DySCO: Dynamic Attention-Scaling Decoding for Long-Context Language Models

Anonymous authors
Paper under double-blind review

Abstract

Understanding and reasoning over long contexts is a crucial capability for language models (LMs). Although recent models support increasingly long context windows, their accuracy often deteriorates as input length grows: models struggle to keep attention focused on the most relevant context throughout decoding. We propose DySCO, a training-free decoding algorithm for long-context reasoning. At each decoding step, DySCO uses *retrieval heads*—a small subset of attention heads specialized for long-context retrieval—to identify task-relevant tokens, and explicitly up-weights attention to them across all heads. DySCO applies directly to off-the-shelf LMs and is compatible with FlashAttention, as its intervention reduces to a per-token logit bias inside the attention kernel. Across three model families, spanning instruction-tuned and reasoning models, DySCO consistently improves performance on challenging long-context reasoning benchmarks, with relative gains of 22% and 21% for Qwen3-8B on MRCR and LongBenchV2 at 128K context length. DySCO incurs about 1.8× decoding latency but makes more effective use of test-time compute, outperforming self-consistency baselines with up to 8 parallel samples. Further analysis shows that both dynamic rescaling and retrieval-head guided selection are important to its effectiveness, and provides insights into decoding-time attention behavior.

1 Introduction

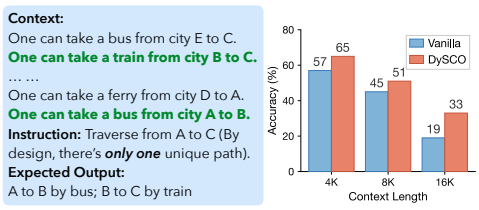

Figure 1: Left: A simplified Path Traversal task, which requires dynamically locating relevant context during decoding. Right: Qwen3-8B accuracy drops sharply with context length, even at only 16K tokens.

Recent advances in language models (LMs) have enabled processing of extremely long context windows, unlocking applications such as repository-level code understanding and long-document question answering. Driven by improvements in data curation and transformer architectures, modern LMs now support context lengths of 128K tokens and beyond (Gemini Team, 2025; Yang et al., 2025; Kimi Team, 2025; OpenAI, 2025; Anthropic, 2026). However, model performance often degrades significantly as input length increases, even on simple tasks, commonly known as "context rot" (Hong et al., 2025; Goldman et al., 2024). Figure 1 illustrates this phenomenon using a simple Path Traversal task (Ye et al., 2025), which requires LMs to trace a path in a graph where each node has exactly one outgoing edge. Recent models such as Qwen3-8B and Llama-3.1-8B see an accuracy drop from around 60% at 4K tokens to below 20% at 16K tokens, despite supporting context lengths of up to 128K tokens.

We attribute this degradation to a key challenge in how LMs utilize long and information-dense contexts during generation: effective long-context reasoning requires models to continually shift attention toward the most task-relevant parts of the context as the generation state evolves. In practice, however, vanilla decoding often fails to exhibit this behavior over long contexts. For example, in the Path Traversal task shown in Figure 1, LMs must iteratively identify the next edge from the context based on the current node. Our analysis (§2.2) shows that attention is often insufficiently focused on the relevant context at each generation step, leading to errors even when the required information is present. In this work, we propose a novel training-free decoding algorithm, DySCO

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

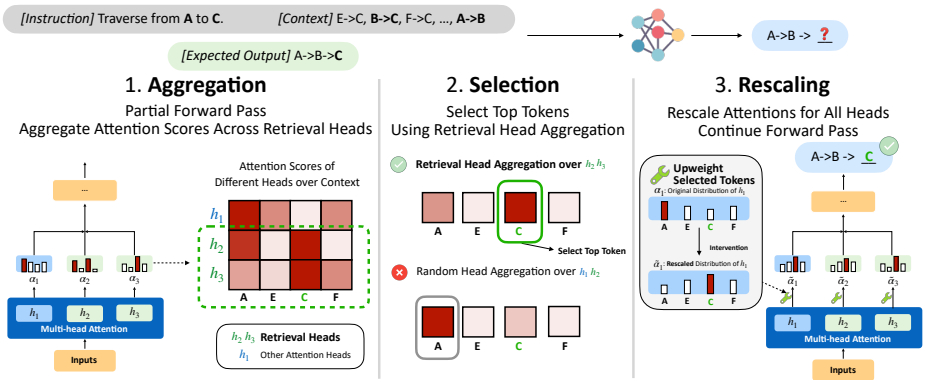

Figure 2: Overview of DySCO. At each decoding step: (1) **Aggregation**: a partial forward pass obtains attentions of retrieval heads and uses them to assign relevance scores to context tokens; (2) **Selection**: the most relevant tokens are selected; (3) **Rescaling**: these tokens are up-weighted in the attention logits of all heads, and a full forward pass samples the next token.

(**Dy**namic Attention-**S**caling De**CO**ding), which dynamically adjusts attention on the fly during generation. DySCO operates entirely at the decoding stage, requiring no modification to model parameters or inputs. This makes DySCO highly flexible, allowing it to be applied directly to off-the-shelf LMs and selectively enabled based on the input length. DySCO can be viewed as a form of test-time scaling (Snell et al., 2025; Qu et al., 2025), trading additional decoding compute for improved long-context accuracy.

The key idea of DySCO is to identify relevant tokens at each decoding step using *retrieval heads* (Wu et al., 2025a; Zhang et al., 2025c), and then up-weight attention to these tokens during generation. Retrieval heads are a specialized subset (1–2%) of attention heads that are responsible for long-context retrieval: compared to other heads, they assign higher and more stable attention to context that is critical for next-token prediction. Notably, we find that retrieval heads remain consistently focused on relevant tokens even when overall model performance degrades with increasing context length (§2.2). After prefilling, DySCO operates in three stages at each decoding step (Figure 2). 1) *aggregation*: we run a partial forward pass (§3.1) to aggregate attention scores from retrieval heads; 2) *selection*: we identify a small set of context tokens with the highest aggregated attention scores; 3) *rescaling*: we up-weight attention to the selected tokens by intervening on the attention logits of *all* heads and run a full forward pass to generate the next token.

DySCO introduces a lightweight mechanism for *dynamic* attention shaping during decoding. Prior work on attention scaling largely relies on *static* patterns. For example, long-context extension methods uniformly rescale all attention logits by a constant factor (Peng et al., 2024; Chen et al., 2026; Nakanishi, 2025). Other approaches impose fixed, position-dependent patterns (Hsieh et al., 2024b; Zhang et al., 2024c) to mitigate the "lost-in-the-middle" issue (Liu et al., 2024a), which is less pronounced in modern LMs (e.g., Qwen3-8B performs strongly on RULER (Yang et al., 2025)). In contrast, our method performs *dynamic, token-selective* scaling driven by internal signals from retrieval heads. Retrieval heads have also been previously used for improving long-context training pipelines through data curation (Qiu et al., 2025), auxiliary objectives (Liu et al., 2025), or localized interventions (Zhu et al., 2025). Our work shows that they can be seamlessly integrated into the decoding procedure, leading to improved long-context reasoning without any additional training.

We evaluate DySCO on both instruction-tuned and reasoning LMs across a wide range of long-context tasks. DySCO consistently improves performance on challenging long-context reasoning benchmarks. Notably, for Qwen3-8B, DySCO delivers 22% and 21% relative improvements on MRCR (Vodrahalli et al., 2024) and LongBenchV2 (Bai et al., 2025a) at 128K compared to YaRN alone. DySCO is also compatible with FlashAttention (Dao et al., 2022): the aggregation stage recomputes attention only for the small set of retrieval heads, and the rescaling stage adds a per-token bias inside the kernel's online softmax (§3.2). The extra partial forward pass at each decoding step incurs about 1.8× decoding latency, yet DySCO makes more effective use of test-time compute than parallel sampling, outperforming self-consistency baselines with up to 8 parallel samples (§4.3).

Further analysis highlights the importance of both dynamic scaling and the use of retrieval heads for identifying important tokens. In summary, we introduce dynamic attention scaling, a new way of shaping attention in long-context LMs. Our results demonstrate that DySCO leads to consistent gains in long-context reasoning while offering mechanistic insights into LMs' long-context behavior.

2 Background and Motivation

We first set up the preliminaries of retrieval heads (Wu et al., 2025a; Zhang et al., 2025c), an important building block of our approach (§2.1). We then study Path Traversal, a synthetic task designed to stress-test basic long-context reasoning, and use it to reveal the connection between retrieval heads and long-context reasoning capabilities (§2.2).

2.1 Preliminaries: Retrieval Heads

We consider an autoregressive transformer LM $\mathcal{M}$ (Vaswani et al., 2017) that generates the next token x_{t+1} conditioned on the prefix $x_{\leq t}$, which includes both the input context and previously generated tokens. Let $\mathcal{H} = \{h_i\}$ denote the set of attention heads in $\mathcal{M}$. At decoding step t, each head $h \in \mathcal{H}$ produces an attention distribution: $\boldsymbol{\alpha}_t^{(h)} \in \mathbb{R}^t$, where $\alpha_{t,i}^{(h)}$ is the attention mass for token $x_i \in \boldsymbol{x}_{\leq t}$.

Retrieval Heads. Wu et al. (2025a) discovered a universal set of attention heads that exhibit copy-like behavior during decoding, which they term *retrieval heads*. Concretely, when generating token x_{t+1}, a retrieval head h concentrates its attention on a prior occurrence of the same token in the context. That is, $\alpha_{t,i}^{(h)}$ is high for some $i \leq t$ such that $x_i = x_{t+1}$. These heads are typically sparse (less than 5% of all heads) and provide a mechanistic explanation for how language models perform explicit token lookup and copy-paste from long context.

Query-Focused Retrieval Heads (QRHead). Retrieval heads can be identified using different criteria (Zhang et al., 2025c; Qiu et al., 2025; Zhu et al., 2025). Our method builds on QRHead (Zhang et al., 2025c), which identifies attention heads based on query–context attention mass using examples from realistic long-context tasks. We adopt QRHead due to its strong generalization across both diverse domains in the BEIR re-ranking benchmark (Thakur et al., 2021) and a range of challenging long-context reasoning tasks. Additional details on QRHead are provided in Appendix A.

2.2 Retrieval Heads Stay Focused on Relevant Context

Diagnostic task: Path Traversal. Long-context reasoning requires dynamically attending to relevant information at each decoding step, conditioned not only on the inputs but also on the intermediate state encoded in $x_{\leq t}$. To analyze this capability in a controlled setting, we use a synthetic task, Path Traversal (Ye et al., 2025). Path Traversal takes a list of edges and requires finding a path from a start to a target node. The graph is constructed so that each node on the gold path has exactly one outgoing edge, reducing the task to iteratively retrieving the correct next node (a deterministic process explicitly specified in the prompt). We control context length (up to 32K) by varying the number of nodes. Despite its simplicity, performance degrades sharply with longer context: step-level accuracy drops from about 90% to ∼20% at 32K tokens (Figure 3). This highlights the need for repeated, dynamic retrieval as a core challenge of long-context reasoning. Further details are provided in Appendix I.

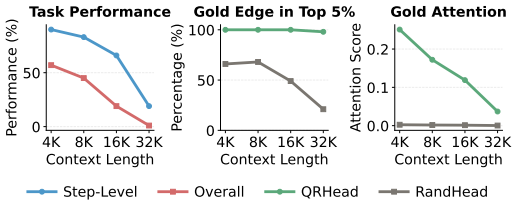

Figure 3: Qwen3-8B on Path Traversal. **Left**: Accuracy. **Middle**: How often the gold edge ranks in the top 5% of edges by attention. **Right**: Attention mass on gold edges. QRHead keeps ranking gold edges highly even as accuracy and attention mass decline.

Behavior of retrieval heads. We further analyze the behavior of retrieval heads, which provides insight into LMs' failure modes on long-context reasoning tasks. We partition the context into

Algorithm 1 DySCO

Input: LM $\mathcal{M}$ with attention heads $\mathcal{H}$, retrieval heads (QRHead) $\mathcal{H}^*$, input $\boldsymbol{x}_{\leq T}$, rescale strength β, momentum γ, cumulative probability threshold p (for token selection).
Output: Generated sequence $\boldsymbol{x}_{\leq T'}$

1: $t \leftarrow T$; $\mathbf{r}_T \leftarrow \mathbf{r}_{\text{init}}$ `// Initialize relevance`
2: **while** not finished **do**
3: $t \leftarrow t + 1$
4: **(Aggregation)** Run a partial forward pass to obtain attention logits $\mathbf{a}_t^{(h)}$ for $h \in \mathcal{H}^*$
5: $\mathbf{r}_t \leftarrow \gamma \cdot \mathbf{r}_{t-1} + (1-\gamma) \cdot \frac{1}{|\mathcal{H}^*|} \sum_{h \in \mathcal{H}^*} \text{Softmax}(\mathbf{a}_t^{(h)})$ `// Momentum`
6: **(Selection)** $\boldsymbol{x}^* \leftarrow \text{SelectTop}(\boldsymbol{x}_{\leq t}, \mathbf{r}_t, p)$ `// Select top-p tokens`
7: **(Rescaling)** $\mathbf{v}[i] \leftarrow \log(\beta)$ if $x_i \in \boldsymbol{x}^*$, else 0; $\tilde{\boldsymbol{\alpha}}_t^{(h)} \leftarrow \text{Softmax}\big(\mathbf{a}_t^{(h)} + \mathbf{v}\big)$ for all $h \in \mathcal{H}$
8: $x_{t+1} \sim \text{Softmax}\big(\mathcal{M}(\boldsymbol{x}_{\leq t} \mid \tilde{\boldsymbol{\alpha}}_t)\big)$ `// Rescaled forward`
9: **end while**

edge-level spans and compute the sum of attention mass assigned by QRHead to each span at every decoding step. Our analysis focuses on two metrics: 1) the fraction of decoding steps for which the gold edge (i.e., the next edge on the correct path) appears among the top 5% most-attended spans (Gold Edge in Top 5%). 2) the total attention mass assigned to the gold edge at each decoding step (Gold Attention). For this analysis (Figure 3, middle and right), we only consider the second step (edge) among the four steps (each path has four edges) to explicitly study the attention dynamics occurring in the middle of the reasoning process.

We additionally include randomly selected attention heads as a baseline for comparison. As shown in Figure 3 (middle), **retrieval heads remain consistently aligned with the gold edge.** Even as single-step prediction accuracy drops from about 90% to approximately 20% with increasing context length, retrieval heads continue to rank the gold edge among the top 5% at most steps. At the same time, **we observe a sharp decline in absolute attention to the gold edge**, closely mirroring the overall performance degradation (see Figure 3 right). In contrast, random attention heads exhibit a much steeper drop in relative attention and much lower attention mass on gold spans.

Steering attention with retrieval heads. Our analysis finds that QRHead consistently ranks gold edges highly and preserves strong relative retrieval signals. However, as context length grows, absolute attention to gold spans declines across all heads, leading to performance degradation. This suggests that while QRHead identifies relevant context, its signal is diluted. Motivated by this observation, we investigate whether the stable retrieval behavior of QRHead can be leveraged to steer overall attention toward relevant context at decoding time.

3 DySCO: Dynamic Attention Scaling

The core idea of DySCO is to dynamically *up-weight* tokens in the context based on the attention distribution of retrieval heads during decoding. As shown in Figure 2, each decoding step of DySCO contains three stages: **Aggregation** of QRHead attention into context relevance scores via a partial forward pass, **Selection** of the most relevant tokens, and **Rescaling** of their attention logits across *all* heads in a full forward pass. DySCO only modifies the decoding procedure, requires no training, and relies solely on attention scores produced by the model itself. It can thus be directly applied to off-the-shelf LMs without architectural changes or task-specific preprocessing.

3.1 The DySCO Algorithm

Algorithm 1 details the full procedure of DySCO. Given an LM $\mathcal{M}$ with attention heads $\mathcal{H}$ and a set of retrieval heads (QRHead) $\mathcal{H}^* \subset \mathcal{H}$, we decode from an input sequence $\boldsymbol{x}_{\leq T}$. At each decoding step $t \geq T+1$, DySCO maintains context relevance scores $\boldsymbol{r} = r_1, ..., r_t$ where each r_i denotes the relevance of each token x_i in the prefix $\boldsymbol{x}_{\leq t}$ for the current generation step. For clarity of presentation, we assume the relevance scores are properly initialized (line 1), and defer the details of initialization after we introduce the aggregation step. DySCO produces the next token in three stages:

Aggregation. We first perform a *partial* forward pass at decoding step t, and extract the attention distributions of the selected retrieval heads. The forward pass is partial as we skip the forward pass

over higher layers given that retrieval heads are primarily located in the middle layers (see §3.2 for details). We then compute a *relevance score* over tokens in the prefix by *averaging* the attention scores across these heads: $\mathbf{r}_t = \frac{1}{|\mathcal{H}^*|} \sum_{h \in \mathcal{H}^*} \boldsymbol{\alpha}_t^{(h)}$.

where $\boldsymbol{\alpha}_t^{(h)}$ is the attention distribution after softmax at step t. To incorporate information from previous decoding steps, we maintain a moving average of the relevance scores with momentum $\gamma \in [0, 1)$: $\mathbf{r}_t \leftarrow \gamma \cdot \mathbf{r}_{t-1} + (1 - \gamma) \cdot \mathbf{r}_t$. Empirically, this momentum-based smoothing stabilizes the relevance estimates and makes DySCO more robust to hyperparameter variations.

We note that at the first decoding step, we need to obtain an initial relevance score. We use a short warm-up window of length T_w, relying on the last T_w tokens of the input prompt to initialize the relevance distribution. Specifically, at the last prompt position T (before the first decoding step), we compute $\mathbf{r}_T = \mathrm{Normalize}\big(\sum_{d=0}^{T_w-1} \gamma^d \cdot \boldsymbol{\alpha}_{T-d}^{\mathcal{H}^*}\big)$, where $\boldsymbol{\alpha}_{T-d}^{\mathcal{H}^*} = \frac{1}{|\mathcal{H}^*|} \sum_{h \in \mathcal{H}^*} \boldsymbol{\alpha}_{T-d}^{(h)}$ is the averaged attention of QRHead at time $T - d$, and $\mathrm{Normalize}(\cdot)$ rescales the scores to form a valid distribution. We *do not* tune γ or T_w, and fix them to $\gamma = 0.4$ and $T_w = 8$ in all experiments.

Selection. At each decoding step t, we use the relevance distribution to estimate the importance of context tokens in the prefix $\boldsymbol{x}_{\leq t}$. Based on this distribution, we select a subset of relevant tokens $\boldsymbol{x}^* \subseteq \boldsymbol{x}_{\leq t}$. Our selection strategy follows nucleus (top-p) sampling (Holtzman et al., 2020) used for LM decoding, but is applied over relevance scores rather than next-token probabilities. Specifically, we rank tokens in $\boldsymbol{x}_{\leq t}$ by their relevance scores and retain the smallest set of tokens whose cumulative relevance mass exceeds a threshold p, while additionally enforcing a maximum of K tokens (4K or 8K) to avoid an overly large number of selected tokens. In practice, we find performance to be robust to moderate variations in parameters. In all experiments, we determine these parameters using a validation set of fully synthetic tasks, and directly use the resulting parameters on diverse downstream datasets without further tuning (§4). Concretely, we set top-p to be 0.95 or 0.975.

Rescaling. Given the selected relevant tokens $\boldsymbol{x}^*$, we intervene on the attention computation to up-weight the selected tokens by modifying the attention logits. Concretely, for each layer and each attention head, we compute an intervention vector of bias terms added to the attention logits. The bias is $\log \beta$ for the selected tokens, and 0 otherwise, where $\beta > 1$ is the rescale strength factor: $\mathbf{v}[i] = \log(\beta)$ if $x_i \in \boldsymbol{x}^*$, else 0.

Then, we add the intervention vector to the attention logits $\mathbf{a}_t^{(h)}$ of *all* attention heads $h \in \mathcal{H}$ before the softmax operation, and obtain rescaled attention distributions $\tilde{\boldsymbol{\alpha}}_t^{(h)} = \mathrm{Softmax}\big(\mathbf{a}_t^{(h)} + \mathbf{v}\big)$. We apply the intervention to all heads to propagate the relevance signal identified by QRHead throughout the model's attention computation. This is equivalent to multiplying the unnormalized attention weights of the selected tokens by β. Finally, we perform a full forward pass with the modified attention logits across all layers, and the next token x_{t+1} is sampled or selected from the output distribution produced by this rescaled forward pass. Importantly, this procedure operates entirely at inference time and does not require any modification to model parameters.

3.2 Efficient Implementation

Early stopping of attention aggregation. We describe our design that makes DySCO more efficient. Prior work shows that attention heads responsible for retrieval and reasoning tend to concentrate in the middle layers of transformer models (Wu et al., 2025a; Zhao et al., 2025). For instance, for Qwen3-8B (36 layers), QRHead are located in layers 17–20. Hence, we early-stop attention aggregation during the partial forward pass and only collect QRHead attention up to the middle layers of the model, rather than across all layers. With early stopping, this adds approximately 60% extra computation *per decoding step*.

Overhead analysis. As mentioned above, the attention aggregation stage introduces an additional *partial* forward pass during decoding, while leaving prefilling unchanged. DySCO adds no peak memory, since peak memory is dominated by prefilling. Its main cost is decoding latency: the extra partial forward pass incurs about 1.8× decoding latency. DySCO is compatible with FlashAttention (Dao et al., 2022): the aggregation stage recomputes attention only for the 16 selected heads, and the rescaling stage adds a per-token bias inside the kernel's online softmax, which preserves its

block-wise, IO-aware computation. We provide measured throughput and latency, together with the implementation details, in Appendix D.

4 EXPERIMENTS

We test the effectiveness of DySCO on a diverse array of long-context reasoning tasks.

Models. We experiment with multiple open-weight LMs from two families, including **Qwen3-4B, 8B, and 32B** of the Qwen3 family (Yang et al., 2025), and **Llama-3.1-8B-Instruct** (Llama-3 Team, 2024). Qwen3 models support a thinking mode that uses long CoT (Guo et al., 2025), and Llama-3.1-8B-Instruct can use CoT prompting (Wei et al., 2022). We further evaluate Mixtral-8x7B-Instruct-v0.1, a mixture-of-experts model, in §4.3.

Head selection. We directly use the set of QRHEAD (Zhang et al., 2025c) from the official implementation across all experiments. We note that these heads are detected on Natural Questions (Kwiatkowski et al., 2019) and applied directly across all tasks evaluated in this paper, demonstrating strong cross-task transfer.

Baselines. Our approach operates on the decoding process for improving long-context reasoning. We compare against baselines that directly operate on decoding, including: 1) **Vanilla**, standard decoding without alteration. 2) **YaRN** (Peng et al., 2024), which extends context window by modifying rotary position embeddings (Su et al., 2024). YaRN primarily targets extending the usable context window of pretrained models, rather than improving long-context reasoning accuracy. 3) **Uniform Attention Scaling (UNIATTNS)**. Prior work has shown that sharpening attention distributions can improve long-context extrapolation (Peng et al., 2024; Chen et al., 2026; Nakanishi, 2025). Concretely, this is achieved by applying a temperature $\tau \in (0, 1]$ to rescale attention logits, i.e., $\mathbf{a}' = \mathbf{a}/\tau$, where $\mathbf{a}$ denotes the original attention logits. Following this line of work, we adopt length-dependent attention temperature scaling, using different values of τ for different input lengths. While prior approaches typically scale attention with a factor that grows linearly with $\log n$, we instead tune τ separately for each context length. Importantly, UNIATTNS rescales attention logits *uniformly* across all tokens, whereas DySCO selectively up-weights step-relevant tokens at each decoding step. We note that both UNIATTNS and DySCO **can be combined with YaRN**.

Setting parameters for DySCO and UNIATTNS. DySCO and UNIATTNS both require setting additional parameters for controlling the decoding. We use MRCR (Vodrahalli et al., 2024) (Multi-Round Coreference Resolution) as a development dataset for deciding hyperparameters. We use MRCR as it is a fully synthetically generated dataset, which makes it less biased for a particular real-world dataset. For each model, we pick one set of parameters for length span (0,64K) and one set of parameters for length span (64K, 128K) using performance on MRCR, and we directly apply the *same set of parameters* on other datasets. We include more details in Appendix G.

Diagnostic test on PATH TRAVERSAL. We first use PATH TRAVERSAL to evaluate whether reshaping attention with DySCO or UNIATTNS helps LMs maintain focus on relevant context at each decoding step. We do not include YaRN in this setting, as the maximum input length (32K) is well within the supported context window of the evaluated models. Results are reported in Table 1. As shown, DySCO improves over vanilla decoding in all settings except Qwen3-32B at 4K, with the largest gains at 16K (+14 points for both models). UNIATTNS also yields gains in some settings (e.g., +8 points for Qwen3-32B at 16K), though they are smaller and less consistent. These results suggest that attention sharpening alone can yield modest improvements, whereas explicitly up-weighting important tokens identified by QRHEAD leads to substantially larger gains.

Table 1: PATH TRAVERSAL accuracy with Uniform Attention Scaling (UNIATTNS) and DySCO.

PATH TRAVERSAL	4K	8K	16K	32K
Qwen3-32B	80	67	32	21
+ UNIATTNS	**82**	68	40	19
+ DySCO	79	**71**	**46**	**33**
Qwen3-8B	57	45	19	1
+ UNIATTNS	59	46	25	**2**
+ DySCO	**65**	**51**	**33**	**2**

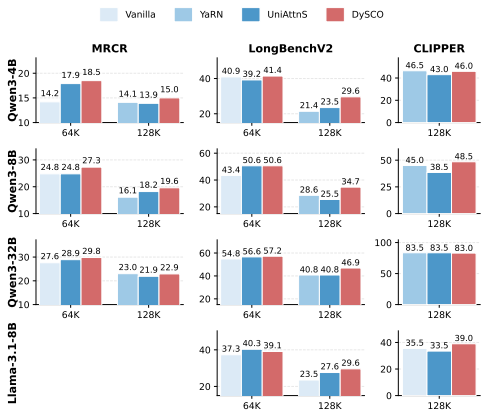

Figure 4: Results on MRCR, LONGBENCHV2, and CLIPPER. Qwen models use YaRN at 128K; Llama-3.1-8B-Instruct natively supports 128K.

Table 2: Performance on long-context (128K input) recall tasks. We apply YaRN for Qwen models.

	MRCR 128K	HOTPOT 128K	INFMC 128K	INFQA 128K	Avg 128K
Qwen3-4B^{YaRN}	16.7	37	50	27.1	32.7
+ UNIATTNS	18.4	34	39	24.1	28.9
+ DYSCO	**19.0**	**39**	**54**	**31.3**	**35.8**
Qwen3-8B^{YaRN}	16.7	**61**	61	39.6	44.6
+ UNIATTNS	17.5	58	63	**39.7**	44.6
+ DYSCO	**19.5**	58	**65**	39.2	**45.4**
Qwen3-32B^{YaRN}	24.4	51	**79**	**48.7**	50.8
+ UNIATTNS	24.2	51	78	48.6	50.5
+ DYSCO	**25.3**	**53**	77	**48.7**	**51.0**
Llama-3.1-8B-Inst.	17.2	46	61	37.1	40.3
+ UNIATTNS	**18.9**	51	56	35.7	40.4
+ DYSCO	18.6	**52**	**62**	**38.4**	**42.8**

4.1 LONG CONTEXT REASONING TASKS

We now turn to the primary focus of this paper: multi-step CoT reasoning over long context.

Datasets and settings. We briefly summarize the datasets and settings used for our experiments. Please refer to Appendix C for more details.

We use three datasets with natural texts: 1) **MRCR** (Vodrahalli et al., 2024) requires LMs to retrieve a conversation following a query from a list of highly similar conversations. 2) **LONGBENCHV2** (Bai et al., 2025a) requires multi-step reasoning over realistic long contexts spanning diverse domains. 3) **CLIPPER** (Pham et al., 2025) evaluates claim verification over full-length books (90–128K tokens), requiring multi-step reasoning over evidence distributed throughout the text.

For Qwen3 models, we activate thinking mode to use long CoT (4096-token budget for MRCR, CLIPPER and a 10,240-token budget for LONGBENCHV2). For 128K-context experiments, we use YaRN (factor 4.0) for Qwen3 models to extend their context window from 32K to 128K, following the recommendation in Yang et al. (2025). We note that both UNIATTNS and DYSCO are applied on top of YaRN for 128K. For Llama models, we use CoT prompting template from original datasets to enable CoT. We do not evaluate Llama models on MRCR in this setting, as they do not support long CoT (direct recall results are reported in §4.2).

Results. As shown in Figure 4, **DYSCO improves CoT reasoning under long-context settings** across model families, benchmarks, and context lengths. At 128K context length, DYSCO (with YaRN) improves Qwen3-8B by 3.5 absolute points (22% relative) on MRCR, 6.1 points (21%) on LONGBENCHV2, and **3.5** points on CLIPPER compared to YaRN alone. While UNIATTNS yields gains in some cases (e.g., a 7.2-point improvement for Qwen3-8B on LONGBENCHV2 at 64K), these improvements are not consistent across datasets or context lengths. In contrast, DYSCO provides larger and more stable gains, particularly at longer context lengths, and outperforms UNIATTNS in nearly all settings.

4.2 LONG CONTEXT RECALL TASKS

We next evaluate DYSCO on long-context recall tasks, which primarily follow a needle-in-a-haystack setup (Kamradt, 2023). For these tasks, models directly produce the final answer without using CoT (we turn off thinking for Qwen models).

Datasets. We evaluate MRCR under its intended direct-answering setup (directly outputting the final response). Additionally, we consider the following tasks: HotpotQA (Yang et al., 2018), QA and multiple-choice tasks from InfBench (Zhang et al., 2024a). For Hotpot and InfBench, we use

Table 3: Ablations of DySCO on head selection and dynamic rescaling with Qwen3-8B. DySCO outperforms its ablations with random attention heads and with static scaling.

	MRCR 64K		LBV2 64K	Path 16K	InfMC 128K
Think	✓	✗	✓	✗	✗
Vanilla	24.8	24.8	43.4	19	61
DySCO	**27.3**	**26.3**	**50.6**	**33**	**66**
w/ RandomHead	26.6	24.2	47.0	20	63
w/ StaticScaling	22.5	23.1	47.5	23	64

Table 4: Comparison between DySCO and self-consistency (SC; 4 or 8 samples) with Qwen3-8B (Think). All methods use YaRN at 128K.

	LBV2 64K	LBV2 128K	Clip 128K
Vanilla	43.4	28.6	45.0
SC (4 samples)	45.8	29.6	45.5
SC (8 samples)	47.6	32.7	46.5
DySCO	**50.6**	**34.7**	**48.5**

the processed versions released in HELMET (Yen et al., 2025), and evaluate all tasks with a context length setting of 128K tokens.

Results. As shown in Table 2, DySCO is also able to improve model accuracy on long-context recall tasks. For example, with Llama-3.1-8B-Instruct, DySCO increases HotpotQA accuracy from 46 to 52, and InfQA accuracy from 37.1 to 38.4. While UniAttnS yields modest gains on MRCR, it also degrades performance in several other cases. Notably, these recall tasks require only very short outputs (typically tens of tokens), and DySCO introduces negligible overhead.

4.3 Ablations & Analysis

Ablation: Importance of head selection. We compare DySCO instantiated with QRHead against a variant that uses randomly selected heads. Both variants use 16 heads. Table 3 reports results on multiple datasets with Qwen3-8B. DySCO with random heads can outperform vanilla decoding on some tasks, since random heads may still capture weak retrieval signals (§2.2). However, using QRHead consistently yields the best results across all settings, highlighting the advantage of more reliable relevance signals with head selection.

Ablation: Importance of dynamic rescaling. We further ablate the role of dynamic rescaling by comparing DySCO with a static rescaling variant. Static rescaling applies attention reweighting to a *fixed* set of context tokens, selected during an initial warm-up stage, and reuses this set throughout decoding. Conceptually, static rescaling can be viewed as an adaptation of KV-cache eviction methods that avoids explicit context pruning, as it relies on similar criteria for identifying important tokens (e.g., SnapKV (Li et al., 2024a)). As shown in Table 3, DySCO with static rescaling yields improvements over vanilla on recall tasks (e.g., InfMC), but dynamic rescaling is still consistently better than static scaling, highlighting its importance in DySCO, as the set of relevant tokens can shift over the course of generation.

Analysis: Comparison with test-time scaling. DySCO trades additional test-time computation for improved long-context reasoning accuracy. To contextualize this tradeoff, we compare against self-consistency (Wang et al., 2023) (SC), a standard test-time scaling method that draws multiple samples and takes a majority vote over final answers. Table 4 reports results with Qwen3-8B (Think). DySCO incurs less decoding overhead than both variants of SC (4 samples and 8 samples) while still outperforming both of them. Full details are provided in Appendix L.

Analysis: DySCO with Mixture-of-Experts models. We additionally evaluate the effectiveness of DySCO on Mixtral-8x7B-Instruct-v0.1, a sparse MoE model (Jiang et al., 2024a). DySCO improves over both vanilla decoding and UniAttnS without any MoE-specific adaptation to head detection. Full results are provided in Appendix K.

Analysis: Hyperparameter robustness. We examine the robustness of DySCO to variations of its hyperparameters. DySCO shows consistent performance across reasonable choices of hyperparameters (details in Appendix H). DySCO also remains effective even when only a fraction of the QRHead set is correctly identified (details in Appendix J).

Analysis: Impact of context length. We analyze how the effectiveness of DySCO varies with context length by comparing results from §4.1 and §4.2. The gains are particularly pronounced for CoT reasoning under long-context settings. See Appendix F for details.

Analysis: Additional comparisons. We provide further comparisons in Appendix E, showing that DySCO outperforms retrieval-augmented generation and prompt compression approaches.

Analysis: Qualitative examples. We provide qualitative examples showcasing how DySCO effectively steers LMs in long-context reasoning in Appendix M.

Discussion: scope of our approach. DySCO is designed for long-context regimes where the input context is substantially longer than the output. As an inference-time technique, DySCO can be applied selectively at deployment: it can be enabled for long inputs, and disabled for shorter inputs. This flexibility allows practitioners to adapt the method based on input length, since its benefits are most pronounced in long-context settings rather than universally across all generation regimes.

5 Related Work

We focus our discussion on closely related inference-time techniques for long-context LMs. Please refer to Appendix B for more general methods for long-context modeling.

Efficient long-context inference techniques. One line of work studies inference-time techniques for long context, primarily focusing on improving efficiency. Representative approaches include streaming inference (Xiao et al., 2024; Zhang et al., 2023), KV cache eviction or compression (Xu et al., 2024; Li et al., 2024a; Cai et al., 2024; Corallo & Papotti, 2024; Kim et al., 2024), and sparse attention mechanisms (Xu et al., 2025; Jiang et al., 2024b; Xiao et al., 2025). These methods trade off accuracy for computational or memory efficiency. In contrast, DySCO is explicitly designed to improve *accuracy* under long context.

Inference techniques for long-context modeling. Recent work has explored improving long-context modeling at inference time, including training on input context at test time (Bansal et al., 2025; Chen et al., 2025c) and learning intervention vectors over query and key matrices (Zhu et al., 2025). These approaches typically require additional training, while DySCO requires no training. Most closely related to our approach, prior work has explored modifying attention distributions to better utilize long context, including global attention scaling methods (Peng et al., 2024; Nakanishi, 2025; Chen et al., 2026; Puvvada et al., 2025) and positional calibration methods (Hsieh et al., 2024b; Zhang et al., 2024c) that mitigate the "lost-in-the-middle" issue (Liu et al., 2024a). In contrast, DySCO performs selective rescaling and up-weights attention to task-relevant tokens, unlocking more effective long-context reasoning.

Scaffolding and external systems for long context. Another line of work builds external scaffolding around LMs. This includes RAG (Zhao et al., 2024b; Li et al., 2024b), prompt compression modules (Jiang et al., 2024c; Wu et al., 2025b), recursive or multi-stage LM calls (Zhang et al., 2025a), memory systems (Chen et al., 2024a), and agentic frameworks (Zhang et al., 2024b; Zhao et al., 2024a). DySCO directly improves the LM's internal attention behavior during decoding, without introducing additional scaffolding, and can be integrated with any scaffolding.

6 Conclusion

We have presented DySCO, a training-free decoding algorithm that improves long-context reasoning. DySCO leverages retrieval heads to dynamically identify relevant context tokens at each decoding step and intervenes on attention logits to up-weight their attention across heads. Across multiple instruct and reasoning LMs, DySCO consistently improves accuracy on challenging long-context benchmarks. Our analyses further indicate that long-context failures are associated with degraded focus on relevant context, and that dynamic attention scaling can mitigate this issue. Overall, DySCO provides an inference-time approach for improving long-context accuracy and offers insight into the mechanisms underlying long-context failures in LMs.

AI USE STATEMENT

Generative AI tools were used to support improvements in writing quality and to assist with debugging code. The authors carefully reviewed all AI-supported material and remain fully responsible for the final manuscript, its claims, and all associated artifacts.

REFERENCES

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. Make Your LLM Fully Utilize the Context. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Anthropic. Claude 4.6 Opus, 2026. URL https://www.anthropic.com/news/claude-opus-4-6.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria, July 2025a. Association for Computational Linguistics.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongWriter: Unleashing 10,000+ word generation from long context LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Kale, and Samy Jelassi. Let's (not) just put things in context: Test-time training for long-context llms. *ArXiv*, abs/2512.13898, 2025.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling, 2024.

Guanzheng Chen, Xin Li, Michael Shieh, and Lidong Bing. LongPO: Long context self-evolution of large language models through short-to-long preference optimization. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Howard Chen, Ramakanth Pasunuru, Jason E Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading, 2024a.

Jianghao Chen, Junhong Wu, Yangyifan Xu, and Jiajun Zhang. LADM: Long-context training data selection with attention-based dependency measurement for LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, July 2025b.

Shi Chen, Zhengjiang Lin, Yury Polyanskiy, and Philippe Rigollet. Critical attention scaling in long-context transformers. In *The Fourteenth International Conference on Learning Representations*, 2026.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. *ArXiv*, abs/2507.06415, 2025c.

Giulio Corallo and Paolo Papotti. FINCH: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532, 2024. doi: 10.1162/tacl_a_00716.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128K context. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fu24d.html.

Chaochen Gao, Xing W, Zijia Lin, Debing Zhang, and Songlin Hu. NExtlong: Toward effective long-context training without long documents. In *Forty-second International Conference on Machine Learning*, 2025a.

Muhan Gao, TaiMing Lu, Kuai Yu, Adam Byerly, and Daniel Khashabi. Insights into LLM long-context failures: When transformers know but don't tell. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, November 2024.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025b.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv*, abs/2507.06261, 2025.

Omer Goldman, Alon Jacovi, Aviv Slobodkin, Aviya Maimon, Ido Dagan, and Reut Tsarfaty. Is It Really Long Context if All You Need Is Retrieval? Towards Genuinely Difficult Long Context NLP. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2024. URL https://aclanthology.org/2024.emnlp-main.924.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://research.trychroma.com/context-rot.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=kIoBbc76Sy.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, August 2024b.

Zhiyuan Hu, Yuliang Liu, Jinman Zhao, Suyuchen Wang, Yan Wang, Wei Shen, Qing Gu, Anh Tuan Luu, See-Kiong Ng, Zhiwei Jiang, et al. LongRecipe: Recipe for Efficient Long Context Generalization in Large Language Models. *arXiv preprint arXiv:2409.00509*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Barbara Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *ArXiv*, abs/2401.04088, 2024a. URL https://api.semanticscholar.org/CorpusID:266844877.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, 2024b.

Huiqiang Jiang, Qianhui Wu, , Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.91.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM Maybe LongLM: Self-Extend LLM Context Window Without Tuning. In *Forty-first International Conference on Machine Learning*, 2024.

Gregory Kamradt. Needle In A Haystack - pressure testing LLMs, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot: Infinite context processing on memory-constrained LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, November 2024.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, November 2024b.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Haim Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avshalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: Hybrid Transformer-Mamba language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://api.semanticscholar.org/CorpusID:268793596.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Weihao Liu, Ning Wu, Shiping Yang, Wenbiao Ding, Shining Liang, Ming Gong, and Dongmei Zhang. MuDAF: Long-context multi-document attention focusing through contrastive learning on attention heads. *ArXiv*, abs/2502.13963, 2025. URL https://api.semanticscholar.org/CorpusID:276449816.

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. LongGenBench: Long-context generation benchmark. In *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP Findings)*. Association for Computational Linguistics, November 2024b. URL https://aclanthology.org/2024.findings-emnlp.48.

Llama-3 Team. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ken M. Nakanishi. Scalable-softmax is superior for attention. *ArXiv*, abs/2501.19399, 2025.

OpenAI. OpenAI GPT-5.2, 2025. URL https://developers.openai.com/api/docs/models/gpt-5.2.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

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

Chau Minh Pham, Yapei Chang, and Mohit Iyyer. CLIPPER: Compression enables long-context synthetic data generation. In *Second Conference on Language Modeling*, 2025.

Krishna C Puvvada, Faisal Ladhak, Santiago Akle Serano, Cheng-Ping Hsieh, Shantanu Acharya, Somshubra Majumdar, Fei Jia, Samuel Kriman, Simeng Sun, Dima Rekesh, and Boris Ginsburg. SWAN: An efficient and scalable approach for long-context language modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2025.

Yifu Qiu, Varun R. Embar, Yizhe Zhang, Navdeep Jaitly, Shay B Cohen, and Benjamin Han. Eliciting in-context retrieval and reasoning for long-context large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, July 2025.

Yuxiao Qu, Matthew Y. R. Yang, Amrith Rajagopal Setlur, Lewis Tunstall, Edward Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://api.semanticscholar.org/CorpusID:276928248.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jef frey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Venkatesh Ramasesh, Quoc Le, Ed Huai hsin Chi, Yifeng Lu, Orhan Firat, Angeliki Lazaridou, Jean-Baptiste Lespiau, Nithya Attaluri, and Kate Olszewska. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *ArXiv*, abs/2409.12640, 2024.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Xiangkun Hu, Zheng Zhang, Qian Wang, and Yue Zhang. NovelQA: Benchmarking Question Answering on Documents Exceeding 200K Tokens. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2403.12766.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:247595263.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=EytBpUGB1Z.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization. *ArXiv*, abs/2509.13313, 2025b. URL https://api.semanticscholar.org/CorpusID:281325930.

Yuhao Wu, Ming Shan Hee, Zhiqing Hu, and Roy Ka-Wei Lee. LongGenBench: Benchmarking Long-Form Generation in Long Context LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cFu7ze7xUm.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. Recycled attention: Efficient inference for long-context language models. *arXiv preprint arXiv:2411.05787*, 2024.

Guo-Hao Xu, Jingzhen Ding, Huping Ding, Zhao Xu, and Kaifu Zhang. FTP: Efficient prefilling for long-context LLM inference via FFN token pruning, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shi-Qiang Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *ArXiv*, abs/2505.09388, 2025. URL https://api.semanticscholar.org/CorpusID:278602855.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. LongProc: Benchmarking long-context language models on long procedural generation. In *Second Conference on Language Modeling*, 2025.

Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. In *Proceedings of the Annual Conference of the Association for Computational Linguistics (ACL)*, pp. 2588–2610, 2024.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context language models effectively and thoroughly. In *The Thirteenth International Conference on Learning Representations*, 2025.

Fangcong Yin, Xi Ye, and Greg Durrett. LoFiT: Localized fine-tuning on LLM representations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dfiXFbECSZ.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. Jasper and Stella: distillation of SOTA embedding models, 2025b. URL https://arxiv.org/abs/2412.19048.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1214. URL https://aclanthology.org/2025.emnlp-main.1214/.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024a. URL https://arxiv.org/abs/2402.13718.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan O Arik. Chain of agents: Large language models collaborating on long-context tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=LuCLf4BJsr.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. LONGAGENT: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, November 2024a.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. LongRAG: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, November 2024b.

Xinyu Zhao, Fangcong Yin, and Greg Durrett. Understanding synthetic context extension via retrieval heads. In *Proceedings of ICML*, 2025.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. PoSE: Efficient context window extension of LLMs via positional skip-wise training. In *The Twelfth International Conference on Learning Representations*, 2024.

Youxiang Zhu, Ruochen Li, Danqing Wang, Daniel Haehn, and Xiaohui Liang. Focus directions make your language models pay more attention to relevant contexts. *ArXiv*, abs/2503.23306, 2025.

A Additional Details on Query-Focused Retrieval Heads (QRHead)

Original retrieval heads are identified through strict copy behavior, which overlooks more general semantic retrieval. To address this limitation, Zhang et al. (2025c) propose Query-Focused Retrieval Heads (QRHeads), which generalize retrieval behavior to query-conditioned context lookup.

Specifically, consider a long-context QA setting in which the input consists of a query q and a large context containing one gold document d^* (the "needle") along with many distractor documents. Zhang et al. (2025c) define a query-context retrieval score (QRScore) for each attention head h as:

$$\mathrm{QRScore}(h) = \sum_{i \in q} \sum_{j \in d^*} \alpha_{i,j}^{(h)},$$

where $\alpha_{i,j}^{(h)}$ denotes the attention weight from query token i to context token j under head h. Heads are ranked by their QRScore, and the top-ranked heads (typically 1–2% of all attention heads) are selected as QRHeads.

Unlike original retrieval heads, QRHeads capture semantic context lookup rather than exact token copying. Prior work shows that attention from QRHeads is more effective for retrieving relevant information from long contexts across tasks and domains (Zhang et al., 2025c).

B Extended Related Work

In the main body of the paper, we focus on closely related inference-time techniques; here, we provide an overview of work on improving long-context LMs more broadly.

The need for better support of long context has motivated extensive research across multiple dimensions, including architectural designs (Gu & Dao, 2024; Lieber et al., 2025; Peng et al., 2023; Xiao et al., 2024; Bertsch et al., 2023; Jin et al., 2024; Yen et al., 2024), data engineering strategies (Xiong et al., 2024; An et al., 2024; Gao et al., 2025b; Hu et al., 2024; Fu et al., 2024; Chen et al., 2025a; Gao et al., 2025a; Bai et al., 2025b; Chen et al., 2025b), context window extension techniques (Peng et al., 2024; Chen et al., 2024b; 2023; Zhu et al., 2024), evaluation benchmarks (Yen et al., 2025; Ye et al., 2025; Bai et al., 2024; 2025a; Liu et al., 2024b; Wu et al., 2025c; Hsieh et al., 2024a; Wang et al., 2025), and analyses of long-context behavior and failure modes (Wu et al., 2025a; Zhao et al., 2025; Goldman et al., 2024; Liu et al., 2024a; Gao et al., 2024; Yin et al., 2024). Most existing approaches improve long-context performance by modifying model parameters, architectures, or training data. In contrast, our work explores a new direction: improving long-context accuracy purely at inference time by modifying the decoding procedure.

C Details of Experimental Setup

Datasets. We use 3 datasets with natural texts and enable CoT as follows:

1) **MRCR** (Vodrahalli et al., 2024) requires LMs to retrieve a conversation following a query from a list of highly similar conversations. The task is introduced to test LMs' direct recall. To allow LMs to use CoT, we activate thinking mode (4096-token budget) for Qwen3 models. We do not evaluate Llama models in this setting, as they do not support long CoT (direct recall results are reported in §4.2).

2) **LongBenchV2** (Bai et al., 2025a) requires multi-step reasoning over realistic long contexts spanning diverse domains. We use a subset of data points with input length ranging from 0–128K (264 data points). For Qwen models, we enable thinking mode with a maximum of 10,240 tokens to accommodate the difficulty; for Llama models, we apply CoT prompting from Bai et al. (2025a).

3) **Clipper** (Pham et al., 2025) evaluates claim verification over full-length books. The model is given a complete book (90–128K tokens) along with a claim to verify, requiring multi-step reasoning over evidence distributed throughout the text. We include this dataset because it provides a native

CoT prompting template, which is well-suited for evaluating instruction-tuned models (Llama-3.1-8B-Instruct). We follow the original experimental setup and response template from Pham et al. (2025).

Settings. We use YaRN (factor 4.0) for Qwen3 models to extend their context window from 32K to 128K, following the recommendation in Yang et al. (2025). For 128K-context experiments, both UNIATTNS and DYSCO are applied on top of YaRN. For Llama models with explicit CoT prompting, we use greedy decoding. For Qwen3 models, we use the recommended generation configurations, while fixing the random seed for decoding across different methods per run, following Bansal et al. (2025); Bai et al. (2025a) to control the variance across methods.

D DETAILED DISCUSSION OF OVERHEADS

We begin by briefly reviewing the standard generation procedure of language models. Given an input prefix $x_{\leq T}$, the model generates outputs $x_{T+1:T'}$ autoregressively until termination. This process consists of two phases: (1) *prefilling*, where the full input $x_{\leq T}$ is processed in a single forward pass to construct KV caches, and (2) *autoregressive decoding*, where tokens are generated one at a time conditioned on the cached states.

DYSCO operates only at the autoregressive decoding phase, while the prefilling stage remains unchanged and can leverage standard optimizations such as FlashAttention (Dao et al., 2022; Dao, 2024). This design ensures that DYSCO introduces no additional peak memory overhead from prefilling, and isolates all extra cost to decoding-time computation. We analyze this overhead in terms of FLOPs, memory, and throughput below.

Table 5: Throughput (tokens/s) comparison between vanilla decoding and DYSCO, under standard attention (EAGER) and our Flash-compatible implementation (Flash; §D.1). Results are obtained on Qwen3-8B with 128K input tokens and 4K output tokens on an H100 GPU.

	EAGER	Flash
Vanilla	4.8	13.9
DYSCO	2.8	7.9

Peak memory. DYSCO does not increase peak memory usage. In standard decoding, peak memory typically occurs during prefilling, which processes the full input sequence in a single forward pass. In contrast, DYSCO only materializes attention patterns for a small subset of heads during decoding (i.e., $|\mathcal{H}^*| \times 1 \times T$), rather than full attention maps across all heads and tokens.

As a result, the peak memory footprint remains dominated by prefilling. For example, with a 100K-token input and 4K-token output, prefilling requires approximately 39GB of memory, while decoding requires only $\sim$32GB. DYSCO does not exceed this bound.

Throughput. As mentioned in §3.2, the attention aggregation stage introduces an additional *partial* forward pass during decoding, which reduces throughput. Table 5 reports measured throughput for generating 4K tokens conditioned on 128K inputs: DYSCO is 1.7–1.8$\times$ slower than vanilla decoding. This reflects our design that DYSCO sacrifices decoding latency for improved long-context accuracy. We note that further optimizations could be possible. For example, interventions could be applied only from intermediate layers after identifying relevant signals from QRHEAD; we leave this for future work.

FLOPs. Only the decoding stage incurs overhead and there is no additional cost during prefilling. As a result, the total FLOP overhead of DYSCO remains small for long-context workloads, where computation is dominated by the quadratic-cost prefilling pass over the input context. To make this concrete, consider a setting with a 128K-token input context. Following Hoffmann et al. (2022), our estimate of the FLOPs is that generating 4K and 8K output tokens accounts for roughly 4% and 8.4% of the prefilling FLOPs, respectively. When using DYSCO, the additional partial decoding pass increases the total computation by only about 2.5% and 5.0% relative to prefilling for these two settings. We note that FLOPs understate the practical cost in terms of wall-clock latency (Table 6).

D.1 FLASH-COMPATIBLE IMPLEMENTATION

While DYSCO requires materializing attention scores for a small subset of attention heads, it can still be implemented in a manner compatible with FlashAttention. We describe the implementation details below; all Flash throughput and latency numbers in this section are measured with this implementation, and we will release the code, including the modified kernel.

Aggregation stage. Recall that the aggregation stage (§3.1) requires access to the attention distributions of QRHEAD in order to compute relevance scores $\mathbf{r}_t$. Standard FlashAttention kernels never materialize these probabilities, since they are consumed on-chip inside the fused kernel. We therefore keep the model's main forward pass on ordinary (unmodified) FlashAttention, and separately recompute attention probabilities only for the small set of selected heads $\mathcal{H}^*$, using the same query/key projections (after normalization and RoPE) already produced by that forward pass. Because decoding involves a single query position ($L_q = 1$), this recomputation materializes only $O(|\mathcal{H}^*| \cdot L_k)$ scores per layer with selected heads, rather than a full attention tensor; we use $|\mathcal{H}^*| = 16$ throughout.

Rescaling stage. The rescaling stage adds a position-dependent logit bias $\mathbf{v}$ before the softmax (the rescaling step in §3.1). This is compatible with FlashAttention's block-wise, online-softmax computation: the bias only needs to be added to each key-tile's scores *before* the running-maximum and normalizer are updated, and does not change the structure of the streaming computation. Concretely, inside each score tile, we add the bias to the unscaled $QK^\top$ accumulator (dividing by the kernel's softmax scale c, since the accumulator at that point is not yet scaled), and apply the existing causal/padding mask before the online-softmax rescaling step. We apply this modification consistently to both the ordinary forward path and the split-key-value (split-K) forward path that FlashAttention uses for long key/value sequences with few queries, since modifying only one path would miss decoding steps that are automatically routed to the other.

Algorithm 2 summarizes one Flash-compatible decoding step at this level of abstraction, and Algorithm 3 details how the bias enters the tile-level online-softmax update inside the (modified) FlashAttention kernel itself.

Algorithm 2 Flash-compatible DYSCO decoding step

Input: current token x_t, cache $\mathcal{C}$, previous relevance $\mathbf{r}_{t-1}$, selected heads $\mathcal{H}^*$, strength β
`// (Aggregation) unmodified FlashAttention forward, probing selected heads`
for layer l up to the deepest layer containing heads in $\mathcal{H}^*$ **do**
  $q_l, k_l, v_l \leftarrow \mathrm{ProjectAndRoPE}(x_t, l)$
  $K_l, V_l \leftarrow \mathrm{Concat}(\mathcal{C}[l], k_l, v_l)$ `// not committed to cache yet`
  $h_l \leftarrow \mathrm{FlashAttention}(q_l, K_l, V_l)$ `// unmodified kernel`
  if l contains heads in $\mathcal{H}^*$ **then**
    $\boldsymbol{\alpha}_t^{\mathcal{H}^*} \leftarrow \mathrm{Softmax}\big(q_l^{\mathcal{H}^*}(K_l^{\mathcal{H}^*})^\top/\sqrt{d}\big)$ `// selected heads only, aggregation step in §3.1`
  end if
end for
$\mathbf{r}_t \leftarrow \gamma \cdot \mathbf{r}_{t-1} + (1-\gamma) \cdot \mathrm{Mean}_{\mathcal{H}^*}(\boldsymbol{\alpha}_t^{\mathcal{H}^*})$ `// Selection follows as in Alg. 1`
$\mathbf{v} \leftarrow \mathrm{Bias}(\mathrm{SelectTop}(\mathbf{r}_t, p), \beta)$
`// (Rescaling) full forward with bias fused into FlashAttention`
for every layer l **do**
  $q_l, k_l, v_l \leftarrow \mathrm{ProjectAndRoPE}(x_t, l)$
  $K_l, V_l \leftarrow \mathrm{Append}(\mathcal{C}[l], k_l, v_l)$ `// committed once`
  $h_l \leftarrow \mathrm{BiasedFlashAttention}(q_l, K_l, V_l, \mathbf{v})$ `// modified kernel, Alg. 3`
end for
$\ell_t \leftarrow \mathrm{LMHead}(h)$; sample $x_{t+1} \sim \mathrm{Softmax}(\ell_t)$
return x_{t+1}, updated cache $\mathcal{C}$, $\mathbf{r}_t$

Because the additive bias only changes the pre-softmax scores seen by this streaming update, and not the traversal order or tile structure, the modification preserves FlashAttention's IO-aware, block-

Algorithm 3 Biased online-softmax tile update inside the modified FlashAttention kernel

Input: query q, key/value tiles $\{K_J, V_J\}$, per-key bias $\mathbf{v}$, scale $c = 1/\sqrt{d}$
Initialize running max $m \leftarrow -\infty$, normalizer $\ell \leftarrow 0$, output accumulator $u \leftarrow \mathbf{0}$
for each key/value tile J in kernel traversal order **do**
  $z \leftarrow c \cdot (qK_J^\top) + \mathbf{v}[J]$ `// bias added before rescaling, not after`
  Set z to $-\infty$ at masked/invalid positions
  $m_{\text{new}} \leftarrow \max(m, \max(z))$; $\alpha \leftarrow \exp(m - m_{\text{new}})$ ($\alpha = 0$ if $\ell = 0$)
  $p \leftarrow \exp(z - m_{\text{new}})$
  $u \leftarrow \alpha \cdot u + pV_J$; $\ell \leftarrow \alpha \cdot \ell + \sum p$; $m \leftarrow m_{\text{new}}$
end for
return u/ℓ `// final normalized attention output for this head`

wise computation and its $O(N)$ memory footprint in the number of keys N; it does not require materializing an $N \times N$ attention matrix at any point.

Latency w.r.t. different input and output lengths. To further characterize this overhead across realistic long-context generation settings, Table 6 reports end-to-end wall-clock decoding latency for both EAGER and Flash-compatible implementations, for 64K and 128K input contexts with 4K- and 10K-token outputs.

Table 6: End-to-end decoding latency (seconds) for vanilla decoding and DySCO, under EAGER attention and our Flash-compatible implementation, across input/output length settings.

Input	Output	Vanilla EAGER	DySCO EAGER	Vanilla Flash	DySCO Flash
64K	4K	432.8	727.2	147.3	269.7
64K	10K	1053.2	1777.4	359.6	629.5
128K	4K	859.5	1453.9	294.5	519.3
128K	10K	2098.1	3555.6	719.1	1249.0

We view closing the remaining gap between vanilla and DySCO decoding throughput, e.g., selectively applying interventions to a subset of layers, as promising future engineering work.

E Comparison with RAG and Prompt Compression

We compare our decoding-time approach with methods that reduce the effective context length via external scaffolding. Specifically, we consider two representative classes of approaches: (1) Retrieval-Augmented Generation (RAG), which employs a dense retriever to select the most relevant portions of the input context given a query; and (2) Prompt compression methods, which explicitly rewrite or prune the input prompt to fit within a shorter token budget. In particular, we evaluate LongLLMLingua (Jiang et al., 2024c), which uses an auxiliary LM to compress long prompts while attempting to preserve task-relevant information.

Both RAG and prompt compression methods are less flexible than DySCO. They require additional system components (e.g., external retrievers or compression models) and rely on explicit partitioning of the input into instruction, context, and query segments. In contrast, DySCO operates purely at decoding time on the original input sequence, without modifying the prompt or introducing any external scaffolding.

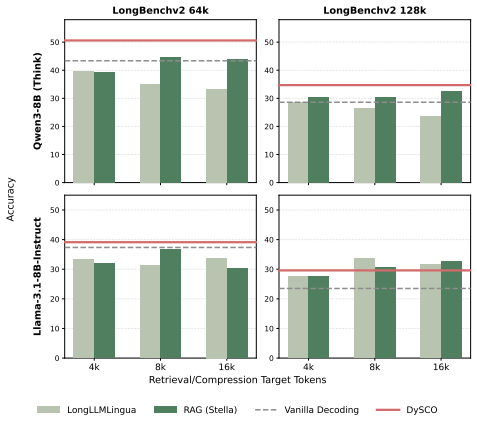

Figure 5: Comparison between DySCO, RAG (Stella), LongLLMLingua, and vanilla decoding. For RAG and LongLLMLingua, we report the results after reducing the context to different lengths (4K, 8K, and 16K tokens).

We evaluate DySCO, RAG, and LongLLMLingua on the challenging long-context reasoning benchmark LongBenchV2. We do not include benchmarks such as MRCR and InfBench, which are specifically designed to stress-test long-context processing and are largely solvable by RAG-style designs.

Setup. For RAG, we adopt a strong dense retriever, Stella-1.5B-V5 (Zhang et al., 2025b). Following Bai et al. (2025a), we apply a chunk-and-concatenate strategy to reduce the context length. Specifically, we first partition the original long context into chunks of 1024 tokens. Given a query, Stella V5 retrieves the top-$\{4, 8, 16\}$ most relevant chunks, which are then concatenated in their original order, resulting in shortened contexts of 4K, 8K, and 16K tokens. For LongLLMLingua, we use the official implementation released by Jiang et al. (2024c). We partition the input prompt into context and question components, set the target compressed length to $\{4\text{K}, 8\text{K}, 16\text{K}\}$ tokens, and feed the compressed prompt to the model for decoding.

Results. Figure 5 summarizes the results. Overall, DySCO outperforms both RAG and LongLLMLingua on Qwen3-8B (Think), and achieves performance comparable to these baselines on Llama-3.1-8B. As both models have undergone substantial post-training and already exhibit strong long-context capabilities, neither RAG nor LongLLMLingua yields consistent improvements on LongBenchV2 when the input context length is within 64K tokens. In contrast, DySCO continues to improve performance in this regime. When the context length increases to 128K tokens, RAG-based systems begin to outperform vanilla decoding. However, they still lag behind DySCO on Qwen3-8B.

F Impact of CoT Reasoning and Context Length

DySCO dynamically focuses on key context as generation progresses, making it naturally compatible with CoT, where different reasoning steps rely on different parts of the context. We analyze the interaction between DySCO and CoT across varying context lengths. We evaluate DySCO and vanilla decoding with and without CoT on MRCR and LongBenchV2 at 64K and 128K context lengths, under four decoding settings (direct answer vs. CoT, each with vanilla decoding or DySCO). We report the results for Qwen3-8B.

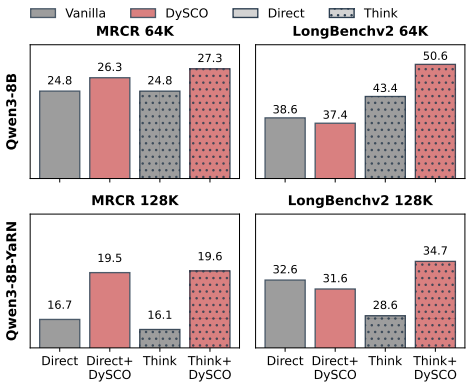

Figure 6: Performance of Qwen3-8B on MRCR and LongBenchV2 at 64K and 128K under four decoding settings: direct answer vs. Think, with vanilla decoding or DySCO. DySCO yields larger gains when combined with CoT.

Interplay between DySCO and CoT reasoning. Figure 6 summarizes the results. Overall, **DySCO yields larger benefits when combined with CoT reasoning**. On MRCR, which primarily evaluates long-context recall, DySCO improves performance in both the direct-answer and CoT settings, with larger gains when CoT is enabled. On LongBenchV2, which emphasizes multi-step reasoning over long contexts, DySCO substantially improves performance in the CoT setting, while causing only minor degradation when CoT is disabled. We hypothesize that this occurs because LongBenchV2 is inherently reasoning-intensive, and disabling CoT limits the model's ability to capitalize on improved attention to relevant context.

Impact of context length. We find that the interaction between CoT reasoning and decoding strategy depends on input length. Under vanilla decoding, CoT outperforms direct answering at 64K context length, but not at longer context lengths (128K). However, DySCO reverses this trend at 128K: **when combined with CoT reasoning, DySCO substantially improves performance at longer input lengths**, where direct-answer decoding is comparatively weaker. By dynamically rescaling attention, DySCO restores effective long-context reasoning at 128K input length.

G PARAMETER SELECTION FOR DYSCO AND UNIATTNS

Our method performs inference-time attention scaling to improve long-context accuracy. Both DYSCO and the baseline UNIATTNS introduce a small number of inference-time hyperparameters. Here, we describe 1) how these parameters are selected, and 2) the robustness of DYSCO to reasonable variations around the chosen defaults.

G.1 CHOOSING PARAMETERS FOR DYSCO

We select hyperparameters using MRCR, a fully synthetic long-context recall benchmark. We choose MRCR for parameter tuning for two main reasons: 1) it is synthetically generated and therefore does not risk data contamination with real-world evaluation benchmarks, and 2) its context length is configurable, allowing controlled analysis across different lengths. Importantly, for each model, we fix a single set of hyperparameters and apply it uniformly across all downstream tasks.

For Qwen models, we distinguish between two context-length settings. For inputs up to 64K tokens, we use the native context window without extrapolation. For 64–128K tokens, we use YaRN-based extrapolation, which globally changes the attention computation. As a result, we select parameters separately for the 0–64K and 64–128K settings, using MRCR-64K and MRCR-128K, respectively. Within each length span, a single parameter configuration is shared across all tasks.

Hyperparameters. We perform a small grid search on MRCR over the following values:

$$p \in \{0.95, 0.975\}, \quad K \in \{4096, 8192\}, \quad \beta \in \{2.0, 2.5, 3.0\}.$$

When multiple configurations yield comparable performance, we prefer smaller β and smaller K, which intervene more conservatively on the attention distribution, and larger p, which spreads the boost over more tokens rather than concentrating it on a few.

The parameter choices used throughout the paper are summarized below:

Model / Length	$\boldsymbol{p}$	$\boldsymbol{K}$	$\boldsymbol{\beta}$
Qwen3-4B (0–64K)	0.975	4096	2.0
Qwen3-4B + YaRN (64–128K)	0.95	8192	2.5
Qwen3-8B (0–64K)	0.975	4096	2.0
Qwen3-8B + YaRN (64–128K)	0.975	8192	2.5
Qwen3-32B (0–64K)	0.975	4096	2.0
Qwen3-32B + YaRN (64–128K)	0.95	8192	2.5
Llama-3.1-8B (0–128K)	0.975	4096	2.0

We observe that models within the same family exhibit consistent optimal configurations. For example, all Qwen models without extrapolation favor $(K = 4096, \beta = 2.0)$, while Qwen models with YaRN consistently favor $(K = 8192, \beta = 2.5)$.

G.2 CHOOSING PARAMETERS FOR UNIATTNS

We select the temperature parameter τ for UNIATTNS using the same MRCR-based protocol. We search over $\tau \in \{0.975, 0.95, 0.9, 0.85\}$. Due to differences in attention sharpness across model families and scales, the optimal τ varies slightly:

$$\tau = 0.95 \text{ (Qwen3-4B)}, \ \tau = 0.975 \text{ (Qwen3-8B)}, \ \tau = 0.975 \text{ (Qwen3-32B)}, \ \tau = 0.9 \text{ (Llama)}.$$

These values are fixed across all tasks and context lengths for each model.

H ROBUSTNESS TO HYPERPARAMETER VARIATIONS

We evaluate the robustness of DYSCO to hyperparameter variations on MRCR-128K and LONGBENCHV2-128K using Qwen3-8B. We vary one hyperparameter at a time while fixing the others to the default configuration.

Table 7: Evaluation of hyperparameter robustness of DySCO with Qwen3-8B. DySCO is robust to modest variations in the rescale strength β, selection constraint K, and selection threshold p.

	p	K	β	**MRCR**	LongBenchV2
Vanilla	–	–	–	16.1	28.6
Default	0.975	8192	2.5	19.6	34.7
Vary p					
	0.95	8192	2.5	18.4	31.7
Varying K					
	0.975	2048	2.5	**19.6**	**36.8**
	0.975	4096	2.5	19.0	32.7
Varying β					
	0.975	8192	3.0	18.5	34.7
	0.975	8192	3.5	18.1	31.7

Table 7 shows that DySCO **consistently outperforms vanilla decoding across all tested settings**. While more aggressive configurations (e.g., $\beta = 3.5$) lead to smaller gains, they still outperform the vanilla baseline.

Importantly, we observe a **strong correspondence between performance on MRCR and LongBenchV2**: hyperparameters that perform well on the synthetic MRCR benchmark also tend to yield strong improvements on LongBenchV2. This trend indicates that the effects of hyperparameter choices are consistent across tasks, and that parameters selected on a synthetic long-context benchmark generalize well to more realistic long-context reasoning tasks.

I Details of the Path Traversal Task

In §2.2, we use Path Traversal to stress-test the basic long-context reasoning capability of LMs. The input to Path Traversal consists of a long list of graph edges $E = \{\langle v_i, v_j \rangle\}$ between nodes. The task is to find a path $\mathcal{T} = (\langle v_{\text{start}}, v_1 \rangle, \ldots, \langle v_{t-1}, v_{\text{target}} \rangle)$ connecting a given start node to a target node. Crucially, the graph is designed in a way that each node along the gold path has *exactly one* outgoing edge. As a result, solving the task reduces to repeatedly identifying the next correct outgoing node and chaining them together. This constraint is *explicitly stated* in the prompt, ensuring a deterministic reasoning procedure. This design eliminates variation due to search order while preserving the core difficulty of long-context reasoning: repeatedly retrieving key information from a long and semantically dense context at multiple decoding steps. By varying the number of nodes, we can control the input length and construct tasks with different context sizes. We provide a full prompt in Prompt I.1.

Specifically, we generate instances with approximately 4, 8, 16, and 32K tokens, corresponding to roughly 250, 500, 1000, and 2000 edges, respectively. LMs are required to find a path of 5 nodes (four edges). As shown in Figure 3 (left), although the task is structurally simple, model performance degrades rapidly as context length increases to 32K, well below the claimed context size, with step-level accuracy (correctness of each $\langle v_{t-1}, v_t \rangle$ along the path) dropping from about 90% to approximately 20%. Path Traversal thus isolates a central challenge of long-context reasoning: the need for repeated dynamic key-context lookup during decoding.

Prompt I.1: Prompt for the Path Traversal Task

[TASK]
In a completely hypothetical world, there are a number of cities. Each city has a one-way connection to only one other city via a specific transit method (bus, train, plane, or ferry). Your task is to provide a route from a city to another city. You should follow the specific instruction provided later and output the route following the format provided in the instruction.

[IMPORTANT NOTES]
- All connections are one-way. If city A is connected to city B, you can travel from A to B, but not the other

way around.
- Because each city is connected to only one other city, so there's only one possible route. To find the route, you can simply start from the starting city, identify the next city it's connected to, and repeat the process until you reach the destination city.
- Please follow the exact format specified below when outputting the route.

[OUTPUT FORMAT]
Please mark the route with <Route>and </Route>tags. The route should be in the following format, where one line is one step of the route:
<Route>
From <CITY_NAME>, take a <TRANSIT_METHOD>to <CITY_NAME>.
...
From <CITY_NAME>, take a <TRANSIT_METHOD>to <CITY_NAME>.
</Route>

[EXAMPLE]
In a hypothetical world, there are a number of cities. Each city has a one-way connection to only one other city via a specific transit method. The details of the cities are as follows:

Fort Worth is a lively city. You can travel from Fort Worth to Manchester by ferry.
Leeds is a lively city. You can travel from Leeds to London by bus.
Manchester is a lively city. You can travel from Manchester to Indianapolis by plane.
Houston is a lively city. You can travel from Houston to London by ferry.
Charlotte is a lively city. You can travel from Charlotte to Charlotte by bus.
London is a lively city. You can travel from London to San Antonio by train.
San Antonio is a lively city. You can travel from San Antonio to Kitchener by train.
Seattle is a lively city. You can travel from Seattle to London by train.
Indianapolis is a lively city. You can travel from Indianapolis to Houston by ferry.

Now find the route from Manchester to Kitchener based on the information above.

<Route>
From Manchester, take a plane to Indianapolis.
From Indianapolis, take a ferry to Houston.
From Houston, take a ferry to London.
From London, take a train to San Antonio.
From San Antonio, take a train to Kitchener.
</Route>

[PROBLEM]
`problem context`
Now find the route from `src city` to `dst city` based on the information above. Some reminders:
- All connections are one-way. You can solve the problem by iteratively finding the next city to travel to until you reach the destination city.
- Follow the specific format for the route output. Mark the route with <Route>and </Route>tags.

J Robustness to the Portion of Selected QRHead Retained

In Table 3, we already show that replacing QRHead entirely with the same number of *randomly* selected heads (w/ RandomHead) still improves over vanilla decoding in several settings, though it consistently underperforms DySCO with the full QRHead set. Here, we interpolate between these two settings to study a more fine-grained robustness question: what happens if only a *fraction* of the selected QRHead is correctly identified?

Concretely, starting from the 16 QRHead used throughout the paper, we retain a fraction (25%, 50%, or 75%) of the originally identified heads for the aggregation stage and replace the remaining heads with randomly selected heads, keeping the total number of aggregation heads fixed at 16. All other components of DySCO (selection and rescaling) are kept unchanged. Table 8 reports results with Qwen3-8B on Path Traversal (16K), MRCR (64K), and LongBenchV2 (64K).

As shown in Table 8, DySCO's gains remain robust and substantial under moderate perturbation of the head set (retaining 50–75% of the identified QRHead), while performance degrades as the

Table 8: Effect of retaining only a fraction of the originally identified QRHEAD set, with the remainder replaced by randomly selected heads, on DYSCO's performance with Qwen3-8B.

	PATH TRAVERSAL 16K	MRCR 64K	LONGBENCHV2 64K
Vanilla	19	24.8	43.4
25% QRHEAD + 75% Rand	26	25.8	45.6
50% QRHEAD + 50% Rand	28	26.4	47.6
75% QRHEAD + 25% Rand	28	25.8	48.1
100% QRHEAD (DYSCO)	**33**	**27.3**	**50.6**

head set is corrupted more heavily (retaining only 25%). Notably, even in this heavily corrupted setting, DYSCO still improves over vanilla decoding on all three datasets. This suggests that DYSCO does not rely on a perfectly identified head set to be effective, and degrades gracefully rather than catastrophically as head-identification quality decreases. We also note that prior work on QRHEAD identification shows that selecting top retrieval heads from disjoint data subsets yields highly overlapping (over 75% intersection) head sets, so in practice we do not expect the selected QRHEAD set to be heavily corrupted.

K GENERALIZATION TO MIXTURE-OF-EXPERTS MODELS

The experiments in the main paper focus on dense transformer models. To test whether QRHEAD-guided dynamic attention scaling generalizes to sparse architectures, we evaluate DYSCO on Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024a), a mixture-of-experts (MoE) model.

Head detection. Importantly, we do *not* change the head detection pipeline for MoE models. We use the same held-out 256 NQ (Kwiatkowski et al., 2019) examples used for dense models to identify QRHEAD, and directly apply the detected heads zero-shot to downstream tasks, without any MoE-specific adaptation.

Existence of QRHEAD in Mixtral. We first verify that the identified heads remain informative retrieval signals for Mixtral by evaluating on the BEIR re-ranking benchmark (Thakur et al., 2021), following the evaluation protocol of Zhang et al. (2025c). Using QRHEAD improves BEIR re-ranking performance from 44.4 (using all heads) to 45.4, confirming that query-focused retrieval heads exist and remain useful in this sparse MoE architecture.

Long-context reasoning results. We further evaluate DYSCO on Mixtral (which natively supports a 32K context window, extended to 64K via YaRN) on MRCR and LONGBENCHV2 at 64K context length. Table 9 reports the results.

Table 9: Results on Mixtral-8x7B-Instruct-v0.1 (with YaRN) at 64K context length. DYSCO improves over both vanilla decoding and UNIATTNS on this sparse MoE architecture.

	MRCR 64K	LONGBENCHV2 (CoT) 64K	LONGBENCHV2 64K
Vanilla	17.2	23.5	33.7
+ UNIATTNS	18.0	27.1	33.7
+ DYSCO	**18.7**	**32.5**	**36.8**

As shown in Table 9, DYSCO consistently outperforms both vanilla decoding and UNIATTNS on Mixtral, indicating that the benefits of QRHEAD-guided dynamic attention scaling are not specific to dense transformer architectures, and that QRHEAD behavior generalizes to sparse MoE models without any changes to head detection.

L Comparison with Test-Time Scaling

DySCO trades additional decoding-time computation for improved long-context reasoning accuracy. To contextualize this tradeoff, we compare DySCO against a standard test-time scaling baseline, self-consistency (Wang et al., 2023), where multiple samples are drawn and a majority vote is taken over the final answers. Table 10 reports results with Qwen3-8B (Think) on LongBenchV2 (64K and 128K) and Clipper.

Table 10: Comparison between DySCO and self-consistency (majority voting over 4 or 8 sampled generations) with Qwen3-8B (Think). All methods use YaRN at 128K (LongBenchV2 128K and Clipper).

	LongBenchV2 64K	LongBenchV2 128K	Clipper
Vanilla	43.4	28.6	45.0
Majority Vote (4 samples)	45.8	29.6	45.5
Majority Vote (8 samples)	47.6	32.7	46.5
DySCO	**50.6**	**34.7**	**48.5**

As shown in Table 10, DySCO consistently outperforms self-consistency, even with as many as 8 samples. Importantly, DySCO incurs additional decoding-time latency (Appendix D) but does not require multiple parallel generations: self-consistency can incur substantial additional latency when samples are drawn sequentially, or substantial memory overhead when generated in parallel. These results suggest that DySCO is an effective way to trade additional decoding-time compute for improved long-context reasoning performance, compared to standard parallel (multi-sample) test-time scaling.

M Qualitative Examples

We present qualitative examples illustrating how DySCO improves long-context reasoning in LMs.

In Figure 7, we show a Path Traversal example where vanilla decoding produces an incorrect output, while DySCO yields the correct result. In Figure 8, we visualize the averaged attention patterns at the first error step, comparing QRHead with all attention heads. The results highlight the robustness of QRHead in consistently attending to relevant context for retrieval.

We further present two failure cases of vanilla decoding on MRCR (Figures 9 and 10). DySCO fully corrects the first and partially corrects the second.

Path Traversal: Route Hallucination

Context (16K tokens, graph description)

Yaroslavl is a lively city. You can travel from Yaroslavl to Hayward by train. AllHead #1
...
Kazan is a lively city. You can travel from Kazan to Sarnia by ferry. AllHead #2
...
Newmarket is a lively city. You can travel from Newmarket to Thornton by ferry. QRHead #2
Thornton is a lively city. You can travel from Thornton to Leatherhead by ferry. ✓ needle at the first error QRHead #1
...
Bordon is a lively city. You can travel from Bordon to Warsaw by ferry. QRHead #3
...
Halesowen is a lively city. You can travel from Halesowen to Spokane by bus. AllHead #3
...
User: Find the route from **Herne** to **Leatherhead**. (task)

Vanilla (accuracy: 0.0)

From Herne, take a plane to Garden Grove.
From Garden Grove, take a bus to Newmarket.
From Newmarket, take a ferry to Thornton.
From Thornton, take a ferry to Bordon. (✗ wrong destination) ← error starts
From Bordon, take a ferry to Gloucester.
From Gloucester, take a train to Reno.
... (never reaches Leatherhead) (✗ path drift)

DySCO (ours) (accuracy: 1.0)

From Herne, take a plane to Garden Grove.
From Garden Grove, take a bus to Newmarket.
From Newmarket, take a ferry to Thornton.
From Thornton, take a ferry to Leatherhead. (✓ correct destination)

Figure 7: Path Traversal example (Herne → Leatherhead, 16K context). The context box shows the top-3 attended lines at the first error step under QRHeads (green) and all heads (gray): QRHeads rank the needle edge (Thornton → Leatherhead) first, while all heads attend to irrelevant cities. Both vanilla and DySCO decoding follow the first three steps correctly; at step 4, vanilla decoding takes the wrong edge (Thornton → Bordon) and drifts indefinitely (accuracy 0.0), whereas DySCO—guided by QRHead attention—retrieves the correct edge and terminates on target (accuracy 1.0).

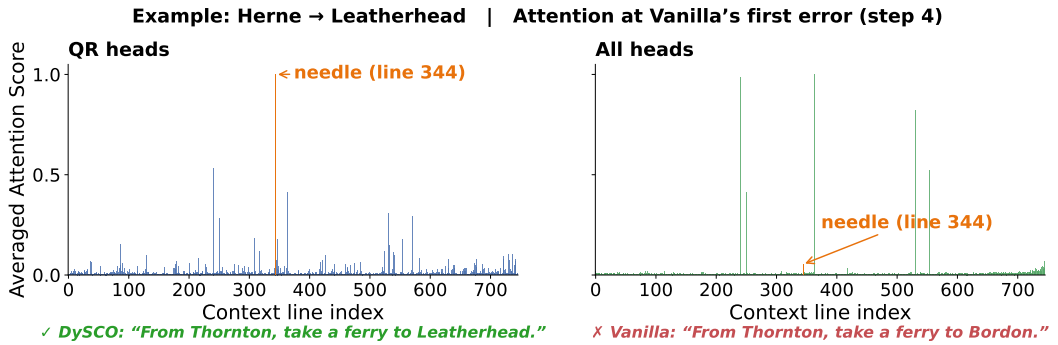

Figure 8: Attention distribution at the first error step (step 4) of the Herne → Leatherhead PATH TRAVERSAL example, where the model must generate "From Thornton, take a ferry to Leatherhead." **Left (QRHeads):** attention is highly concentrated on the needle line describing the Thornton → Leatherhead connection, which ranks first; DySCO leverages this signal and generates the correct step. **Right (all heads):** the average attention across all heads is more dispersed and the needle line's rank drops significantly, causing vanilla decoding to misidentify the next city and generate "From Thornton, take a ferry to Bordon" instead. This illustrates that QRHEADs inherently possess stronger retrieval signal focus *without* any rescaling intervention, and DySCO amplifies these signals to guide generation correctly where vanilla fails.

MRCR Failure Mode 1: Wrong Turn Retrieval

Context (242 turns)

User: Write a short essay about individuals.
Assistant: Individual Responsibility in Modern Society… (1st essay)

⋮ other turns

User: Write a short essay about individuals.
Assistant: The Strength of the Individual… (2nd essay)

⋮ other turns

User: Write a short essay about individuals.
Assistant: The Power and Complexity of Individuals… (3rd essay) ← vanilla retrieves this

⋮ other turns

User: Write a short essay about individuals.
Assistant: **Title: The Unique Impact of Individuals on Society** ✓ **reference (4th essay)**
In the tapestry of human civilization, individuals act as the distinct threads…

⋮ other turns

User: Prepend `[X]` to the **4th** essay about **individuals**. (task)

Vanilla (match ratio: 0.048)

<think>

… the fourth essay was titled "The Power and Complexity of Individuals" (✗ wrong turn identified) …

The Power and Complexity of Individuals (✗ wrong essay title)
In the tapestry of human society (✗ civilization), individuals are the threads that weave together the intricate patterns of communities, cultures, and nations. Each person, unique in their own right, contributes distinct perspectives, experiences, and talents that build the dynamic structure of our world…

DySCO (ours) (match ratio: 1.000)

<think>

… The fourth essay… was about the unique impact of individuals on society (✓ correct turn identified) …

Title: The Unique Impact of Individuals on Society (✓ correct essay retrieved)
In the tapestry of human civilization, individuals act as the distinct threads that, when woven together, create the intricate and colorful fabric of society. These unique beings, each with their own experiences, perspectives, and abilities, contribute to the collective advancement and evolution of the human race…

Figure 9: Failure Mode 1 on MRCR: wrong turn retrieval (64K context, 242 turns). The context embeds four essays about "individuals"; vanilla decoding mistakes the 3rd essay for the 4th (match ratio 0.048), while DySCO correctly identifies and retrieves the target turn (match ratio 1.000).

MRCR Failure Mode 2: Imprecise Copying

Context (156 turns)

User: Write a short scene in a play about application.
Assistant: The Negotiation... (1st scene)

: other turns
User: Write a short scene in a play about application.
Assistant: The Presentation... (2nd scene)

: other turns
User: Write a short scene in a play about application.
Assistant: The Follow-Up... (3rd scene)

: other turns
User: Write a short scene in a play about application.
Assistant: **Title: The Interview** ✓ **reference (4th scene)**

...
EMMA: It would have to be seeing my ideas come to life... diverse perspectives.
MR. JENKINS: That's one of the crucial things—teamwork...
EMMA: Certainly! I'm proficient in Google Analytics...
MR. JENKINS: What's a challenge you've faced?
EMMA: During a campaign, our engagement metrics...
MR. JENKINS: Problem-solving and adaptability... Before concluding, is there anything you'd like to ask?
EMMA: What's the most rewarding part about working here?
...

: other turns
User: Prepend `[X]` to the **4th** scene in a play about **application**. (task)

Vanilla (match ratio: 0.542)

<think>
...The 4th scene... was titled "The Interview" (✓ correct turn identified) with characters MR. JENKINS and EMMA...

Title: The Interview
...
EMMA: It would have to be seeing my ideas come to life... diverse perspectives.
MR. JENKINS: Problem-solving... What's a challenge? (✗ wrong order) ← error starts
EMMA: During a campaign, our engagement metrics... (✗ wrong order)
MR. JENKINS: That's one of the crucial things—teamwork...
EMMA: Certainly! I'm proficient in Google Analytics...
MR. JENKINS: What's a challenge? (✗ duplicated)
EMMA: During a campaign... (✗ duplicated)
MR. JENKINS: teamwork... (✗ duplicated)
EMMA: Certainly! Google Analytics... (✗ duplicated)
MR. JENKINS: Excellent... Before concluding, is there anything you'd like to ask?
EMMA: Yes... What's the most rewarding part about working here?
...

DySCO (ours) (match ratio: 0.750)

<think>
...there was a scene titled "The Interview" (✓ correct turn identified) where Emily applies for a job...

Title: The Interview
...
EMMA: It would have to be seeing my ideas come to life... diverse perspectives.
MR. JENKINS: Problem-solving... What's a challenge? (✗ wrong order)
EMMA: During a campaign, our engagement metrics... (✗ wrong order)
MR. JENKINS: That's one of the crucial things—teamwork...
EMMA: Certainly! I'm proficient in Google Analytics...
MR. JENKINS: Excellent... Before concluding, is there anything you'd like to ask?
EMMA: Yes... What's the most rewarding part about working here? (✓ no duplication)
...

Figure 10: Failure Mode 2 on MRCR: imprecise copying (64K context, 156 turns). Both vanilla and DySCO retrieve the correct turn (Title: The Interview) but misorder the dialogue. Vanilla further corrupts the output by duplicating a full block of dialogue (match ratio 0.542). DySCO avoids the repetition, reproducing the content more faithfully (match ratio 0.750).