LISA: LINEAR-INDEXED SPARSE ATTENTION FOR EFFICIENT LONG CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long chain-of-thought (CoT) reasoning models such as DeepSeek-R1 keep lengthening their inference contexts under test-time scaling, yet the $O(n^2)$ complexity of standard self-attention makes long-CoT reasoning costly to deploy in production. We therefore propose **LISA** (**L**inear-**I**ndexed **S**parse **A**ttention), a plug-and-play attention replacement module that requires no pretraining from scratch. LISA integrates two lightweight components in parallel inside the original model — a Linear Attention branch giving long-range memory at $O(n)$, and a Lightning Indexer that selects the top-M tokens from the full context for a Sparse Self-Attention branch — and fuses them by gating, replacing dense attention in each LISA layer with attention over a fixed budget of M selected tokens ($M \ll n$). Because full-attention layers are retained periodically and the Indexer still scores the full context, the overall sequence-level complexity class remains quadratic; the practical gain is a measured reduction in wall-clock latency rather than a change of asymptotic class. In a two-stage pipeline, Stage 1 trains the linear-attention branch with a next-token cross-entropy objective while retaining sliding-window attention and freezing the original model parameters. Stage 2 replaces that static window with the Indexer, enabling dynamic token selection over broader contexts; it is trained by a novel per-head KL divergence loss that aligns its selection behavior with the teacher's attention patterns. On DeepSeek-distilled-Qwen models, LISA cuts measured eager-mode mean sample latency by 23.9% ($1.31\times$) on the mathematical benchmarks for the 1.5B model, while improving equal-weight average accuracy by 4.2 percentage points over the full-attention baseline across nine benchmarks. A secondary decoding-FLOPs estimate (Appendix A), counting Indexer full-context scoring, is consistent with a moderate constant-factor saving at long context (about 26% for 1.5B at 16K).

1 INTRODUCTION

Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) has become a core technique for enhancing the reasoning ability of large language models (LLMs) on complex tasks. By prompting step-by-step reasoning, CoT enables LLMs to decompose complex problems into simpler subtasks, thus improving their problem-solving capabilities (Yao et al., 2023; Wang et al., 2023; Zhou et al., 2023). Recent studies, including OpenAI o1 (OpenAI et al., 2026), DeepSeek-R1 (DeepSeek et al., 2025a), and other reasoning models (Qwen Team, 2024; Zhao et al., 2024), demonstrate that scaling up CoT length can further enhance the reasoning abilities of LLMs.

However, since most current LLMs are built upon the Transformer architecture (Vaswani et al., 2017), the computational complexity of their attention grows quadratically with context length, and the memory overhead of their KV-cache increases linearly. Hence, generating long CoT substantially increases the computational and memory cost of LLMs, limiting their practical efficiency on complex reasoning tasks.

To improve reasoning efficiency, prior work shortens the chain itself via prompting (Han et al., 2024; Ma et al., 2025a), SFT (Liu et al., 2024; Munkhbat et al., 2025) or RL (Aggarwal & Welleck, 2025; Shen et al., 2025), but this often costs accuracy, since shortening conflicts with test-time

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.

scaling (Jin et al., 2024; Merrill & Sabharwal, 2024; OpenAI et al., 2026). Others re-express the same reasoning more concisely (Ma et al., 2025b; Kang et al., 2025; Xia et al., 2025), risking the loss of decisive steps (Wang et al., 2025). Yan et al. (2025) instead interleave short reasoning segments with intermediate summaries to keep context bounded, but require reconstructing training data into this iterative format. None of these changes the quadratic attention that makes long chains expensive in the first place.

In this paper, we propose an efficient hybrid inference framework **LISA** (**L**inear-**I**ndexed **S**parse **A**ttention). We replace the standard self-attention of a pretrained model with a hybrid attention architecture: a global linear attention vector maintains long-range context state, while a fixed-size self-attention module preserves precise contextual retrieval capability. This design restricts sparse self-attention in each LISA layer to a fixed budget of M selected tokens while retaining precise access to critical context. Our framework requires no complex data preprocessing, does not alter the original model parameters or output format, and achieves plug-and-play improvement in both inference efficiency and quality.

Our key contributions are as follows:

1. We propose a **plug-and-play hybrid attention module (LISA)** that combines global linear attention with fixed-size self-attention, restricting per-step sparse self-attention to a fixed budget of M tokens while preserving precise contextual retrieval.
2. We design a **two-stage training strategy**: the first stage cold-starts linear attention via pure cross-entropy loss to avoid overfitting to the teacher distribution; the second stage introduces an indexer module with top-M dynamic selection, augmented with KL loss to ensure stable important-token selection.
3. We conduct **extensive experiments** on five mathematical reasoning benchmarks (GSM8K, MATH-500, AMC23, AIME24, AIME25) and four non-mathematical benchmarks (GPQA-D (Rein et al., 2024), HumanEval (Chen et al., 2021), LiveCodeBench (Jain et al., 2024) and BigCodeBench (Zhuo et al., 2025)), demonstrating that LISA cuts measured eager-mode mean sample latency by 23.9% ($1.31\times$) on the mathematical benchmarks for the 1.5B model and improves its equal-weight average accuracy by 4.2 percentage points over the full self-attention baseline across these nine benchmarks. A secondary decoding-FLOPs estimate that includes Indexer scoring (Appendix A) indicates a moderate constant-factor saving at long context.

2 Related Work

Efficient Reasoning Methods. A growing body of work seeks to improve the efficiency of CoT reasoning. One line of research uses prompt engineering, SFT, or RL to encourage models to generate shorter CoT sequences (Nayab et al., 2024; Aggarwal & Welleck, 2025; Munkhbat et al., 2025). However, these methods often compromise reasoning capability, as shortening CoT conflicts with test-time scaling principles. Alternative approaches aim to express CoT more compactly without losing information. LightThinker (Zhang et al., 2025) proposes a "thinking step-by-step compression" paradigm that inserts learnable special tokens into hidden state sequences and compresses the KV states of completed reasoning steps before each generation step. INFTYTHINK (Yan et al., 2025) compresses the processed lengthy context and prior reasoning trajectories into compact summary representations at each step, freeing up context window capacity for subsequent reasoning. Additionally, TokenSkip (Xia et al., 2025) achieves controllable CoT compression through a learnable token-skipping mechanism, and R1-Compress (Wang et al., 2025) explores long CoT compression via chunk-level compression and search-based optimization. Zhang et al. (2026) maintain a global state matrix via linear attention to record historical information, enabling the model to retrieve relevant information without explicitly attending to prior reasoning steps. Our work differs from these methods in that we do not compress the output CoT itself but rather accelerate the internal inference process while preserving the full reasoning capability.

KV-Cache Reduction Methods. Reducing KV-cache size is another major direction for efficient LLM inference. H2O (Zhang et al., 2023) proposes a layer-wise eviction strategy that retains only KV vectors of Heavy-Hitter tokens and recently generated tokens, maintaining a constant-size KV-cache while preserving generation quality. StreamingLLM (Xiao et al., 2023) discovers the "atten-

tion sink" phenomenon and proposes retaining only initial and recent tokens' KV vectors to enable efficient streaming generation. SnapKV (Li et al., 2024) observes that the model can identify important KV pairs before generation and proposes a query-driven KV cache compression method. PyramidKV (Cai et al., 2025) further reveals a pyramidal information funneling structure across layers and proposes layer-wise adaptive KV cache budgets. SepLLM (Chen et al., 2025a) compresses each segment into a single separator token and stores only the KV vectors of initial, neighboring, and separator tokens. These methods focus on reducing the stored KV-cache footprint during decoding; our work instead keeps the full KV cache and targets reducing decoding compute and latency for long-reasoning tasks.

Linear Attention. To address the quadratic computational cost of softmax attention, many linear attention methods have been proposed to improve training and inference efficiency. Early studies primarily investigated kernel functions for linear attention, such as random feature attention (Peng et al., 2021) and RNN-izing Transformers (Kasai et al., 2021). Subsequently, several works introduced gating mechanisms to more precisely control information flow in the model state: GLA (Yang et al., 2024a) achieves hardware-efficient training with gated linear attention, HGRN2 (Qin et al., 2024) introduces state-expanded gated linear RNNs, and Liger (Lan et al., 2025) linearizes large language models into gated recurrent structures. Recent studies have provided theoretical evidence for the Test-Time Training (TTT) property of linear attention (Yang et al., 2024b; Sun et al., 2025; Liu et al., 2025; Chen et al., 2025b), offering theoretical foundations for designing online learning objectives to enhance sequence modeling capability.

3 METHOD

3.1 PRELIMINARIES

Softmax Attention. Decoder-only LLMs rely on multi-head softmax attention (Vaswani et al., 2017), whose output is a softmax-weighted sum over values. With a KV cache, attending for one newly generated token costs $O(t)$ in the current context length t (ignoring model width); generating a full sequence of length n therefore accumulates to $O(n^2)$. The KV cache itself grows linearly with sequence length, a significant memory bottleneck in long-context settings.

Linear Attention. Linear attention (Katharopoulos et al., 2020) replaces the exponential kernel with a decomposable feature map $\phi(\cdot)$ and uses the associativity of matrix multiplication to aggregate KV interactions into a single state matrix. While many kernels have been explored (Kasai et al., 2021; Peng et al., 2021), recent studies (Sun et al., 2023; Yang et al., 2024a) set $\phi(\cdot)$ to the identity and drop the normalization term: $\mathbf{o}_t = \mathbf{q}_t\mathbf{S}_t$, with state $\mathbf{S}_t = \sum_i \mathbf{k}_i^\top \mathbf{v}_i \in \mathbb{R}^{d_k \times d_v}$ accumulating history. This recurrent form gives $O(1)$ per-step computation and constant memory.

Test-Time Training (TTT). A line of work (Yang et al., 2024b; Sun et al., 2025; Liu et al., 2025; Chen et al., 2025b) reads $\mathbf{S}_t$ as a fast-adapting parameter updated at every token by lightweight gradient descent. Under the same left-multiply convention as above, a natural objective is $\mathcal{L}(\mathbf{S}) = -\langle \mathbf{k}_t\mathbf{S}, \mathbf{v}_t \rangle$, with update $\mathbf{S}_t = \mathbf{S}_{t-1} + \beta \mathbf{k}_t^\top \mathbf{v}_t$. At $\beta = 1$ this coincides with the linear-attention state update, i.e. training $\mathbf{S}_t$ on that objective. No direct empirical evidence yet shows that linear attention improves reasoning, but its theoretical properties suggest considerable potential.

3.2 LISA: LINEAR-INDEXED SPARSE ATTENTION

We design LISA to replace the original attention module. Specifically, our method combines a linear attention stream with a fixed-size self-attention stream, where the former records long-range contextual information and the latter preserves precise detail retrieval. A gating mechanism controls the information flow ratio between the two streams. Following related works on linear attention (MiniMax et al., 2025), we replace the standard attention in the teacher model with our linear attention variant while retaining full attention at Transformer blocks $0, 7, 14, 21, 27$ (0-indexed)—five of the 28 blocks, with six LISA blocks between consecutive full-attention blocks except before the final full block—to preserve exact retrieval capability (Arora et al., 2024). The sparse self-attention after selection costs $O(M)$ per decoded token, but Indexer scoring still scans the full context ($O(t)$ per step with K heads), and the retained dense layers remain $O(t)$; thus LISA reduces attention compute

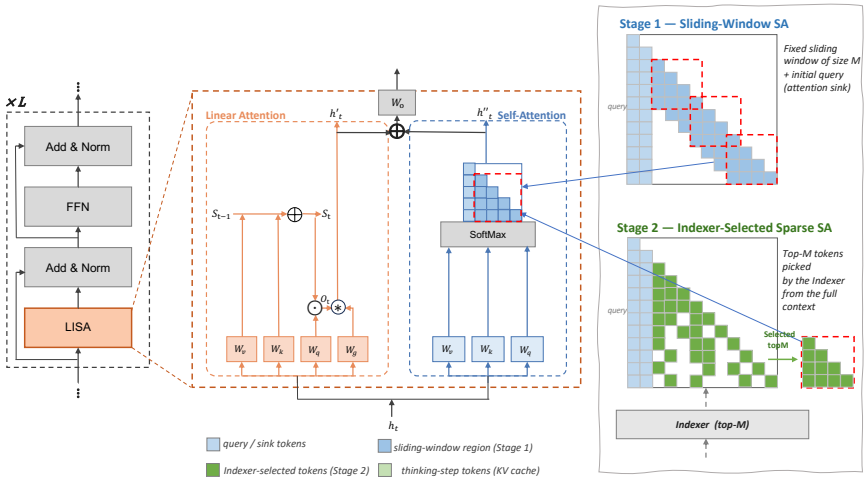

Figure 1: Overview of the LISA architecture.

via a smaller selected-token budget while retaining quadratic sequence-level complexity, consistent with the Lightning Indexer analysis in DeepSeek-V3.2 (DeepSeek et al., 2025b).

Specifically, we divide the training into two stages.

Stage 1: Linear Attention Cold-Start. In this stage, we convert the standard self-attention to a fixed-budget sliding window of size M, while introducing a linear attention as in Figure 1.

During training, we freeze all original model parameters and only train the newly added linear attention parameters. We construct an attention block that fuses linear attention and self-attention through a learnable gating parameter:

$$\mathbf{o}_{\mathrm{LISA}} = g \cdot \mathbf{o}_{\mathrm{LA}} + (1 - g) \cdot \mathbf{o}_{\mathrm{SA_M}}, \tag{1}$$

where g is the learnable gate, $\mathbf{o}_{\mathrm{LA}}$ is the linear attention output, and $\mathbf{o}_{\mathrm{SA_M}}$ is the sliding-window self-attention output with window size M.

We aim for the LISA output to remain close to the teacher's generation behavior. Rather than using an explicit distillation loss (e.g., KL over attention maps) in Stage 1, we train with standard next-token cross-entropy. Empirically, applying KL over attention distributions in early training led to severe overfitting and degraded downstream generation; we therefore defer attention-distribution matching to the Indexer objective in Stage 2.

Stage 2: Indexer-Guided Token Selection. In this stage, we further introduce a lightweight token selector called the *Indexer*. Its design is inspired by the Lightning Indexer in DeepSeek V3.2 (DeepSeek et al., 2025b) and GLM-5 (Zhipu AI, 2026): it contains only query/key projections and a ReLU dot product, used to select important tokens based on the current context.

Specifically, we maintain the KV cache for the full context. At each generation step t, the Indexer performs an attention computation over the preceding $t-1$ tokens, selecting M important tokens to fill into the self-attention, which together with the linear attention state at step t produces the t-th token. The resulting state is used to update both the linear attention state and the KV cache.

The Indexer's role is to select important tokens from the full context for subsequent token generation, which is similar to DeepSeek V3.2 (DeepSeek et al., 2025b) and GLM-5 (Zhipu AI, 2026). The key difference is that, due to the top-M (argmax) strategy, training gradients cannot flow back to the Indexer through the discrete selection. Their approach maintains a distillation loss to update the Indexer; V3.2 in particular continues from a prior checkpoint with a backbone warm-up rather than training from a randomly initialized LLM, which differs from our post-hoc conversion of a frozen reasoning checkpoint. Building on this, we introduce a candidate-set dropout as regularization: during early training we take the top-$\lceil 1.25M \rceil$ scored positions and uniformly subsample exactly M of them to fill the self-attention stream, forcing the model to make robust selections among near-top candidates. As training progresses, we gradually shrink the candidate pool toward M. This

dropout regularizes the selected set and the downstream sparse branch; it does *not* restore gradients through the hard top-M gate—the Indexer parameters are updated by the KL objective below.

However, this still limits the model's performance to some extent. Therefore, we introduce a **multi-head mechanism** for the Indexer, with the loss applied directly on each head. We set the Indexer width to d_I=256 and assign one Indexer head to every four teacher attention heads, i.e. K=$P/4$ where P is the number of teacher heads (yielding K=3 on the 1.5B model with P=12 and K=7 on the 7B model with P=28). Each Indexer head is trained to fit the attention distribution obtained by summing and re-normalizing its four assigned teacher heads.

Concretely, we sum the scores of K Indexer heads over all tokens and select the M positions with the largest values as **important tokens**. Based on the Indexer's selection, we fill the cached hidden states at the corresponding positions into SA_M for subsequent computation. This approach mitigates the loss of precise contextual information caused by linear attention state compression.

The linear attention continues to be updated via cross-entropy loss. The Indexer loss is defined as:

$$\mathcal{L}_{\text{Indexer}} = \frac{1}{KL} \sum_{j=1}^{K} \sum_{t=1}^{L} \mathrm{KL}\Big(T_{t,:}^{(j)} \,\Big\|\, \mathrm{softmax}(S_{t,:}^{(j)})\Big), \tag{2}$$

where $S_{t,:}^{(j)} \in \mathbb{R}^L$ is the ReLU dot-product score vector of the j-th Indexer head over all keys, computed as $\mathrm{ReLU}\big(\langle q_t^{(j)}, k_{\cdot}^{(j)} \rangle\big)$, with $q_t^{(j)} = \mathrm{RoPE}(W_j^Q h_t)$ and $k_{\cdot}^{(j)} = \mathrm{RoPE}(W_j^K h_{\cdot})$. The target distribution $T_{t,:}^{(j)}$ is the normalized teacher target distribution for the j-th Indexer head, obtained by summing the attention distributions of its four assigned teacher heads and re-normalizing (with stop-gradient applied). Both $\mathrm{softmax}(\cdot)$ and the normalization in $T_{t,:}^{(j)}$ are restricted to the causal triangle for row-wise normalization.

3.3 Inference State Transfer

During inference, leveraging the test-time training (TTT) properties inherent in linear attention, we apply state-transfer operations to stabilize the model's reasoning trajectory. For segment-level correction we follow Zhang et al. (2026) (Liang Zhang et al.), partitioning generation into fixed-length segments and applying their dual-update rule to synchronize linear-attention (LA) state transitions with segment-level corrections. Our contributions relative to that line of work are the dynamic Indexer, per-head Indexer supervision, and their combination with this adopted correction rule.

LA State Update. In the l-th layer LA submodule, after completing the first $t-1$ reasoning segments, the model state is updated to $\mathbf{S}_{t-1}^{(l)}$. This serves as the initial state $\mathbf{S}_{t,0}^{(l)} = \mathbf{S}_{t-1}^{(l)}$ for the current reasoning segment and is updated token-by-token:

$$\mathbf{S}_{t,i-1}^{(l)} = \mathbf{S}_{t,0}^{(l)} + \sum_{j=1}^{i-1} {\mathbf{k}_j^{(l)}}^{\top} \mathbf{v}_j^{(l)}. \tag{3}$$

We then use the query vector $\mathbf{q}_{i-1}^{(l)}$ of input token $\mathbf{x}_{i-1}$ to extract relevant historical information from this state:

$$\mathbf{o}_{i-1}^{(l)} = \mathbf{q}_{i-1}^{(l)} \mathbf{S}_{t,i-1}^{(l)}. \tag{4}$$

We obtain the submodule output via a gating mechanism:

$$\check{\mathbf{h}}_{t,i-1}^{(l)} = \sigma\Big(\mathbf{w}_g^{\top} \mathbf{h}_{t,i-1}^{(l-1)}\Big) \odot \mathbf{o}_{i-1}^{(l)}, \tag{5}$$

where σ is the sigmoid and $\mathbf{w}_g \in \mathbb{R}^d$ is a lightweight per-layer gate (a scalar after the sigmoid, broadcast over channels). The LA submodule is implemented via a LoRA (Hu et al., 2022) strategy with rank r=256.

Inference-Time Segment Correction. Noisy reasoning segments can push the linear-attention state off a stable trajectory. We adopt the segment correction of Zhang et al. (2026) without claiming it as a contribution: partition generation into fixed-size segments, take each segment's state change as its reasoning direction, and pull it toward the momentum-averaged direction of preceding segments before applying the update. A brief restatement is given in Appendix C; the contribution of applying this rule in our setting is isolated in Table 4 (*No segment correction*, Avg 50.4 vs. 57.1).

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

Our experiments utilize the Qwen-2.5 model series (Qwen et al., 2025). To construct the training set, we randomly sample 100K samples from the OpenR1-Math-220K dataset. The model is initialized using the weights of its corresponding DeepSeek-R1 distilled version (DeepSeek et al., 2025a).

During Stage 1 of training, we employ a learning rate of 2×10^{-5} and a batch size of 128 for 2 epochs with $M = 256$. To initialize the linear attention layers, we evaluated both random initialization and Singular Value Decomposition (SVD) of the base model's self-attention weights. Since SVD initialization yielded no measurable performance gains, all reported experiments utilize random initialization accompanied by a 50-step linear warmup. This stage takes approximately 240 GPU hours on an NVIDIA H100. The Indexer uses per-head width d_I=256 with one Indexer head per four teacher heads (K=$P/4$; K=3 on 1.5B and K=7 on 7B). Linear-attention adapters are implemented with LoRA (Hu et al., 2022) of rank r=256. Full attention is retained at blocks $0, 7, 14, 21, 27$ (5 of 28); the remaining 23 blocks are LISA layers.

For Stage 2, the model is trained for 1 epoch using a learning rate of 2×10^{-5} and a batch size of 64 with $M = 256$. We initialize the Indexer randomly with a 50-step warmup. To mitigate the impact of initial noisy Indexer representations, the linear attention module is frozen for the first 200 steps. Subsequently, the Indexer and linear attention are updated simultaneously. With a halved batch size relative to Stage 1, the number of optimizer steps is comparable, but wall-clock cost rises substantially because each step requires teacher attention computation, taking approximately 240 GPU hours on an NVIDIA H100.

Parameter and Memory Overhead. LISA retains the full KV cache, so the stored KV footprint is not reduced by the proposed method. Beyond the lightweight auxiliary parameters (Linear Attention with LoRA rank r=256, the Indexer with d_I=256, and a per-layer scalar gate)—roughly 111M extra parameters (+7.4%) for the 1.5B model and 420M (+6.0%) for the 7B model—LISA also maintains recurrent LA states and an Indexer-key cache that grows linearly with context length. The bulk of the parameter overhead comes from the LA adapters and Indexer; the gate contributes only a small residual. The auxiliary *parameter* count itself is fixed and independent of context length.

4.2 BASELINES

We compare our framework with two types of baselines: (1) **Efficient reasoning methods**: **LightThinker** (Zhang et al., 2025) uses learnable special tokens to compress the reasoning information of completed reasoning steps; **INFTYTHINK** (Yan et al., 2025) compresses prior reasoning steps into a concise summary; **Zhang et al. (2026)** proposes a state-transition framework that uses linear attention to maintain a global state matrix for recording historical information, enabling the model to retrieve relevant information without explicitly attending to prior reasoning steps. (2) **KV-cache reduction methods**: **H2O** (Zhang et al., 2023) retains only the KV vectors of heavy-hitter tokens and recent tokens; **SepLLM** (Chen et al., 2025a) only stores the KV vectors of initial, neighboring, and separator tokens. All baselines are trained using the LoRA (Hu et al., 2022) strategy with a comparable number of trainable parameters to our model.

4.3 DATASETS & METRICS

We evaluate our framework on five mathematical benchmarks (GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AMC23 (AMC, 2023), AIME24 (AIME, 2024) & 25 (AIME, 2025)), and assess generalization on four non-mathematical benchmarks (GPQA-D (Rein et al., 2024), HumanEval (Chen et al., 2021), LiveCodeBench (Jain et al., 2024), and BigCodeBench (Zhuo et al., 2025)). We report three metrics: (1) *Accuracy* (**Acc**) denotes the percentage of correct answers; (2) *Token Number* (**Tok**) refers to the average length of the CoT sequence; (3) *Reasoning Latency* (**ReL**) is defined as the average wall-clock inference time per sample (seconds). We measure reasoning latency on a single NVIDIA A100 GPU, using eager mode with BF16 precision.

Note that latency measured in eager mode does not reflect the actual speedup attainable under production-grade inference frameworks such as vLLM or SGLang, whose advantages stem largely

Method	GSM8K			MATH-500			AMC23			AIME24			AIME25			AVG.	
	Acc↑	Tok↓	ReL↓	Acc↑	Tok↓	ReL↓	Acc↑	Tok↓	ReL↓	Acc↑	Tok↓	ReL↓	Acc↑	Tok↓	ReL↓	Acc↑	ReL↓
Qwen2.5-1.5B Series																	
Full-attention baseline	80.1	2086	76.2	78.8	3958	139.4	62.5	6392	242.9	20.0	13765	536.8	16.7	12684	504.7	51.6	300.0
+ H2O	76.1	2116	71.7	75.4	3835	128.5	55.0	6230	209.7	13.3	13921	462.2	10.0	12802	428.6	46.0	260.1
+ SepLLM	77.3	2230	75.8	76.0	3890	130.3	57.5	6573	220.2	13.3	13690	452.6	13.3	12690	423.9	47.5	260.6
LightThinker	78.8	2109	69.6	77.6	4060	134.8	60.5	6312	214.6	16.7	13827	461.6	13.3	12934	434.6	49.4	263.0
INFTYTHINK	79.5	2503	89.3	78.0	4750	170.2	62.5	7605	263.5	16.7	16318	566.2	16.7	15020	501.0	50.7	318.0
Zhang et al. (2026)	80.1	2116	69.7	81.2	3812	125.8	67.5	6460	213.2	26.7	13610	440.1	23.3	12910	416.0	55.8	253.0
LISA	**80.2**	**1983**	**61.5**	**82.0**	**3636**	**102.5**	**70.0**	**6157**	**187.7**	**26.7**	**12966**	**392.3**	**26.7**	**12532**	**398.1**	**57.1**	**228.4**
Qwen2.5-7B Series																	
Full-attention baseline	89.4	2649	96.4	87.4	4253	161.6	82.5	6704	265.2	40.0	12901	522.1	30.0	13204	548.1	65.9	318.7
+ H2O	83.5	2730	92.0	82.6	4389	147.1	77.5	6843	221.5	30.0	13274	448.9	23.3	13204	443.7	59.4	270.6
+ SepLLM	84.9	2582	92.3	84.2	4244	144.3	80.0	6511	231.2	33.3	13100	484.7	23.3	12972	435.9	61.1	277.7
LightThinker	85.6	2779	90.0	84.8	4321	145.2	80.0	6790	242.5	33.3	13307	449.0	26.7	13250	443.9	62.1	274.1
INFTYTHINK	87.9	3237	117.0	86.2	5050	184.8	82.5	8061	261.9	36.7	15094	508.7	30.0	15448	530.5	64.7	320.6
Zhang et al. (2026)	90.9	2603	87.2	90.0	4196	140.6	85.0	6665	245.1	43.3	13017	434.8	33.3	13191	440.6	68.5	269.7
LISA	**90.9**	**2462**	**82.1**	**91.2**	**4253**	**117.9**	**87.5**	**6841**	**196.3**	**43.3**	**12587**	**385.1**	**40.0**	**13642**	**398.1**	**70.6**	**235.9**

Table 1: Results of our model and baselines on mathematical reasoning benchmarks. The *Full-attention baseline* row is the unmodified original model. All cells report a single representative run; run-to-run variance on the small-sample benchmarks is reported separately in Table 2.

	AMC23		AIME24		AIME25	
	Acc↑	±std	Acc↑	±std	Acc↑	±std
Full-attention baseline	62.5	2.24	19.6	1.65	16.1	1.83
LISA	**70.0**	2.74	**25.8**	1.96	**25.9**	2.29

Table 2: Mean accuracy (%) over 16 independent evaluation runs on the small-sample benchmarks, Qwen2.5-1.5B. These benchmarks contain only about 30 problems each, so we report the standard deviation alongside the mean; the corresponding single-run values in Table 1 fall within one standard deviation of these means.

from kernel-level optimizations (e.g., PagedAttention, FlashInfer). Integrating LISA's linear attention, indexed sparse attention, and gating into such frameworks requires custom kernel development and is beyond the scope of this work. A secondary decoding-FLOPs estimate that includes Indexer scoring is given in Appendix A; our primary efficiency claim is the measured ReL.

In addition, since AMC23, AIME24, and AIME25 contain only about 30 problems each, we evaluate these three benchmarks on the Qwen2.5-1.5B model over 16 independent runs and report the mean ± standard deviation in Table 2. Table 1 itself reports single representative runs throughout.

4.4 Main Results

We conduct a comprehensive evaluation of LISA across five mathematical reasoning benchmarks. The results, summarized in Table 1, compare LISA against state-of-the-art efficiency-driven baselines using both Qwen2.5-1.5B and Qwen2.5-7B backbones.

LISA improves average accuracy and efficiency. LISA attains the highest average accuracy on the mathematical suite and matches or exceeds the compared methods on each benchmark, while reporting the lowest mean per-sample latency (**ReL**↓) among all tested methods. On the Qwen2.5-7B series, LISA reaches an average accuracy of 70.6%, outperforming the strongest baseline (Zhang et al., 2026) by 2.1 percentage points, while reducing mean ReL from 269.7 to 235.9. Because the full-attention baseline is the unmodified checkpoint without our conversion training, accuracy gains should be read as the combined effect of the hybrid architecture and the additional Stage 1/2 training rather than as a pure architectural delta. The 23.9% ReL reduction quoted in the abstract is this table's 1.5B average (300.0→228.4) and is not a fixed-16K per-token measurement; generation lengths also differ slightly across methods (# Tok), so latency comparisons inherit that confound.

Impact of sequence length and task complexity. A key observation from Table 1 is the correlation between task complexity (and output length) and the performance margin of our method. Specifically, on benchmarks with short outputs like GSM8K, the performance gap between LISA and the full-attention baseline is minimal (e.g., 80.2% vs 80.1% for the 1.5B model). The primary reason is that our method still introduces a fixed budget for self-attention, yielding marginal gains

Method	GPQA-D↑	HumanEval↑	LiveCodeBench↑	BigCodeBench↑
Full-attention baseline	23.23	76.22	35.0	22.7
LISA	**26.26**	**79.27**	**37.4**	**24.7**

Table 3: Generalization results of Qwen2.5-1.5B on non-mathematical tasks (accuracy, %).

Variant	Description	GSM8K	MATH-500	AMC23	AIME24	AIME25	Avg
		Acc↑	Acc↑	Acc↑	Acc↑	Acc↑	
Full LISA (ours)	Complete method (Stage 1 + Stage 2)	**80.2**	**82.0**	**70.0**	**26.7**	**26.7**	**57.1**
Stage 1 only	Sliding-window sparse without indexer-guided training	79.2	80.6	65.0	23.3	23.3	54.3
Linear only	Remove sparse branch	68.3	67.8	55.0	13.4	10.0	42.9
Sparse only	Remove linear branch	66.6	60.8	42.5	6.7	6.7	36.7
No gating	Remove learned gating	78.3	80.2	65.0	23.3	20.0	53.4
Single-head Indexer	Replace multi-head with single-head Indexer	79.5	81.2	67.5	26.7	23.3	55.6
No segment correction	Remove k-step correction at inference	74.9	77.8	62.5	20.0	16.7	50.4

Table 4: Component-level ablation on Qwen2.5-1.5B (accuracy, %). All variants use the same training configuration.

on short sequences. By contrast, on tasks requiring long-horizon reasoning, the advantages of LISA become more pronounced. For instance, on AIME25 with the 7B model, LISA achieves 40.0% accuracy, significantly higher than the 33.3% reached by Zhang et al. (2026). This suggests that our sparse attention mechanism is particularly adept at maintaining coherence and reasoning accuracy in long-sequence generation where standard KV-cache or attention-reduction methods often struggle.

4.5 Generalization to Non-Mathematical Tasks

To verify that LISA generalizes beyond mathematical reasoning, we also evaluate the Qwen2.5-1.5B model on science (GPQA-D (Rein et al., 2024)) and code domains (HumanEval (Chen et al., 2021), LiveCodeBench (Jain et al., 2024), and BigCodeBench (Zhuo et al., 2025)); the results are reported in Table 3. LISA preserves baseline performance on these non-mathematical tasks, indicating that the architectural conversion does not introduce domain-specific degradation.

4.6 Component-Level Ablation

We conduct a full component-level ablation on DeepSeek-R1-Distill-Qwen-1.5B, with all variants sharing the same training configuration; the results are reported in Table 4. Removing either branch (Linear only / Sparse only) causes a substantial drop in performance, indicating that the two branches each provide information the other cannot substitute. Moreover, replacing Indexer-guided selection with a sliding window (Stage 1 only) also degrades performance; this variant additionally omits Stage 2 training, so the gap reflects both the selection mechanism and the missing Stage 2 updates. Among the remaining components the ordering is consistent: removing segment correction costs -6.7 Avg, more than gating (-3.7) and the multi-head Indexer (-1.5) combined. Segment correction is therefore the largest of the three despite being the only one that adds no learnable parameter—it acts purely at inference time. These results support the effectiveness of segment correction; linking it causally to "overthinking" would require more direct measurements (e.g., error-continuation or state-drift statistics) and is left for future analysis.

Framework-agnostic efficiency measure. Since latency measured in eager mode does not track production inference frameworks, we additionally report a secondary per-token decoding-FLOPs estimate in Table 5 (Appendix A), including Indexer full-context scoring. Gains grow with context length and remain moderate; our primary efficiency claim is the measured ReL reduction above. After selection, sparse self-attention in a LISA layer costs $O(M)$ per decoded token, while Indexer scoring remains linear in the current context length and retained full-attention layers remain $O(t)$, so the overall sequence-level complexity stays hybrid / quadratic.

4.7 Analysis

Analyzing the Synergy between Linear and Sparse Attention. To verify that the linear branch actually supplies the global context it is intended to provide, we compare the hidden states of the

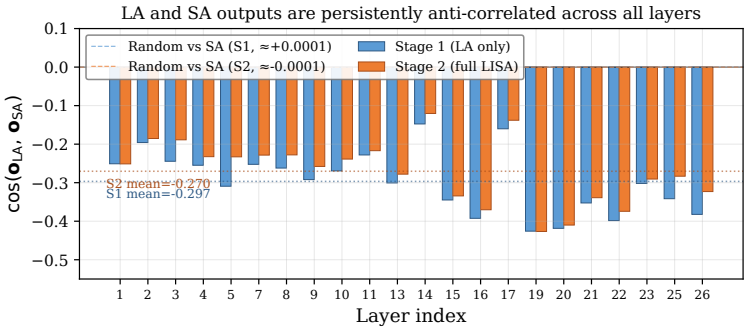

Figure 2: Cosine similarity between the linear and sparse attention outputs across all layers.

two branches by computing the cosine similarity between their outputs (Figure 2). The similarity is consistently negative across all layers, indicating that the two representations are non-trivially anti-aligned in direction rather than near-duplicates.

A negative similarity is by itself only a correlational observation, however. The causal evidence is the component-level ablation in Table 4: removing either branch degrades performance sharply, and the Stage 1-only sliding-window variant also underperforms full LISA. Consistently, the similarity shifts from -0.297 to -0.270 in Stage 2,[1] which indicates that as the Indexer widens the sparse branch's receptive field, the two signals converge partially. A companion analysis of what the Indexer itself selects across layers is provided in Appendix B.

Window Size M and Attention Coverage. We investigate the impact of M on both sliding-window size and Indexer capacity. To quantify coverage of the teacher's attention mass, we define:

$$\text{MASS}(q, M) = \frac{\sum_{k \in \text{top-}M(q)} \text{Attn}(q, k)}{\sum_{\text{all } k \le q} \text{Attn}(q, k)}, \tag{6}$$

where $\text{Attn}(q, k)$ denotes the *teacher* attention weights. When evaluating the Indexer, top-$M(q)$ are the positions selected by the Indexer; the oracle upper bound instead takes the teacher's own top-M positions. At $M = 256$, Indexer MASS reaches 80%, trailing the oracle upper bound by 6 percentage points (86%). The metric is 70% at $M = 128$ and 88% at $M = 512$. We adopt $M = 256$ as the default throughout our experiments as a practical coverage–cost trade-off; establishing optimality would additionally require paired task-accuracy and latency curves, which we do not claim here.

5 Conclusion

We proposed LISA (Linear-Indexed Sparse Attention), a plug-and-play attention mechanism that accelerates long-context chain-of-thought reasoning without pre-training from scratch. It runs two parallel streams — a global linear attention branch for long-range context and a fixed-size sparse self-attention branch for high-fidelity local and salient reasoning — trained in two stages: cold-starting linear attention for long-range dependencies, then introducing an indexer for dynamic token selection. This replaces dense attention in each LISA layer with attention over a fixed budget of M selected tokens ($M \ll n$); because full-attention layers are retained periodically and Indexer scoring still scans the full context, the overall sequence-level complexity class remains quadratic. On mathematical reasoning and on non-mathematical tasks including science question answering and code generation, LISA reduces measured inference latency (23.9% mean ReL on 1.5B) while matching or surpassing competitive baselines. Future work includes adaptive token budgets, integration into production inference frameworks for end-to-end speedups, and long-context retrieval and agent-based settings.

AI use statement

Generative AI was not used to implement LISA, design or run experiments, interpret results, formulate mathematical claims, propose hypotheses, or generate, clean, or reformat datasets. The scien-

[1] These magnitudes are small in absolute value; for scale, the cosine similarity between random vectors and sparse attention outputs is approximately 10^{-4}, three orders of magnitude lower.

tific contributions, experimental conclusions, and final claims were determined and independently verified by the authors.

Generative AI tools were exclusively used to polish the grammar and phrasing of this paper. The authors checked every AI-assisted passage against the original manuscript, experimental records, source code, and cited references. They also manually verified every equation, numerical claim, and citation in the paper. AI-suggested wording or structure was accepted only when it preserved the intended scientific meaning and introduced no unsupported claim. The authors take responsibility for the final content, including text, claims, and artifacts produced with AI assistance.

Ethics statement

This work studies inference efficiency and involves no human participants, personal data, or new data collection. Training and evaluation use the public datasets and benchmarks described in Section 4.3. These resources are used under their published terms, and the authors will not redistribute dataset content beyond what those terms permit. Any release will contain only code, model components, and configurations.

LISA may reduce the compute required to serve long-reasoning models, which could improve access and lower per-query resource use. The same efficiency gain could also reduce the cost of deploying capable models for harmful purposes. LISA adds no safety mechanism and does not remove biases or other risks inherited from the base models; deployments should retain the evaluation, access controls, and monitoring appropriate for those models.

To support research integrity, we report the retained quadratic components, additional training cost, evaluation scope, unmatched training budget, and run-to-run variance instead of presenting the method as an asymptotic or universally applicable improvement. Training the added components also incurs substantial GPU cost, which should be weighed against any inference savings. The method changes the attention mechanism rather than the model's safety properties. No additional privacy, human-subjects, or discrimination concerns arise beyond those associated with the public data and pretrained models used in this study

Reproducibility statement

Section 3 specifies the LISA architecture, objectives, token-selection rule, gating, and inference-time segment correction in closed form. Section 4 gives the base checkpoints used, the 100K-sample training set, the complete two-stage optimization schedule, and the evaluation and latency protocols. It reports learning rates, batch sizes, epochs, warmups, the freezing schedule, LoRA rank, Indexer dimensions, retained-attention layers, token budget, hardware, metrics, and precision. Tables 1-4 report the main results, variance, and ablations, while the appendix provides the FLOP analysis and complete token-selection traces.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv:2503.04697*, 2025.

AIME. American invitational mathematics examination (aime) 2024-i & ii, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

AIME. American invitational mathematics examination (aime) 2025-i & ii, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

AMC. American mathematics competitions, 2023. URL https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2024. URL https://arxiv.org/abs/2402.18668.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *arXiv:2306.12929*, 2023.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2025. URL https://arxiv.org/abs/2406.02069.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. In *Proceedings of ICML*, 2025a.

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

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+3=? on the overthinking of o1-like llms. In *Proceedings of ICML*, 2025b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

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

DeepSeek et al. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv:2512.02556*, 2025b.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv:2412.18547*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of NeurIPS*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *Proceedings of ICLR*, 2022.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In *Proceedings of ACL*, 2024.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of AAAI*, 2025.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. Finetuning pretrained transformers into RNNs. In *Proceedings of EMNLP*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of ICML*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of NeurIPS*, 2022.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *Proceedings of ICML*, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation, 2024. URL https://arxiv.org/abs/2404.14469.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *Proceedings of ICLR*, 2025.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? In *Proceedings of NeurIPS*, 2024.

Chao Lou, Zixia Jia, Zilong Zheng, et al. Sparser is faster and less is more: Efficient sparse attention for long-range transformers. *arXiv:2406.16747*, 2024.

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

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv:2502.09601*, 2025b.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *Proceedings of ICLR*, 2024.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv:2501.08313*, 2025.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. *arXiv:2502.20122*, 2025.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv:2407.19825*, 2024.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa

Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. In *Proceedings of ICLR*, 2021.

Alexander Peysakhovich and Adam Lerer. Attention sorting combats recency bias in long context language models. *arXiv:2310.01427*, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear RNNs with state expansion. In *Proceedings of COLM*, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. *https://qwenlm.github.io/blog/qwq-32b-preview/*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *Proceedings of COLM*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv:2503.04472*, 2025.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025. URL https://arxiv.org/abs/2407.04620.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv:2307.08621*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NeurIPS*, 2017.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR*, 2023.

Yibo Wang, Li Shen, Huanjin Yao, Tiansheng Huang, Rui Liu, Naiqiang Tan, Jiaxing Huang, Kai Zhang, and Dacheng Tao. R1-compress: Long chain-of-thought compression via chunk compression and search. *arXiv:2505.16838*, 2025.

Ziqi Wang, Hanlin Zhang, Xiner Li, et al. Eliminating position bias of language models: A mechanistic approach. *arXiv:2407.01100*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*, 2022.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv:2502.12067*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv:2309.17453*, 2023.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv:2503.06692*, 2025.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of ICML*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of NeurIPS*, 2024b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of NeurIPS*, 2023.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv:2502.15589*, 2025.

Liang Zhang, Yu Zhao, Longyue Wang, Tianqi Shi, Weihua Luo, Kaifu Zhang, and Jinsong Su. A state-transition framework for efficient llm reasoning, 2026. URL https://arxiv.org/abs/2602.01198.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of NeurIPS*, 2023.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. Marco-o1: Towards open reasoning models for open-ended solutions, 2024. URL https://arxiv.org/abs/2411.14405.

Zhipu AI. Glm-5: From vibe coding to agentic engineering. *arXiv:2602.15763*, 2026.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of ICLR*, 2023.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2025. URL https://arxiv.org/abs/2406.15877.

Context	Qwen2.5-1.5B			Qwen2.5-7B		
	Full-Attn	LISA	Reduction	Full-Attn	LISA	Reduction
4K	3.33 G	3.20 G	3.8%	14.69 G	14.74 G	−0.3%
8K	4.03 G	3.47 G	13.9%	16.34 G	15.37 G	5.9%
12K	4.73 G	3.74 G	21.0%	17.98 G	16.00 G	11.0%
16K	5.44 G	4.01 G	26.3%	19.63 G	16.63 G	15.3%

Table 5: Secondary decoding-FLOPs estimate per generated token (GFLOPs; SA + Indexer with cached keys + LA + SwiGLU MLP). LISA uses 5 full-attention + 23 LISA blocks. Primary efficiency claim: measured ReL.

A Decoding FLOPs Analysis

Table 5 reports a secondary, framework-agnostic estimate of per-token **decoding FLOPs** (attention block + SwiGLU MLP; excluding the embedding and LM-head). Our primary efficiency claim remains the measured eager-mode latency (ReL) in the main experiments. After selection, sparse self-attention in a LISA layer costs $O(M)$ per decoded token; Indexer scoring remains linear in the current context length and retained full-attention layers remain $O(t)$, so the overall sequence-level complexity is hybrid / quadratic and any FLOPs gain is a constant factor.

Attention block. The attention FLOPs of a single layer for one decoded token are

$$\mathrm{FLOPs}_{\mathrm{attn}} = \underbrace{2d(d+2d_{kv})}_{\mathrm{QKV}} + \underbrace{4\,n_h\,d_h\,L}_{\mathrm{QK}^\top+\mathrm{AV}} + \underbrace{2d^2}_{\mathrm{O}}, \tag{7}$$

where $d_{kv} = n_{kv}d_h$ and L is the number of keys attended in that layer. The full-attention baseline uses $L{=}t$ in all 28 layers. For LISA we use $L{=}t$ in the 5 retained full-attention blocks $(0, 7, 14, 21, 27)$ and $L{=}M{=}256$ in the sparse branch of the remaining 23 LISA layers, and we *add* linear-attention projections/state updates plus Indexer scoring with cached keys ($d_I{=}256$, $K{=}P/4$). With a key cache, Indexer cost per LISA layer is $4dKd_I + 2Kd_It$ (query/key projections for the new token and ReLU scores against t cached keys). Substituting the Qwen2.5-1.5B configuration ($d{=}1536$, $n_h{=}12$, $n_{kv}{=}2$, $d_h{=}128$, $K{=}3$) at $t{=}16$K yields 3.13 GFLOPs over 28 full-attention layers for the baseline attention component before MLP.

MLP. Qwen2.5 uses SwiGLU, so each layer costs three projections per token,

$$\mathrm{FLOPs}_{\mathrm{MLP}} = 6\,d\,d_{\mathrm{ff}}, \tag{8}$$

times 28 layers. LISA leaves the MLP unchanged: 2.31 GFLOPs for the 1.5B model ($d_{\mathrm{ff}}{=}8960$) and 11.41 GFLOPs for the 7B model ($d{=}3584$, $d_{\mathrm{ff}}{=}18944$).

Total and Table 5. Per-token decoding FLOPs $=\mathrm{FLOPs}_{\mathrm{SA}} + \mathrm{FLOPs}_{\mathrm{Indexer}} + \mathrm{FLOPs}_{\mathrm{LA}} + \mathrm{FLOPs}_{\mathrm{MLP}}$. For 1.5B at 16K this gives full attention 5.44 G versus LISA 4.01 G (about 26% reduction); for 7B the corresponding figure is about 15%. Savings are small or negligible at short context and grow with t, as expected once Indexer scoring is counted. The measured 23.9% ReL cut on 1.5B mathematical benchmarks is therefore the headline efficiency result; the table is only a rough upper-bound check under the accounting above.

B Indexer Output Distribution

Indexer Selection Patterns. We conduct an analysis of the Indexer's outputs to check alignment with known sparse-attention behaviors. By visualizing the top-256 tokens selected by the Indexer for two randomly sampled examples (Figure 3), we observe that the mechanism identifies locally recent tokens as well as some global landmarks.

Specifically, the distribution of selected tokens is heavily concentrated in the local context of the current output token, consistent with recency bias in transformers (Peysakhovich & Lerer, 2023;

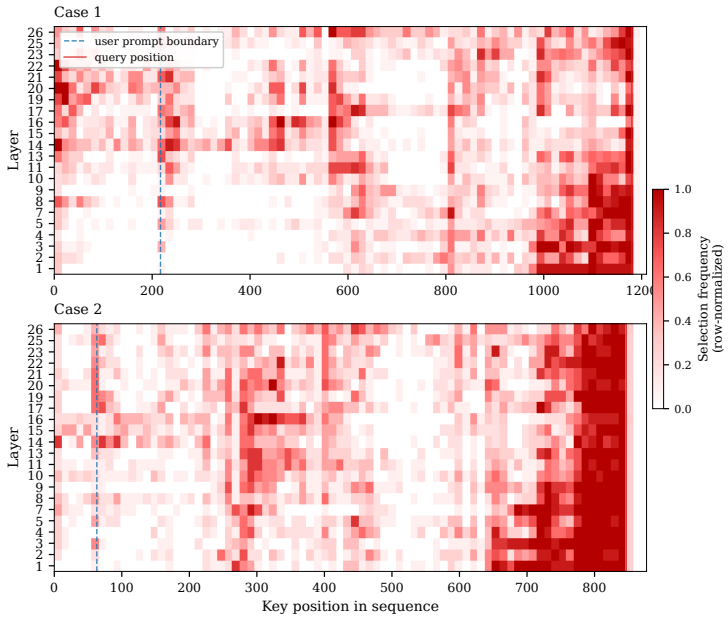

Figure 3: Visualization of the top-256 tokens selected by the Indexer across layers.

Wang et al., 2024; Xiao et al., 2023). Beyond the local window, the Indexer selectively attends to global features such as punctuation tokens, in line with prior sparse-attention observations (Lou et al., 2024; Bondarenko et al., 2023), and attention-sink behavior (Xiao et al., 2023) is visible. These qualitative case studies are illustrative rather than a proof that the Indexer recovers the full teacher attention distribution; quantitative agreement (e.g., top-M overlap or KL to the teacher) over a larger sample is left for future work.

The visualisations below repeat this analysis per layer: they show the top-256 keys selected by the LISA Indexer for the same two reasoning examples at different network depths, where the background colour intensity (white $\rightarrow$ dark red) reflects the attention score assigned to each candidate key token.

B.1 Layer 1 Indexer Output Distribution

Question: <|begin of sentence|> <|User|> The center of a circle touching the legs $A C$ and $B C$ of a right triangle $A B C$ lies on the hypotenuse $A B$. Find the radius of the circle, if it is six times smaller than the sum of the legs, and the area of triangle $A B C$ is 27.<|Assistant|> <think>

Response: <think>
Okay, so I have this problem about a right triangle ABC where the circle touches the legs AC and BC, and its center is on the hypotenuse AB. I need to find the radius of the circle, which is six times smaller than the sum of the legs. Also, the area of the triangle is 27. Hmm, let me break this down step by step.
First, let me visualize the triangle. Since it's a right triangle, let's assume angle C is the right angle. So, legs AC and BC form the right angle, and AB is the hypotenuse. The circle touches both legs AC and BC, so it must be tangent to both. The center of the circle lies on AB. That makes me think it's an incircle or maybe an excircle, but since it's touching both legs and the center is on the hypotenuse, it's probably not the standard incircle because the incenter of a

right triangle is located inside the triangle, not on the hypotenuse. Wait, but the problem says the center lies on the hypotenuse. So this must be a different circle, not the incircle.
Let me recall that if a circle is tangent to both legs of a right triangle and its center is on the hypotenuse, there might be a specific formula or relation here. Let me denote the legs as AC = b and BC = a, and hypotenuse AB = c. The area is (a*b)/2 = 27, so a*b = 54. Also, the radius r is six times smaller than the sum of the legs, so r = (a + b)/6. My goal is to find r.
Wait, so I need to relate r to a and b. Let me think about the coordinates. Maybe coordinate geometry can help here. Let me place point C at the origin (0,0), point B at (a,0), point A at (0,b). Then hypotenuse AB goes from (0,b) to (a,0). The center of the circle lies on AB and is equidistant from both legs AC and BC. Since the circle is tangent to both legs, which are the y-axis and x-axis respectively, the distance from the center to each axis must be equal to the radius. Wait, if the center is (r, r) for the incircle, but in this case, the center is on AB. So the center (h,k) must satisfy h = k = r? No, because the center is on AB. Wait, maybe the coordinates of the center can be parametrized along the hypotenuse.

Let me parametrize AB. The hypotenuse AB can be represented by the line from (0, b) to (a, 0). The equation of AB is y = (-b/a)x + b. Any point on AB can be written as (x, (-b/a)x + b). The center of the circle is at some point (x, (-b/a)x + b) on AB, and this point must be at a distance r from both legs AC and BC. The legs AC and BC are the y-axis and x-axis respectively, so the distance from the center to AC (y-axis) is x, and the distance to BC (x-axis) is y. Since the circle is tangent to both legs, these distances must be equal to the radius. So x = r and y = r. Therefore, the center is (r, r), but also lies on AB. Wait, but if the center is (r, r), then substituting into the equation of AB gives r = (-b/a)r + b. Let me check that.

The equation of AB is y = (-b/a)x + b. So if the center is at (r, r), then substituting x = r into the equation gives y = (-b/a)r + b. But the y-coordinate of the center is also r, so we have:
r = (-b/a)r + b
Let me solve for r:
r + (b/a)r = b
r(1 + b/a) = b
r = b / (1 + b/a) = (ab) / (a + b)
But from the problem statement, the radius r is (a + b)/6. So according to this, (ab)/(a + b) = (a + b)/6
Multiply both sides by (a + b):
ab = (a + b)^2 / 6
Multiply both sides by 6:
6ab = (a + b)^2
But we also know that ab = 54, so substitute:
6 * 54 = (a + b)^2
324 = (a + b)^2
Therefore, a + b = sqrt(324) = 18
So then r = (a + b)/6 = 18/6 = 3

Wait , so the radius is 3 ? Let me check if that makes sense .
Wait , so starting from the center being (r , r) because it ’s distance r from both axes , and since
it ’s on AB , substit uting into the equation gives r = (- b /a)r + b .
Then solving that gives r = (ab)/(a + b). Then according to the problem , r is (a + b)/ 6 . So equ ate them :
(ab)/(a + b) = (a + b)/ 6
Cross multiply :
6 ab = (a + b)^ 2
But since ab = 5 4 , 6 * 5

Question: <|begin of sentence|> <|User|> The solution set of $\{\log_{\frac{1}{2}}|x- \frac{\pi}{3}| \geq \log_{\frac{1}{2}} \frac{\pi}{2}$ is ( )
A: $\{\{x|-\frac{\pi}{6} \leq x \leq \frac{5\pi}{6}\}\}$
B: $\{\{x|x \leq -\frac{\pi}{6}, \text{ or } x \geq \frac{5\pi}{6}\}\}$
C: $\{\{x|-\frac{\pi}{6} \leq x \leq \frac{5\pi}{6} \text{ and } x \neq \frac{\pi}{3}\}\}$
D: $\{\{x|-\frac{5\pi}{6} \leq x \leq \frac{5\pi}{6} \text{ and } x \neq \frac{\pi}{3}\}\}$<|Assistant|><think>

Response: <think>
Okay, so I need to solve this logarithmic inequality: log base 1/2 of |x - π/3| is greater than or equal to log base 1/2 of π/2. Then choose the correct solution set from the options A to D. Let me think step by step.
First, I remember that logarithmic functions have certain properties, especially when dealing with inequalities. Since the base here is 1/2, which is between 0 and 1, the logarithmic function is decreasing. That means when I apply it to both sides of an inequality, the direction of the inequality will reverse. So, if $\log_b(a) \geq \log_b(c)$, then for $0 < b < 1$, $a \leq c$. Right? Let me confirm that. Yes, because as the input to a decreasing function increases, the output decreases. So, the original inequality $\log_{1/2}(A) \geq \log_{1/2}(B)$ implies $A \leq B$. Therefore, applying this property here, the given inequality $\log_{1/2}|x - \pi/3| \geq \log_{1/2}(\pi/2)$ implies that $|x - \pi/3| \leq \pi/2$. But wait, also, we need to consider the domain of the logarithm. The argument of the logarithm must be positive. So, $|x - \pi/3| > 0$. Which means $x \neq \pi/3$. Because absolute value is always non-negative, but logarithm is undefined at 0. So x cannot be $\pi/3$.
So, putting this together, the solution set is the set of all x such that $|x - \pi/3| \leq \pi/2$ and $x \neq \pi/3$. Now, let's solve $|x - \pi/3| \leq \pi/2$.
The absolute value inequality $|A| \leq B$ (where $B > 0$) is equivalent to $-B \leq A \leq B$. So here, $-\pi/2 \leq x - \pi/3 \leq \pi/2$. Let's solve for x.
First, add $\pi/3$ to all parts:
$-\pi/2 + \pi/3 \leq x \leq \pi/2 + \pi/3$
Compute $-\pi/2 + \pi/3$. To add these, find a common denominator. 6 is the least common multiple of 2 and 3. So:
$-\pi/2 = -3\pi/6$

π/3 = 2π/6
So, −3π/6 + 2π/6 = (−3π + 2π)/6 = −π/6
Similarly, π/2 + π/3:
π/2 = 3π/6
π/3 = 2π/6
3π/6 + 2π/6 = 5 π / 6
Therefore, the inequality |x − π/3| ≤ π / 2 simplifies to −π/6 ≤ x ≤ 5π/6. But remember x ≠ π/3 because |x − π/3| > 0.
So, the solution set is all x in [−π/6, 5π/6] except x = π/3.
Now, looking at the options:
A: {x | −π/6 ≤ x ≤ 5π/6} → Doesn't exclude π/3, so incorrect.
B: {x | x ≤ −π/6 or x ≥ 5π/6} → This is the complement of the interval, which would be if the inequality was reversed.
But since our inequality was |x − π/3| ≤ π/2, this is the outside, which is wrong.
C: { x | − π / 6 ≤ x ≤ 5 π / 6 and x ≠ π / 3 } → This matches our solution. Includes the interval and excludes π / 3.
D: { x | − 5 π / 6 ≤ x ≤ 5 π / 6 and x ≠ π / 3 } → The interval here is from − 5 π / 6 to 5 π/ 6, which is different. Let me check if I made a mistake in solving the inequality.
Wait, when solving | x − π / 3 | ≤ π / 2, adding π / 3 to − π / 2 gives − π / 6, and adding π / 3 to π / 2 gives 5 π / 6. That's correct.
So D is incorrect because the interval is from − 5 π / 6, which is more negative. So that's not matching. So the answer must be C.
But let me just verify once more. The original inequality is log_{1 /

B.2 Layer 26 Indexer Output Distribution

Question: <|begin of sentence|> <|User|> The center of a circle touching the legs $A C$ and $B C$ of a right triangle $A B C$ lies on the hypotenuse $A B$. Find the radius of the circle, if it is six times smaller than the sum of the legs, and the area of triangle $A B C$ is 27. <|Assistant|> <think>

Response: <think>

Okay, so I have this problem about a right triangle ABC where the circle touches the legs AC and BC, and its center is on the hypotenuse AB. I need to find the radius of the circle, which is six times smaller than the sum of the legs. Also, the area of the triangle is 27. Hmm, let me break this down step by step.
First, let me visualize the triangle. Since it's a right triangle, let's assume angle C is the right angle. So, legs AC and BC form the right angle, and AB is the hypotenuse. The circle touches both legs

AC and BC, so it must be tangent to both. The center of the circle lies on AB. That makes me think it's an incircle or maybe an excircle, but since it's touching both legs and the center is on the hypotenuse, it's probably not the standard incircle because the incenter of a right triangle is located inside the triangle, not on the hypotenuse. Wait, but the problem says the center lies on the hypotenuse. So this must be a different circle, not the incircle. Let me recall that if a circle is tangent to both legs of a right triangle and its center is on the hypotenuse, there might be a specific formula or relation here. Let me denote the legs as AC = b and BC = a, and hypotenuse AB = c. The area is (a*b)/2 = 27, so a*b = 54. Also, the radius r is six times smaller than the sum of the legs, so r = (a + b)/6. My goal is to find r.

Wait, so I need to relate r to a and b. Let me think about the coordinates. Maybe coordinate geometry can help here. Let me place point C at the origin (0,0), point B at (a,0), point A at (0,b). Then hypotenuse AB goes from (0,b) to (a,0). The center of the circle lies on AB and is equidistant from both legs AC and BC. Since the circle is tangent to both legs, which are the y-axis and x-axis respectively, the distance from the center to each axis must be equal to the radius. Wait, if the center is (r, r) for the incircle, but in this case, the center is on AB. So the center (h,k) must satisfy h = k = r? No, because the center is on AB. Wait, maybe the coordinates of the center can be parametrized along the hypotenuse. Let me parametrize AB. The hypotenuse AB can be represented by the line from (0, b) to (a, 0). The equation of AB is y = (-b/a)x + b. Any point on AB can be written as (x, (-b/a)x + b). The center of the circle is at some point (x, (-b/a)x + b) on AB, and this point must be at a distance r from both legs AC and BC. The legs AC and BC are the y-axis and x-axis respectively, so the distance from the center to AC (y-axis) is x, and the distance to BC (x-axis) is y. Since the circle is tangent to both legs, these distances must be equal to the radius. So x = r and y = r. Therefore, the center is (r, r), but also lies on AB. Wait, but if the center is (r, r), then substituting into the equation of AB gives r = (-b/a)r + b. Let me check that. The equation of AB is y = (-b/a)x + b. So if the center is at (r, r), then substituting x = r into the equation gives y = (-b/a)r + b. But the y-coordinate of the center is also r, so we have:

r = (-b/a)r + b

Let me solve for r:
r + (b/a)r = b
r(1 + b/a) = b
r = b / (1 + b/a) = (ab) / (a + b)
But from the problem statement, the radius r is (a + b)/6. So according to this, (ab)/(a + b) = (a + b)/6
Multiply both sides by (a + b):
ab = (a + b)^2 / 6
Multiply both sides by 6:
6ab = (a + b)^2
But we also know that ab = 54, so substitute:
6 * 54 = (a + b)^2
324 = (a + b)^2
Therefore, a + b = sqrt(324) = 18
So then r = (a + b)/6 = 18 / 6 = 3
Wait, so the radius is 3? Let me check if that makes sense.
Wait, so starting from the center being (r, r) because it's distance r from both axes, and since it's on AB, substituting into the equation gives r = (-b/a)r + b. Then solving that gives r = (ab)/(a + b). Then according to the problem, r is (a + b)/6. So equate them:
(ab)/(a + b) = (a + b)/6
Cross multiply:
6ab = (a + b)^2
But since ab = 54, 6 * 54

Question: <|begin of sentence|><|User|>The solution set of $\{}log_{\{}frac{1}{2}}|x-\{}frac{\{}pi}{3}|\{}geq \{}log_{\{}frac{1}{2}}\{}frac{\{}pi}{2}$ is ( )
A: $\{}{x|-\{}frac{\{}pi}{6}\{}leq x\{}leq\{}frac{5\{}pi}{6}\{} }$
B: $\{}{x|x\{}leq -\{}frac{\{}pi}{6}, \{}text{ or } x\{}geq \{}frac{5\{}pi}{6}\{} }$
C: $\{}{x|-\{}frac{\{}pi}{6}\{}leq x\{}leq\{}frac{5\{}pi}{6}\{}text{ and } x\{}neq\{}frac{\{}pi}{3}\{}}$
D: $\{}{x|-\{}frac{5\{}pi}{6}\{}leq x\{}leq\{}frac{5\{}pi}{6}\{}text{ and } x\{}neq\{}frac{\{}pi}{3}\{}}$<|Assistant|><think>

Response: <think>

Okay, so I need to solve this logarithmic inequality: log base 1/2 of $|x - \pi/3|$ is greater than or equal to log base 1/2 of $\pi/2$. Then choose the correct solution set from the options A to D. Let me think step by step.
First, I remember that logarithmic functions have certain properties, especially when dealing with inequalities. Since the base here is 1/2, which is between 0 and 1, the logarithmic function is decreasing. That means when I apply it to both sides of an inequality, the direction of the inequality will reverse. So, if $\log_b(a) \geq \log_b(c)$, then for $0 < b < 1$, $a \leq c$. Right? Let me confirm that. Yes, because as the input to a decreasing function increases, the output decreases. So, the original inequality $\log_{1/2}(A) \geq \log_{1/2}(B)$ implies $A \leq B$.
Therefore, applying this property here, the given inequality $\log_{1/2}|x - \pi/3| \geq \log_{1/2}(\pi/2)$ implies that $|x - \pi/3| \leq \pi/2$.
But wait, also, we need to consider the domain of the logarithm. The argument of the logarithm must be positive. So, $|x - \pi/3| > 0$. Which means $x \neq \pi/3$. Because absolute value is always non-negative, but logarithm is undefined at 0. So x cannot be $\pi/3$.
So, putting this together, the solution set is the set of all x such that $|x - \pi/3| \leq \pi/2$ and $x \neq \pi/3$. Now, let's solve $|x - \pi/3| \leq \pi/2$. The absolute value inequality $|A| \leq B$ (where $B > 0$) is equivalent to $-B \leq A \leq B$. So here, $-\pi/2 \leq x - \pi/3 \leq \pi/2$. Let's solve for x.
First, add $\pi/3$ to all parts:
$-\pi/2 + \pi/3 \leq x \leq \pi/2 + \pi/3$
Compute $-\pi/2 + \pi/3$. To add these, find a common denominator. 6 is the least common multiple of 2 and 3. So:
$-\pi/2 = -3\pi/6$
$\pi/3 = 2\pi/6$
So, $-3\pi/6 + 2\pi/6 = (-3\pi + 2\pi)/6 = -\pi/6$
Similarly, $\pi/2 + \pi/3$:
$\pi/2 = 3\pi/6$
$\pi/3 = 2\pi/6$
$3\pi/6 + 2\pi/6 = 5\pi/6$
Therefore, the inequality $|x - \pi/3| \leq \pi/2$ simplifies to $-\pi/6 \leq x \leq 5\pi/6$. But remember $x \neq \pi/3$ because $|x - \pi/3| > 0$.
So, the solution set is all x in $[-\pi/6, 5\pi/6]$ except $x = \pi/3$.
Now, looking at the options:

A: {x | $-\pi/6 \leq$ x $\leq 5\pi/6$} $\rightarrow$ Doesn't exclude $\pi/3$, so incorrect.
B: {x | x $\leq -\pi/6$ or x $\geq 5\pi/6$} $\rightarrow$ This is the complement of the interval, which would be if the inequality was reversed. But since our inequality was $|x - \pi/3| \leq \pi/2$, this is the outside, which is wrong.
C: {x | $-\pi/6 \leq$ x $\leq 5\pi/6$ and x $\neq \pi/3$} $\rightarrow$ This matches our solution. Includes the interval and excludes $\pi/3$.
D: {x | $-5\pi/6 \leq$ x $\leq 5\pi/6$ and x $\neq \pi/3$} $\rightarrow$ The interval here is from $-5\pi/6$ to $5\pi/6$, which is different. Let me check if I made a mistake in solving the inequality.
Wait, when solving $|x - \pi/3| \leq \pi/2$, adding $\pi/3$ to $-\pi/2$ gives $-\pi/6$, and adding $\pi/3$ to $\pi/2$ gives $5\pi/6$. That's correct. So D is incorrect because the interval is from $-5\pi/6$, which is more negative. So that's not matching. So the answer must be C.
But let me just verify once more. The original inequality is log -{ 1 /

C Inference-Time Segment State Update

We follow the segment-level state correction of Zhang et al. (2026) (Liang Zhang et al.) and restate it briefly for completeness. During inference, noisy reasoning segments can mislead subsequent generation and push LA state transitions off a stable trajectory. The correction uses a global reasoning direction with a constant coefficient α.

We partition the generated token sequence into segments of fixed size k. Let $\mathbf{S}^{(l)}_{\text{tok}}$ denote the LA state after tok generated tokens in layer l. The state change (reasoning direction) of the t-th segment is

$$\nabla^l_t = \mathbf{S}^{(l)}_{k\cdot t} - \mathbf{S}^{(l)}_{k\cdot(t-1)}. \tag{9}$$

Noisy steps typically produce ∇_t that deviate from the global trend. At the end of segment t, we first aggregate prior *corrected* directions via a running average,

$$\bar{\nabla}^l_{t-1} = \frac{1}{t-1}\sum_{i=1}^{t-1} \hat{\nabla}^l_i, \tag{10}$$

with $\bar{\nabla}^l_0$ a zero matrix, and then mix the current raw direction with this history,

$$\hat{\nabla}^l_t = (1-\alpha)\nabla^l_t + \alpha\bar{\nabla}^l_{t-1}, \tag{11}$$

where α is a fixed correction coefficient. Finally, the segment-level state used going forward is

$$\mathbf{S}^{(l)}_{k\cdot t} \leftarrow \mathbf{S}^{(l)}_{k\cdot(t-1)} + \hat{\nabla}^l_t. \tag{12}$$

This pulls the current segment update toward historical trends and limits accumulation of local noise. We follow Zhang et al. (2026) for this correction rule; LISA's novelty is only in applying it jointly with the Indexer-guided sparse branch.