attention even exhibited slightly slower inference speeds. The speed bottleneck for the Hybrid Cache scheme remains the costly cache copying operation, while the Truncated Dynamic Cache scheme experienced a slight increase in computational complexity after introducing additional attention to sink tokens, leading to a marginal decrease in speed.

This preliminary exploration points us towards a clear direction for future work. We have demonstrated that Attention Sink can effectively compensate for the accuracy shortcomings of cache-optimized schemes. Therefore, our core future research will focus on fundamentally addressing the speed bottleneck while preserving Sink information and optimizing memory usage. Possible exploration paths include designing more efficient, copy-free cache update mechanisms, or leveraging hardware-aware algorithms to optimize access to non-contiguous caches (sliding window + Sink). The ultimate goal is to build a next-generation dynamic inference architecture that achieves state-of-the-art performance in accuracy, speed, and memory efficiency.