LDIS: EFFICIENT LONG-CONTEXT TRAINING DATA SYNTHESIS VIA INTERLEAVED STUDY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context modeling enables language models to process complete documents, code repositories, and extended interaction histories. However, natural text provides sparse supervision for long-range dependencies, while recent hybrid models further face additional constraints from the finite memory capacity of the recurrent blocks. We propose *Long-context training Data synthesis via Interleaved Study* (LDIS). A teacher generates question–answer (QA) pairs from growing document prefixes, which we then interleave with source text at the chunk boundaries. This changes the training prefix distribution across the sequence, adding targets that encourage finite-state blocks to retain and use earlier information. We normalize document and answer losses separately to prevent document tokens from diluting answer supervision. Under matched student training token budgets, LDIS improves long-context performance across model sizes and teacher sources when adapting full attention to sliding-window attention and extending context length, while largely maintaining general capability. Ablations support interleaving and the loss design. We also explore self-improvement without an external teacher, using an internally trained model to supervise a new student initialized from the same base checkpoint.

1 INTRODUCTION

Processing extremely long contexts, e.g., complete documents, code repositories, extended interaction histories, is inarguably one of the core capabilities of large language models. Standard transformers with full attention (FA) process long contexts by directly accessing preceding tokens, with computational and memory costs growing with the context length. The state-space models (SSMs) and sliding-window attention (SWA) reduce these costs through finite-size recurrent memory blocks or local attention (Gu & Dao, 2024; Dao & Gu, 2024; Beltagy et al., 2020), which motivates hybrid models combining recurrent memory with full or sparse attention to support efficient long-context modeling (De et al., 2024; NVIDIA, 2025b; Kimi Team, 2025; NVIDIA, 2025a; MiniCPM Team et al., 2026; Qwen Team, 2026).

A common approach to long-context training is to continually pretrain short-context base models on long-context corpora (Xiong et al., 2024; Fu et al., 2024; Gao et al., 2025b). However, longer sequences alone do not necessarily ensure stronger supervision for long-range dependencies. Most tokens in natural text are predictable from local contexts. Predictions that truly require earlier evidence are sparse. Hybrid models face an additional challenge because their SSM blocks must retain long-range information within a finite memory capacity. In this work, we investigate how to synthesize long-context training data that jointly addresses these two challenges. Our core idea is to change the prefix distribution *across* the long sequence to (i) bias towards more long-context dependencies; (ii) explicitly press finite memory blocks to maintain useful information.

We propose *Long-context training Data synthesis via Interleaved Study* (LDIS), motivated by Cartridges' self-supervised QA for KV cache compression (Eyuboglu et al., 2025). In LDIS, a teacher generates QA pairs from continually growing document prefixes, which are interleaved at their corresponding chunk boundaries. This changes the *training prefix distribution* by introducing questions about earlier content at multiple document positions. The answer targets encourage finite-state blocks to retain useful information through successive state updates. We separately normalize document and answer losses to prevent answer supervision from being diluted by the document tokens.

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.

We evaluate LDIS in two hybrid model settings: (i) fine-tuning adaptation to replace most FA layers in Qwen3.5 (Qwen Team, 2026) with SWA; (ii) extending Nemotron-H-4B-Base (NVIDIA, 2025b) from 8K context length to 64K. We compare ProLong (Gao et al., 2025b) with LDIS-augmented data across different model sizes and multiple teacher sources. LDIS raises the average long-context score (see Section 3) from 34.9 to 39.7 for Qwen3.5-4B-SWA after 256K adaptation, and from 31.7 to 34.8 for Nemotron-H-4B after extension and instruction tuning, relative to ProLong. Various ablations support our interleaving method and loss design. We also explore self-improvement without an external teacher: supervision generated by a ProLong-trained Nemotron model improves the long-context performance in a new student initialized from the same 8K base checkpoint. Our method also largely maintains the overall general capability relative to ProLong.

Related work. Prior work improves long-context training by selecting dependency-rich text (Chen et al., 2025a; Wu et al., 2025), augmenting through distractors or chunk reordering (Gao et al., 2025a; Tian et al., 2025), or building instruction data to teach models to use document information (Cheng et al., 2024; An et al., 2024; Xu et al., 2025). In contrast, we interleave prefix-grounded QAs with the source and normalize document and answer losses separately. We are motivated by Cartridges (Eyuboglu et al., 2025) which compresses KV by using self-generated QA. We refer to Appendix A.

2 Preliminaries

2.1 Efficient Attention Blocks and Hybrid Architectures

For a token sequence $x_{1:L}$, let $\mathbf{q}_t, \mathbf{k}_t \in \mathbb{R}^{d_k}$ and $\mathbf{v}_t \in \mathbb{R}^{d_v}$ denote the projected query, key, and value at position t. Full attention (FA) accesses the entire causal prefix, whereas SSMs compress it into a fixed-size state. In a single-head recurrent-memory view, the state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ produces the readout $\mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t$. Mamba-2 uses a decayed additive write, $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{k}_t \mathbf{v}_t^\top$, while Gated DeltaNet writes a prediction residual, $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \beta_t \mathbf{k}_t (\mathbf{v}_t - (\alpha_t \mathbf{S}_{t-1})^\top \mathbf{k}_t)^\top$ (Dao & Gu, 2024; Yang et al., 2025). Here $\alpha_t \in (0, 1]$ controls decay and $\beta_t \in [0, 1]$ the write step size. Long-range information must survive successive updates of this fixed-size state.

SWA preserves softmax attention within a window of size w (Beltagy et al., 2020), with output $\mathbf{o}_t = \sum_{i \in \mathcal{W}_t} \mathrm{softmax}_{i \in \mathcal{W}_t}(\mathbf{q}_t^\top \mathbf{k}_i / \sqrt{d_k}) \mathbf{v}_i$ over $\mathcal{W}_t = \{i \mid \max(1, t - w + 1) \le i \le t\}$. For fixed hidden dimensions, its per-layer attention cost is $\mathcal{O}(Lw)$ instead of FA's $\mathcal{O}(L^2)$, with $\mathcal{O}(w)$ cache cost. Hybrids such as Griffin (De et al., 2024), Nemotron-H (NVIDIA, 2025b), Kimi Linear (Kimi Team, 2025), MiniCPM-SALA (MiniCPM Team et al., 2026), and Qwen3.5 (Qwen Team, 2026) interleave recurrent layers with full or sparse attention, combining compressed global memory with high-fidelity token interactions. This also motivates supervision that strengthens the retention and propagation of distant information through recurrent memory.

2.2 Interleaved Study

In natural documents, many tokens are predictable from nearby context, while targets requiring distant evidence can be sparse. LDIS supplements these documents with prefix-grounded QAs that provide explicit targets for using earlier information. In recipe names, IS abbreviates Interleaved Study. Our motivation draws on Cartridges (Eyuboglu et al., 2025), which trains a compact, corpus-specific KV cache through *self-study distillation* on synthetic QAs. The shared intuition is to make a bounded representation of document information useful for later queries. We apply this intuition to training shared model parameters across documents, encouraging the fixed-size recurrent states produced during reading to retain useful information through successive updates.

Component 1: Prefix-grounded QA generation and interleaving. A teacher model π_t, which may differ from the student model π_s, generates QAs grounded in successive document prefixes. Given a sampled long-context document $\mathcal{C} = (x_1, \cdots, x_L)$ and a fixed chunk size $\ell \le L$, we first generate a set of prefix-grounded questions $\mathcal{Q}(\mathcal{C}, \pi_t) = \{q_i\}_{i=1}^{\lceil L/\ell \rceil}$ via

$$q_i \sim \pi_t\big(\cdot \,\big|\, (x_1, \cdots, x_{i\ell}) \oplus p_{\mathcal{Q}}\big) \quad \text{for } i = 1, \cdots, \lceil L/\ell \rceil. \tag{2.1}$$

Here $p_{\mathcal{Q}}$ is a carefully designed prompt for question generation, independent of $\mathcal{C}$, and $\oplus$ denotes concatenation. The prompt $p_{\mathcal{Q}}$ encourages questions that require recalling and connecting information

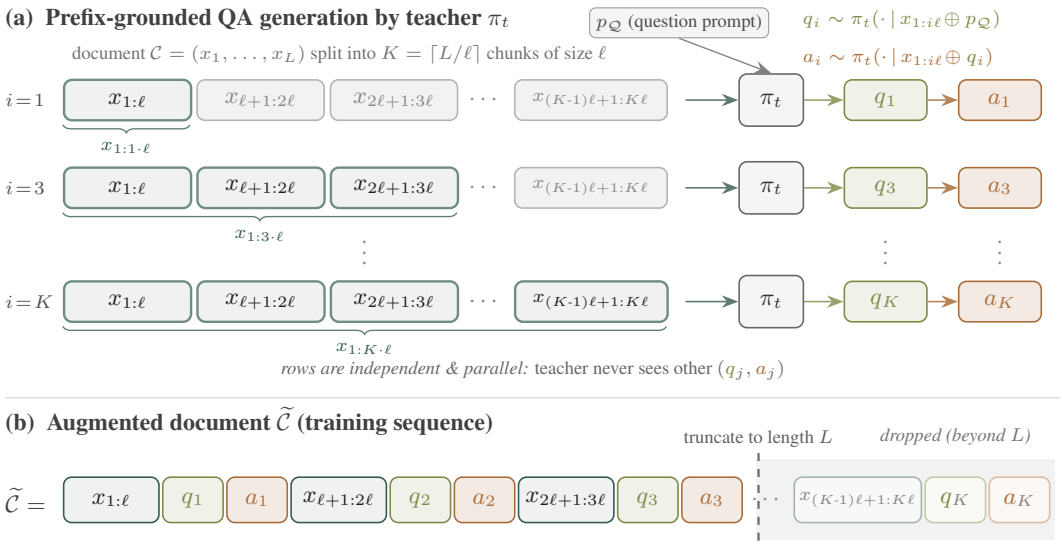


Figure 1: **Interleaved Study data augmentation.** (a) Parallel QA generation. (b) Interleaved training sequence.

from the prefix. Given $\mathcal{Q}(\mathcal{C}, \pi_t)$, we then generate the corresponding answers $\mathcal{A}(\mathcal{C}, \pi_t) = \{a_i\}_{i=1}^{\lceil L/\ell \rceil}$,

$$a_i \sim \pi_t\big(\cdot \,\big|\, (x_1, \cdots, x_{i\ell}) \oplus q_i\big) \quad \text{for } i = 1, \cdots, \lceil L/\ell \rceil. \tag{2.2}$$

In practice, for each single document $\mathcal{C}$ and each QA position $i \in \{1, \cdots, \lceil L/\ell \rceil\}$, we generate multiple QAs and perform a filtering step to ensure the quality of the generated QAs.

The QAs can draw on the entire prefix $(x_1, \cdots, x_{i\ell})$, rather than only its last ℓ tokens. When generating (q_j, a_j), the teacher π_t does not see previous QAs $\{(q_i, a_i)\}_{i<j}$, avoiding conditioning on earlier questions and enabling parallel generation across positions. Finally, we insert the generated QAs into the original context $\mathcal{C}$ to form the augmented training data (see Figure 1 for an illustration of the whole process),

$$\widetilde{\mathcal{C}} := x_{1:\ell} \oplus q_1 \oplus a_1 \oplus \cdots \oplus x_{(\lceil L/\ell \rceil - 1)\ell + 1:\lceil L/\ell \rceil \ell} \oplus q_{\lceil L/\ell \rceil} \oplus a_{\lceil L/\ell \rceil} := (\widetilde{x}_1, \cdots, \widetilde{x}_L),$$

where we truncate the augmented document $\widetilde{\mathcal{C}}$ to the original length L for training efficiency.

Interleaving changes the training distribution of prefixes—the contexts preceding supervised targets—by adding answer targets at multiple document positions. When these targets require distant evidence, they place training pressure on finite-state recurrent layers to retain and use earlier information through successive state updates.

Now for the entire training dataset $\mathcal{D} = \{\mathcal{C}^{(n)}\}_{n=1}^{N}$, we perform the above process on a chosen subset $\mathcal{D}_{\text{studied}} \subset \mathcal{D}$ to obtain the augmented dataset $\widetilde{\mathcal{D}}_{\text{studied}} = \{\widetilde{\mathcal{C}}^{(n)}\}_{\mathcal{C}^{(n)} \in \mathcal{D}_{\text{studied}}}$. We keep the rest $\mathcal{D}_{\text{un}} = \mathcal{D} \setminus \mathcal{D}_{\text{studied}}$ unchanged, and define the final dataset as $\widetilde{\mathcal{D}} = \widetilde{\mathcal{D}}_{\text{studied}} \cup \mathcal{D}_{\text{un}}$. For notational simplicity, for the augmented documents $\widetilde{\mathcal{C}}^{(n)} \in \widetilde{\mathcal{D}}_{\text{studied}}$, we define two mask functions $\mathbf{1}_{t,\text{orig}}^{(n)}$ and $\mathbf{1}_{t,\text{ans}}^{(n)}$ that indicate whether the token $\widetilde{x}_t^{(n)}$ is an original token or an inserted answer token,

$$\mathbf{1}_{t,\text{orig}}^{(n)} := \mathbf{1}\Big\{\widetilde{x}_t^{(n)} \text{ is an original document token}\Big\}, \quad \text{and} \quad \mathbf{1}_{t,\text{ans}}^{(n)} := \mathbf{1}\Big\{\widetilde{x}_t^{(n)} \in \bigcup_{i=1}^{\lceil L/\ell \rceil} a_i^{(n)}\Big\}.$$

For un-augmented documents $\mathcal{C} \in \mathcal{D}_{\text{un}}$, we simply set $\mathbf{1}_{t,\text{orig}} = 1$ for all $t \in \{1, \cdots, L\}$.

Component 2: Separately normalized document and answer losses. Interleaving determines where QA supervision appears; separate normalization controls its weight. A single token average can dilute the comparatively few answer tokens among the document tokens. We therefore average document and answer losses separately and combine them with weight λ,

$$\mathcal{L}_{\text{aug}}(\pi_s; \widetilde{\mathcal{D}}) := \mathcal{L}_{\text{NTP}}(\pi_s; \widetilde{\mathcal{D}}) + \lambda \cdot \mathcal{L}_{\text{QA}}(\pi_s; \widetilde{\mathcal{D}}_{\text{studied}}),$$

where the two losses $\mathcal{L}_{\text{NTP}}$ and $\mathcal{L}_{\text{QA}}$ are defined as follows:

$$\mathcal{L}_{\text{NTP}}(\pi_s; \widetilde{\mathcal{D}}) := \frac{1}{\sum_{n,t=1}^{N,L} \mathbf{1}_{t,\text{orig}}^{(n)}} \left(\sum_{\widetilde{\mathcal{D}}_{\text{studied}}} \sum_{t=1}^{L} -\mathbf{1}_{t,\text{orig}}^{(n)} \log \pi_s\big(\widetilde{x}_t^{(n)} \big| \widetilde{x}_{<t}^{(n)}\big) + \sum_{\mathcal{D}_{\text{un}}} \sum_{t=1}^{L} - \log \pi_s\big(x_t^{(n)} \big| x_{<t}^{(n)}\big) \right),$$

$$\mathcal{L}_{\text{QA}}(\pi_s; \widetilde{\mathcal{D}}_{\text{studied}}) := \frac{1}{\sum_{\mathcal{D}_{\text{studied}}} \sum_{t=1}^{L} \mathbf{1}_{t,\text{ans}}^{(n)}} \cdot \sum_{\widetilde{\mathcal{D}}_{\text{studied}}} \sum_{t=1}^{L} -\mathbf{1}_{t,\text{ans}}^{(n)} \cdot \log \pi_s\big(\widetilde{x}_t^{(n)} \big| \widetilde{x}_{<t}^{(n)}\big). \tag{2.3}$$

Question tokens provide context but receive no direct loss; both supervised terms are computed in one student forward pass. We sometimes also consider a soft answer distillation variant by replacing the QA loss with the KL divergence between the teacher and student answer distributions, i.e. $D_{\text{KL}}(\pi_t(\cdot|\widetilde{x}_{<t}^{(n)})||\pi_s(\cdot|\widetilde{x}_{<t}^{(n)}))$.

2.3 Experiment Setups

2.3.1 Experiment 1: Adapting SSM-FA to SSM-SWA

We adapt the Qwen3.5 series of SSM-FA hybrid models (Qwen Team, 2026) to even more efficient architectures. Specifically, Qwen3.5 models use a $3:1$ interleave of Gated DeltaNet layers and FA layers. Replacing a subset of Qwen3.5's FA layers with SWA yields Qwen3.5-SWA, improving decoding throughput and reducing memory use. This efficiency comes at a cost: SWA cannot directly retrieve tokens outside its local window, and Qwen3.5-SWA suffers a sharp drop in long-context performance. This efficiency–capability comparison is summarized in Table 15 (Appendix H). To recover the long-context capability of Qwen3.5-SWA, our approach is to perform a long-context adaptation fine-tuning stage (Yu et al., 2025; MiniCPM Team et al., 2026; Pan et al., 2026).

To this end, we adopt the ProLong dataset (Gao et al., 2025b), an open-source, long-context training corpus, as our baseline fine-tuning adaptation data. We then compare the long-context capability of Qwen3.5-SWA trained on the original dataset and on LDIS-augmented data (with the new training loss (2.3)). We match training token budgets to compare long-context performance and scaling.

We use Qwen3.5-{0.8, 2, 4}B[1] as student π_s. For the teacher π_t, we ablate (i) the original Qwen3.5-4B, and (ii) a closed source model GPT-5-Nano. The fine-tuning adaptation process is divided into two consecutive stages: a 64K stage and a 256K stage. The 64K stage uses the ProLong@64K split, and the 256K stage truncates the ProLong@512K split into 256K. We set the second stage to a 256K length mainly because the context window of Qwen3.5 is 256K and we include it as the teacher.

2.3.2 Experiment 2: Long-Context Extension of Hybrid Base Model

Our second application concerns the length extension stage in LLM training. We take Nemotron-H-4B-Base-8K[2] (Nemotron-H for short) as the base model. It is a hybrid model interleaving 4 FA layers, 24 Mamba-2 layers, and 24 FFN layers following a specific pattern (NVIDIA, 2025b), without positional embedding (NoPE). The student model π_s is the base model, which has context length 8K and we aim to extend it to 64K via continual pretraining. We use ProLong@64K as the continual pre-training dataset, after which we adopt UltraChat (Ding et al., 2023) for the instruction-tuning stage, and we ablate the continual pre-training data using our augmentation in a token budget controlled manner. For the teacher model π_t, we ablate three models: (i) Nemotron-Nano-4B[3], (ii) GPT-5-Nano, and (iii) the instruction model trained without our data augmentation. For the teacher (iii), *no* external teacher model is used, which genuinely tests whether this data synthesis pipeline can *self-improve* model training. We illustrate this self-improvement loop for teacher (iii) in Figure 6.

2.4 Interleaved Study Pipeline in Practice

Figure 2 summarizes the practical LDIS pipeline; implementation details are provided in Appendix B.

[1] https://huggingface.co/Qwen/Qwen3.5-4B
[2] https://huggingface.co/nvidia/Nemotron-H-4B-Base-8K
[3] https://huggingface.co/blog/nvidia/nemotron-3-nano-4b

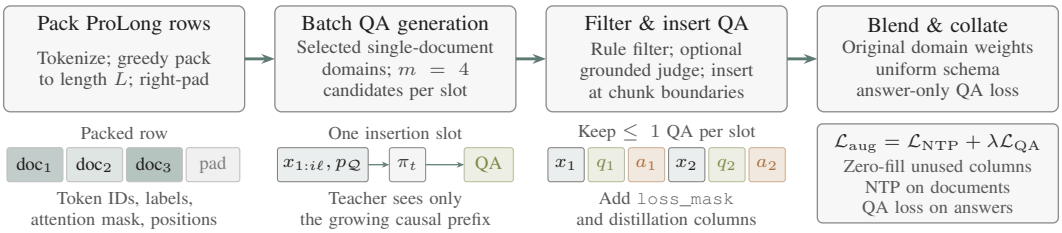


Figure 2: **Practical LDIS pipeline.**

3 Experiment Details and Results

Evaluation Benchmarks. For the long-context evaluation, we use RULER (Hsieh et al., 2024), HELMET (Yen et al., 2025), MRCR-2 (Vodrahalli et al., 2024), LongBench v2 (Bai et al., 2025), and AA-LCR (Artificial Analysis Team, 2025). RULER tests retrieval, multi-hop tracing, and aggregation; HELMET covers application-oriented tasks. MRCR-2 tests co-reference resolution and response retrieval in long dialogues, while LongBench v2 and AA-LCR assess long-context understanding and reasoning. Avg. is the unweighted mean of the final five metric columns (the second RULER column, HELMET, MRCR-2, LongBench v2, and AA-LCR), excluding the first RULER column. For standard capability evaluation, we use MMLU-Redux (Gema et al., 2025), MMLU-Pro (Wang et al., 2024b), GPQA-Diamond (Rein et al., 2024), SuperGPQA (M-A-P Team et al., 2025), IFEval (Zhou et al., 2023), IFBench (Pyatkin et al., 2025), and MultiChallenge (Deshpande et al., 2025).

3.1 Experiment 1: Adapting SSM-FA to SSM-SWA

Our main experiment adapts Qwen3.5-4B to Qwen3.5-4B-SWA. Our training is based on the distributed training framework `AutoModel`[4]. The two-stage training recipe is summarized in Table 1. The first stage restores long-context behavior at 64K, and the second stage continues the adaptation to 256K. Both stages use cross document masking for multiple document rows in the data, following the recipe from Gao et al. (2025b). Specifically, the 256K stage features an implementation of cross-document masking for the Gated DeltaNet under context parallelism. We control the token budget to compare different data recipes.

Table 1: Qwen3.5-4B-SWA recipe.

	Stage 1	**Stage 2**
Context length L	64K	256K
Training budget N	7B	7B
QA chunk size ℓ	4K	8K
QA-loss weight λ	0.5	0.5
Optimizer	AdamW	AdamW
Learning rate η	3e-5 (cosine)	2e-5 (cosine)
Batch size B	4M	8M
Parallelism	DP=32	DP=8, CP=4

In our experiment, we adapt two variants of Qwen3.5-4B-SWA, one with two FA layers kept (Qwen3.5-4B-SWA-2FA), and another with three FA layers kept (Qwen3.5-4B-SWA-3FA). We find that retaining a small number of FA layers helps preserve the model's original long-context behavior while still providing the efficiency benefits of SWA layers. We select the retained FA layers using the layer-placement screen detailed in Appendix G. Based on this ablation study, we keep layers $\{15, 27\}$ for Qwen3.5-4B-SWA-2FA and layers $\{11, 19, 27\}$ for Qwen3.5-4B-SWA-3FA. We set the SWA window size to 4K for both variants in our main experiments.

Main experiment results. Tables 2 and 3 compare ProLong with LDIS data generated by Qwen3.5-4B or GPT-5-Nano under matched token budgets. References include original and unadapted Qwen models and reported SpikingBrain2.0 (Pan et al., 2026), MiniCPM-SALA (MiniCPM Team et al., 2026), and Llama-3-8B-ProLong (Gao et al., 2025b) results. Continued training partially restores performance after replacing FA with SWA; LDIS further improves the five-benchmark average with either teacher in every reported setting. After 256K training, Qwen supervision gives the best 64K-evaluation averages of 43.4 (2FA) and 48.3 (3FA), exceeding ProLong by 3.3 and 3.6 points. GPT supervision leads at 256K evaluation, raising the average from 32.0 to 34.2 (2FA) and 34.9 to 39.7 (3FA). Here, Qwen supervision is stronger on LongBench v2 and AA-LCR, while GPT is stronger on RULER, HELMET, and MRCR-2. Three FA layers yield higher averages across all recipes. However, the GPT/2FA recipe's 64K-evaluation average falls from 42.9 to 41.7 after 256K training, while still exceeding ProLong.

[4]https://github.com/nvidia-nemo/automodel

Table 2: 64K evaluation of Qwen3.5-4B-SWA. In Tables 2–3, **bold**/underline mark the best/second-best trained scores within each 2FA/3FA group across 64/256K stages; displayed ties share marks. Green highlights the best Avg., the mean of the final five metrics (excluding the first RULER column).

Model / Recipe	Stage	RULER @32K	RULER @64K	HELMET @64K	MRCR-2 @64K	LB-V2-S (64K)	AA-LCR	Avg.
References & Related Models								
Qwen3.5-4B	N/A	96.3	95.9	57.2	81.0	48.9	38.0	64.2
SpB2.0-5B	N/A	50.9	–	–	–	–	–	–
MiniCPM-SALA-9B	N/A	–	92.7	–	22.3	–	–	–
Llama-3-8B-ProLong	N/A	89.3	83.2	50.3	–	31.7	–	–
Qwen3.5-4B-SWA-2FA								
Before adapt.	N/A	33.0	22.1	9.5	27.5	21.9	7.2	17.6
ProLong	64K	75.8	60.7	26.8	32.8	33.6	20.5	34.9
	256K	88.7	76.9	**36.0**	32.2	31.3	24.0	40.1
ProLong-IS-GPT	64K	89.9	80.1	34.0	33.8	39.6	26.8	42.9
	256K	90.1	**81.3**	34.8	33.7	34.4	24.2	41.7
ProLong-IS-Qwen	64K	89.1	74.8	34.5	**34.3**	40.0	24.0	41.5
	256K	**90.4**	79.7	33.0	31.9	**43.6**	**29.0**	**43.4**
Qwen3.5-4B-SWA-3FA								
Before adapt.	N/A	68.0	57.9	20.4	31.0	26.4	13.5	29.8
ProLong	64K	89.7	77.2	32.9	40.5	41.4	22.8	43.0
	256K	88.6	81.0	36.3	41.1	39.2	25.8	44.7
ProLong-IS-GPT	64K	94.9	**90.3**	38.2	42.2	39.4	20.8	46.2
	256K	95.1	86.7	37.7	**46.1**	42.4	20.8	46.7
ProLong-IS-Qwen	64K	94.8	85.9	**41.0**	42.3	48.9	19.5	47.5
	256K	**95.7**	87.5	36.7	41.1	**49.9**	**26.5**	**48.3**

Table 3: 256K evaluation of Qwen3.5-4B-SWA. Avg. excludes RULER@128K; ranking conventions follow Table 2.

Model / Recipe	Stage	RULER @128K	RULER @256K	HELMET @128K	MRCR-2 @256K	LB-V2-M (256K)	AA-LCR	Avg.
References & Related Models								
Qwen3.5-4B	N/A	95.6	94.6	53.3	77.6	35.8	38.0	59.9
MiniCPM-SALA-9B	N/A	89.4	–	–	–	–	–	–
Llama-3-8B-ProLong	N/A	81.6	–	49.4	–	29.3	–	–
Qwen3.5-4B-SWA-2FA								
Before adapt.	N/A	16.4	13.5	8.0	19.9	23.7	7.2	14.5
ProLong	256K	64.9	52.3	**30.8**	26.1	27.0	24.0	32.0
ProLong-IS-GPT	256K	**72.1**	**60.3**	**30.8**	**26.2**	29.3	24.2	**34.2**
ProLong-IS-Qwen	256K	64.9	51.7	29.4	25.9	**33.5**	**29.0**	33.9
Qwen3.5-4B-SWA-3FA								
Before adapt.	N/A	40.5	28.3	15.4	24.5	20.0	13.5	20.3
ProLong	256K	69.4	57.1	33.0	36.3	22.3	25.8	34.9
ProLong-IS-GPT	256K	**81.4**	**67.5**	**35.3**	**40.0**	34.9	20.8	**39.7**
ProLong-IS-Qwen	256K	77.7	62.7	33.3	33.9	**36.7**	**26.5**	38.6

General capability results. After 256K adaptation, the overall standard-capability averages are 48.1, 45.7, and 47.3 for ProLong, ProLong-IS-GPT, and ProLong-IS-Qwen with the 2FA variant, respectively. The corresponding averages for the 3FA variant are 46.7, 47.0, and 47.2. Full benchmark results are reported in Table 10 (Appendix C).

Analysis on RULER at different context lengths. Figure 3 shows that differences between data recipes grow near the training horizon. At the 64K-stage checkpoint, GPT-generated QA improves RULER@64K over ProLong by 19.4 points for 2FA and 13.1 for 3FA. After 256K adaptation, the recipes remain close through 16K but separate beyond 64K. At RULER@256K, the GPT recipe improves 52.3 to 60.3 for 2FA and 57.1 to 67.5 for 3FA. Qwen-generated QA is more architecture-dependent: its 2FA score is comparable to ProLong (51.7 versus 52.3), while its 3FA score improves from 57.1 to 62.7.

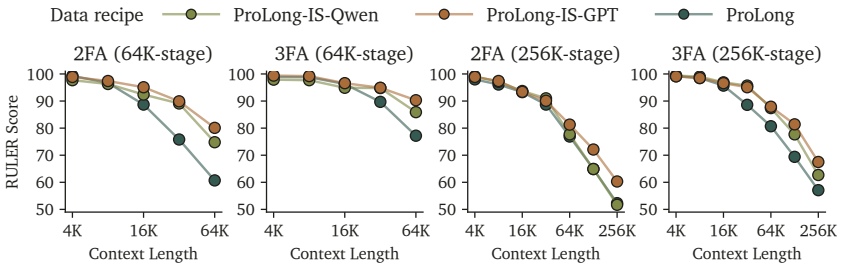


Figure 3: RULER across context lengths after the 64K (left two panels) and 256K (right two panels) adaptation stages. Within each stage, results are shown for the 2FA and 3FA variants.

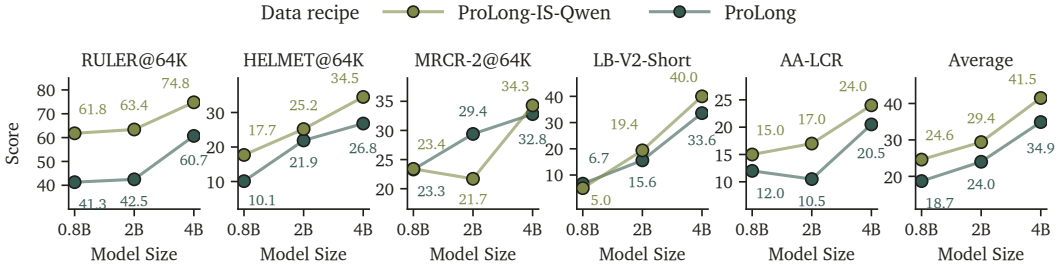


Figure 4: Model-size scaling of Qwen3.5-SWA-2FA after 64K adaptation.

Model size scaling. We also adapt Qwen3.5-{0.8B, 2B}-SWA-2FA using Qwen3.5-4B as teacher and the same FA-layer selection procedure (Appendix G). Both undergo the 64K stage, with 1.5B training tokens for 0.8B and 4B tokens for 2B. Figure 4 compares all three model sizes on the same 64K benchmarks. We can see that the augmented recipe ProLong-IS-Qwen consistently improves the average benchmark performance over ProLong across the scaling curve.

3.2 Experiment 2: Long-Context Extension of Hybrid Model

In this experiment, we extend the context length of the hybrid model Nemotron-H-4B-Base from 8K to 64K, using `AutoModel` as our distributed training framework. The recipe (Table 4) combines 64K continual pretraining with a following SFT stage for instruction following, enabling proper evaluation on certain long-context benchmarks. Both stages use cross-document/instance masking for packed data, as in Experiment 1. We compare CPT data recipes under matched token budgets.

Table 4: Nemotron-H-4B recipe.

	CPT Stage	**SFT Stage**
Context length L	64K	64K
Training budget N	7B	1B
QA chunk size ℓ	4K	N/A
QA-loss weight λ	0.5	N/A
Optimizer	AdamW	AdamW
Learning rate η	3e-5 (cosine)	2e-5 (cosine)
Batch size B	4M	1M
Parallelism	DP=32	DP=8

Main experiment results. Table 5 compares ProLong with LDIS supervision from GPT-5-Nano or Nemotron-Nano-4B after 64K continual pretraining and SFT, under matched training token budgets.

Table 5: Long-context benchmark evaluation (64K) for Nemotron-H-4B length extension.

Model / Recipe	**Stage**	RULER @32K	RULER @64K	HELM ET64K	MRCR-2 @64K	LB-V2 -S(64K)	AA -LCR	Avg.
Nemotron-H-4B-64K								
ProLong	64K CPT + SFT	79.2	71.4	32.5	19.7	31.7	3.0	31.7
ProLong-IS-GPT	64K CPT + SFT	**88.3**	**79.1**	**33.1**	21.4	29.4	**11.0**	**34.8**
ProLong-IS-Nemotron	64K CPT + SFT	81.9	76.7	32.8	**21.6**	**33.3**	5.0	33.9

Both LDIS variants improve the five-benchmark average from ProLong's 31.7 to 34.8 (GPT) and 33.9 (Nemotron), gains of 3.1 and 2.2 points. GPT supervision leads on RULER, HELMET, and AA-LCR;

Nemotron leads on MRCR-2 and LongBench v2 Short. The gains are not uniform: on LongBench v2 Short, the GPT recipe scores 29.4 versus ProLong's 31.7, whereas the Nemotron recipe reaches 33.3.

General capability results. For Nemotron-H-4B-64K, the overall standard-capability averages are 30.7, 31.4, and 29.6 for ProLong, ProLong-IS-GPT, and ProLong-IS-Nemotron, respectively. Full benchmark results are reported in Table 11 (Appendix C).

Analysis on RULER at different context lengths. Figure 5 presents RULER results across five evaluation lengths, with the official Nemotron-H-4B-Instruct-128K[5] instruction-following model as a reference (NVIDIA, 2025b). ProLong-IS-GPT yields the strongest curve at every length, reaching 88.3 at 32K and 79.1 at 64K, improvements of 9.1 and 7.7 points over ProLong, respectively. The gain from ProLong-IS-Nemotron compared to ProLong becomes more pronounced near the 64K training horizon: it reaches 81.9 at 32K and 76.7 at 64K, improving over original ProLong by 2.7 and 5.3 points. At 64K context length, the two LDIS variants also exceed the 76.0 score of Nemotron-H-4B-Instruct-128K by 3.1 and 0.7 points, respectively, showing competitiveness of our methods.

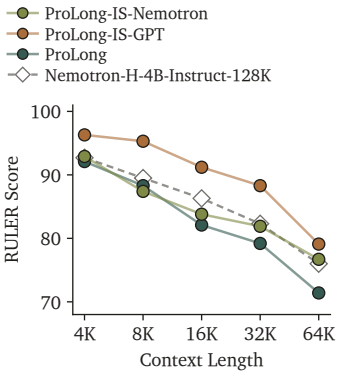


Figure 5: RULER score by length.

Self-improvement Loop. We next ask: *Can LDIS improve long-context capability by bootstrapping existing data without an external teacher?* To test this, we use Nemotron-H-4B-64K trained on original ProLong data as the teacher, instead of Nemotron-Nano-4B or GPT-5-Nano.

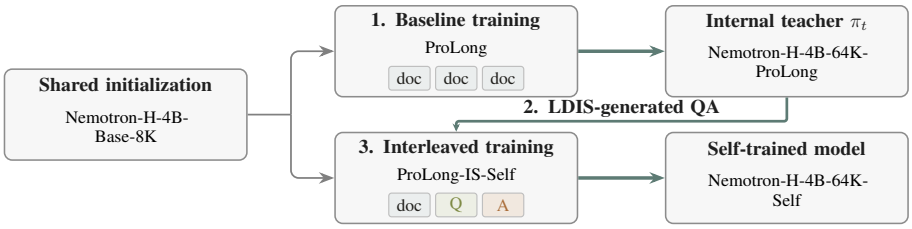


Figure 6: **Self-improvement with LDIS.** Both training branches share the same base model.

Figure 6 illustrates the protocol. We apply LDIS to the original ProLong dataset using the ProLong-trained model as teacher π_t, producing ProLong-IS-Self. We then train a new student, Nemotron-H-4B-64K-Self, on this data. Both models start from the same 8K base checkpoint, so the student does not inherit the teacher's trained weights. Table 6 compares the teacher and student on RULER across five context lengths. The student matches its teacher at 4K and improves at 8K–64K, with the largest gain of 4.3 points at 64K (71.4 to 75.7). These results support self-improvement through interleaved QA supervision in a single round, without an external teacher.

Table 6: RULER, Nemotron-H-4B-64K-ProLong/Self.

Model	4K	8K	16K	32K	64K
Nemotron-H-4B-64K-ProLong	92.1	88.3	82.1	79.2	71.4
Nemotron-H-4B-64K-Self	92.1	88.8	83.2	81.0	75.7
Gain	+0.0	+0.5	+1.1	+1.8	+4.3

3.3 Ablation Studies

We ablate the placement of QA supervision, its loss normalization, and the learning rate for LDIS. All comparisons use Qwen3.5-4B-SWA-2FA after the 64K adaptation stage, with Qwen3.5-4B as teacher for the augmented variants. We follow the training recipe in Table 1 unless varied below.

Ablation 1: QA insertion position. A natural alternative to interleaving is to place all teacher-generated QA pairs at the end of each document. Both arrangements can require information from distant text, but they expose the student to different distributions of prefixes at answer positions.

[5]https://huggingface.co/nvidia/Nemotron-H-4B-Instruct-128K

Interleaving spreads these targets across multiple token spans, encouraging recurrent layers to retain earlier information for later use. We compare ProLong-IS-Qwen with the end-only ProLong-IS-Qwen-End under the same training recipe.

Table 7 shows that the benefit of interleaving QA pairs rather than placing them all at the end grows with evaluation length. These two variants perform similarly at 4K (97.7 vs. 97.8), but interleaving improves over end-only placement by 2.1, 2.9, and 10.3 points at 8K, 16K, and 32K, respectively. These results support distributing supervision across the document rather than concentrating it at its end.

Table 7: RULER scores by QA insertion position.

Model	4K	8K	16K	32K
ProLong	99.3	96.7	88.7	75.8
ProLong-IS-Qwen-End	97.8	94.2	89.5	78.8
ProLong-IS-Qwen	97.7	96.3	92.4	89.1
Gain (v.s. End)	-0.1	+2.1	+2.9	+10.3

Ablation 2: Answer-loss normalization. We compare separate answer normalization with a single average over document and answer tokens, giving them equal per-token weight:

$$\mathcal{L}_{\text{aug}}^{\text{global}} := -\frac{\sum_{n=1}^{N}\sum_{t=1}^{L}(\mathbf{1}_{t,\text{orig}}^{(n)} + \mathbf{1}_{t,\text{ans}}^{(n)})\log \pi_s(\widetilde{x}_t^{(n)} \mid \widetilde{x}_{<t}^{(n)})}{\sum_{n=1}^{N}\sum_{t=1}^{L}(\mathbf{1}_{t,\text{orig}}^{(n)} + \mathbf{1}_{t,\text{ans}}^{(n)})}. \tag{3.1}$$

For unaugmented documents, $\widetilde{x}_t^{(n)} = x_t^{(n)}$ and $\mathbf{1}_{t,\text{ans}}^{(n)} = 0$. Question tokens remain visible in the input but receive no direct supervision.

Table 8 separates the contributions of data augmentation and answer weighting. Adding interleaved QA with global averaging improves over ProLong by 0.4 and 4.1 points at 16K and 32K. Separately normalizing the answer loss yields further gains of 3.3 and 9.2 points at these lengths. ProLong remains strongest at 4K and 8K, so the gains are concentrated at longer contexts. These results support both adding document-grounded QA targets and giving them sufficient weight.

Table 8: RULER scores by loss normalization.

Model	4K	8K	16K	32K
ProLong	99.3	96.7	88.7	75.8
ProLong-IS-Qwen (global avg.)	97.8	95.0	89.1	79.9
ProLong-IS-Qwen (QA avg.)	97.7	96.3	92.4	89.1
Gain (v.s. global avg.)	-0.1	+1.3	+3.3	+9.2

Ablation 3: learning-rate sweep. We sweep the learning rate η for the Qwen adaptation experiments under three training recipes: ProLong, ProLong-IS-Qwen (global avg.), and ProLong-IS-Qwen (QA avg.). For each recipe, we compare $\eta \in \{1, 2, 3, 4\} \times 10^{-5}$. Figure 7 reports the mean RULER accuracy over 4K, 8K, 16K, and 32K contexts. All three recipes attain their best score in this sweep at $\eta = 3 \times 10^{-5}$, where QA averaging reaches 93.8, compared with 90.5 for global averaging and 90.1 for ProLong. The ranking remains unchanged across all four learning rates: global averaging improves over ProLong by 0.4–1.3 points, while QA averaging yields gains of 3.7–5.7 points. Even the lowest score with QA averaging (92.5) exceeds the best score of either alternative. These results suggest that the benefit of QA-specific loss normalization persists across the tested learning rates, rather than depending on a single favorable setting.

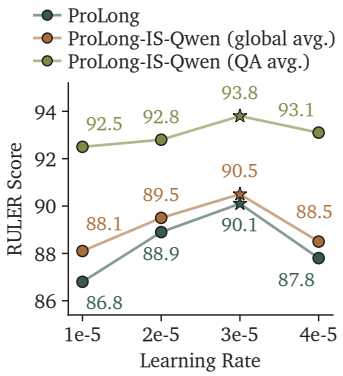


Figure 7: Mean RULER score over 4K, 8K, 16K, and 32K.

4 Conclusion

LDIS combines prefix-grounded QA interleaving with separately normalized answer supervision. It reshapes training prefixes and answer weights to encourage finite-state recurrent layers to retain and use earlier information. Across attention adaptation and context extension, it improves long-context performance under matched student token budgets while largely maintaining general capability relative to ProLong. The Nemotron experiment also demonstrates self-improvement using an internally trained teacher, with a new student initialized from the same base checkpoint.

AI USE STATEMENT

Beyond the model-based data synthesis described in Section 3, we used AI tools to polish language, proofread the manuscript, and check formatting.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Make Your LLM Fully Utilize the Context. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62160–62188, 2024. doi: 10.52202/079017-1986. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/71c3451f6cd6a4f82bb822db25cea4fd-Abstract-Conference.html.

Artificial Analysis Team. Artificial Analysis Long Context Reasoning Benchmark (LCR). Dataset, 2025. URL https://huggingface.co/datasets/ArtificialAnalysis/AA-LCR.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A Recipe for Long Context Alignment of Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395, 2024. doi: 10.18653/v1/2024.findings-emnlp.74. URL https://aclanthology.org/2024.findings-emnlp.74/.

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)*, pp. 3639–3664. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Jianghao Chen, Junhong Wu, Yangyifan Xu, and Jiajun Zhang. LADM: Long-context Training Data Selection with Attention-based Dependency Measurement for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 3076–3090, 2025a. doi: 10.18653/v1/2025.acl-long.154. URL https://aclanthology.org/2025.acl-long.154/.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending Context Window of Large Language Models via Positional Interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/211ab571cc9f3802afa6ffff52ae3e5b-Abstract-Conference.html.

Zhi Chen, Qiguang Chen, Libo Qin, Qipeng Guo, Haijun Lv, Yicheng Zou, Hang Yan, Kai Chen, and Dahua Lin. What Are the Essential Factors in Crafting Effective Long Context Multi-Hop Instruction Datasets? Insights and Best Practices. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 27129–27151, 2025b. doi: 10.18653/v1/2025.acl-long.1316. URL https://aclanthology.org/2025.acl-long.1316/.

Daixuan Cheng, Yuxian Gu, Shaohan Huang, Junyu Bi, Minlie Huang, and Furu Wei. Instruction Pre-Training: Language Models are Supervised Multitask Learners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2529–2550, 2024. doi: 10.18653/v1/2024.emnlp-main.148. URL https://aclanthology.org/2024.emnlp-main.148/.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando de Freitas, and Caglar Gulcehre. Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models, 2024. URL https://arxiv.org/abs/2402.19427.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11091–11104. PMLR, 2024. URL https://proceedings.mlr.press/v235/ding24i.html.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A Hybrid-head Architecture for Small Language Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f32def07618040e540e0a6182e290562-Abstract-Conference.html.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and General-Purpose Long Context Representations via Self-Study. *arXiv preprint arXiv:2506.06266*, 2025.

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*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14125–14134. PMLR, 2024. URL https://proceedings.mlr.press/v235/fu24d.html.

Chaochen Gao, Xing Wu, Zijia Lin, Debing Zhang, and Songlin Hu. NExtLong: Toward Effective Long-Context Training without Long Documents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18566–18584. PMLR, 2025a. URL https://proceedings.mlr.press/v267/gao25n.html.

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)*, pp. 7376–7399, 2025b.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani,

Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are We Done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.262. URL https://aclanthology.org/2025.naacl-long.262/.

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/pdf?id=tEYskw1VY2.

Linda He, Jue Wang, Maurice Weber, Shang Zhu, Ben Athiwaratkun, and Ce Zhang. Scaling Instruction-tuned LLMs to Million-token Contexts via Hierarchical Synthetic Data Generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bb36593e5e438aac5dd07907e757e087-Abstract-Conference.html.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/pdf?id=kIoBbc76Sy.

Zhiyuan Hu, Yuliang Liu, Jinman Zhao, Suyuchen Wang, Yan Wang, Wei Shen, Qing Gu, Anh Tuan Luu, See-Kiong Ng, Zhiwei Jiang, and Bryan Hooi. LongRecipe: Recipe for Efficient Long Context Generalization in Large Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 11857–11870, 2025. doi: 10.18653/v1/2025.acl-long.581. URL https://aclanthology.org/2025.acl-long.581/.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning Pretrained Transformers into RNNs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10630–10643, 2021. doi: 10.18653/v1/2021.emnlp-main.830. URL https://aclanthology.org/2021.emnlp-main.830/.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Kimi Team. Kimi Linear: An Expressive, Efficient Attention Architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing Large Language Models to Gated Recurrent Structures. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 32452–32466. PMLR, 2025. URL https://proceedings.mlr.press/v267/lan25b.html.

Siheng Li, Cheng Yang, Zesen Cheng, Lemao Liu, Mo Yu, Yujiu Yang, and Wai Lam. Large Language Models Can Self-Improve in Long-context Reasoning, 2024a. URL https://arxiv.org/abs/2411.08147.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E. Weston, and Mike Lewis. Self-Alignment with Instruction Backtranslation. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=1oijHJBRsT.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A Hybrid Transformer-Mamba Language Model, 2024. URL https://arxiv.org/abs/2403.19887.

M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, et al. SuperGPQA: Scaling LLM Evaluation across 285 Graduate Disciplines, 2025. URL https://arxiv.org/abs/2502.14739.

MiniCPM Team, Wenhao An, Yingfa Chen, Yewei Fang, Jiayi Li, Xin Li, Yaohui Li, Yishan Li, Yuxuan Li, Biyuan Lin, et al. MiniCPM-SALA: Hybridizing Sparse and Linear Attention for Efficient Long-Context Modeling. *arXiv preprint arXiv:2602.11761*, 2026.

NVIDIA. Nemotron 3 Nano: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning, 2025a. URL https://arxiv.org/abs/2512.20848.

NVIDIA. Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models, 2025b. URL https://arxiv.org/abs/2504.03624.

Yuqi Pan, Jinghao Zhuang, Yupeng Feng, Fangzhi Zhong, Siyu Ding, Xuerui Qiu, Shaowei Gu, Bohan Sun, Zhiyong Qin, Yibo Zhong, et al. SpikingBrain2.0: Brain-Inspired Foundation Models for Efficient Long-Context and Cross-Platform Inference. *arXiv preprint arXiv:2604.22575*, 2026.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1645. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/46499a0622ecf568b72d17b61e45dbd5-Abstract-Datasets_and_Benchmarks_Track.html.

Qwen Team. Qwen3.5. Hugging Face Collection, 2026. URL https://huggingface.co/collections/Qwen/qwen35.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/pdf?id=Ti67584b98.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/84a7fc24ed52e8eff514c33e8ac76ea3-Abstract-Conference.html.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by Distilling Context, 2022. URL https://arxiv.org/abs/2209.15189.

Junfeng Tian, Da Zheng, Yang Chen, Rui Wang, Colin Zhang, and Debing Zhang. Untie the Knots: An Efficient Data Augmentation Strategy for Long-Context Pre-Training in Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 1223–1242, 2025. doi: 10.18653/v1/2025.acl-long.62. URL https://aclanthology.org/2025.acl-long.62/.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, et al. Michelangelo: Long Context Evaluations Beyond Haystacks via Latent Structure Queries. *arXiv preprint arXiv:2409.12640*, 2024.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and Accelerating Hybrid Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-1996. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/723933067ad315269b620bc0d2c05cba-Abstract-Conference.html.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 13484–13508, 2023. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html.

Longyun Wu, Dawei Zhu, Guangxiang Zhao, Zhuocheng Yu, Junfeng Ran, Xiangyu Wong, Lin Sun, and Sujian Li. LongAttn: Selecting Long-Context Training Data via Token-Level Attention. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19367–19380, 2025. doi: 10.18653/v1/2025.findings-acl.991. URL https://aclanthology.org/2025.findings-acl.991/.

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*, pp. 4643–4663, 2024. doi: 10.18653/v1/2024.naacl-long.260. URL https://aclanthology.org/2024.naacl-long.260/.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. ChatQA 2: Bridging the Gap to Proprietary LLMs in Long Context and RAG Capabilities. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a7b562dac391e9c7af691e8ef886ad10-Abstract-Conference.html.

Xiaoyue Xu, Sikui Zhang, Xiaorong Wang, Xu Han, and Chaojun Xiao. Beyond Reward Engineering: A Data Recipe for Long-Context Reinforcement Learning, 2026. URL https://arxiv.org/abs/2606.18831.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4904fad153f6434a7bcf04465d4be2cc-Abstract-Conference.html.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to Evaluate Long-context Models Effectively and Thoroughly. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f5332c8273d02729730a9c24dec2135e-Abstract-Conference.html.

Yijiong Yu, Jiale Liu, Qingyun Wu, Huazheng Wang, and Ji Pei. SWAA: Sliding Window Attention Adaptation for Efficient and Quality Preserving Long Context Processing. *arXiv preprint arXiv:2512.10411*, 2025.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. LoLCATs: On Low-Rank Linearizing of Large Language Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/72163d1c3c1726f1c29157d06e9e93c1-Abstract-Conference.html.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023.

Yanke Zhou, Yiduo Li, Hanlin Tang, Maohua Li, Kan Liu, Tao Lan, Lin Qu, Yuan Yao, and Xiaoxing Ma. Full attention strikes back: Transferring full attention into sparse within hundred training steps. *arXiv preprint arXiv:2605.16928*, 2026.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/524ef58c2bd075775861234266e5e020-Abstract-Conference.html.

CONTENTS

1 Introduction **1**

2 Preliminaries **2**

2.1 Efficient Attention Blocks and Hybrid Architectures 2

2.2 Interleaved Study 2

2.3 Experiment Setups 4

2.3.1 Experiment 1: Adapting SSM-FA to SSM-SWA 4

2.3.2 Experiment 2: Long-Context Extension of Hybrid Base Model 4

2.4 Interleaved Study Pipeline in Practice 4

3 Experiment Details and Results **5**

3.1 Experiment 1: Adapting SSM-FA to SSM-SWA 5

3.2 Experiment 2: Long-Context Extension of Hybrid Model 7

3.3 Ablation Studies 8

4 Conclusion **9**

A Related Work **17**

B Interleaved Study Pipeline in Practice **20**

C General Capability Results **22**

D Additional Ablation Details **23**

E The ProLong Dataset **24**

F Full Seed-Prompt Pool **25**

G Selection of Retained Full-Attention Layers **29**

H Inference Efficiency of Qwen3.5-SWA **30**

H.1 Setup and Implementation 30

H.2 Measurement Protocol 30

H.3 Results 31

A Related Work

Long-context extension and training data. Early context-window extension methods primarily address positional generalization. Position Interpolation rescales position indices into the range observed during pretraining, while YaRN uses frequency-dependent interpolation to improve extension efficiency (Chen et al., 2023; Peng et al., 2024). LongRoPE searches for non-uniform rescaling factors and applies progressive extension, whereas PoSE exposes short training sequences to skipped positions sampled from the target context range (Ding et al., 2024; Zhu et al., 2024). Beyond positional changes, LongLoRA reduces adaptation cost with shifted sparse attention, and LongRecipe combines token selection, transformed position indices, and data replay (Chen et al., 2024; Hu et al., 2025). Subsequent work shows that positional extension alone does not guarantee effective use of long inputs: continual-pretraining performance also depends on the sequence-length curriculum, domain balance, and amount of long-form data (Xiong et al., 2024; Fu et al., 2024). ProLong further studies these choices under broad downstream evaluation and constructs a training mixture from books, code repositories, textbooks, and high-quality short-context data (Gao et al., 2025b). We adopt ProLong as a base corpus, preserve its domain mixture, and instead study how the supervision carried by each fixed-length sequence can be strengthened under matched student-token budgets.

Long-context data curation and synthesis. Recent methods improve long-context supervision either by selecting examples with genuine long-range dependencies or by constructing synthetic long-context tasks. LADM and LongAttn select native sequences using attention-derived dependency signals, while NExtLong and Untie the Knots create longer-range dependencies through hard-negative interleaving or chunk reordering (Chen et al., 2025a; Wu et al., 2025; Gao et al., 2025a; Tian et al., 2025). For alignment, LongLoRA introduces the LongAlpaca dataset, and LongAlign expands task diversity together with packing and loss-weighting strategies (Chen et al., 2024; Bai et al., 2024). IN^2 mixes answer-bearing segments with distractors, ChatQA 2 inserts answer-grounding summaries into long sources, and hierarchical synthesis constructs multi-document QA data at million-token scale (An et al., 2024; Xu et al., 2025; He et al., 2025). MIMG further generates and verifies multi-hop questions over long documents (Chen et al., 2025b). Concurrent work by Xu et al. (2026) constructs an eight-dataset mixture spanning retrieval, multi-evidence synthesis, and reasoning for long-context reinforcement learning with a minimal outcome-based Group Relative Policy Optimization (GRPO) objective. These approaches primarily improve document or task selection and form document-level training instances, including for post-training reasoning. Our pipeline instead preserves the order of the retained source text and inserts prefix-grounded QA probes at multiple causal positions, providing intermediate supervision throughout the recurrent-state trajectory during pretraining, continual training, or architectural adaptation.

Synthetic supervision and context distillation. Model-generated supervision provides a scalable alternative to manually annotated instruction data. Self-Instruct bootstraps instruction–input–output triples from a small seed set, while instruction backtranslation assigns model-generated instructions to naturally occurring text and iteratively curates the resulting examples (Wang et al., 2023; Li et al., 2024b). Instruction Pre-Training extends document-conditioned synthesis to pretraining by augmenting raw corpora with generated instruction–response pairs (Cheng et al., 2024). More broadly, distillation transfers teacher distributions or sequences, behavior conditioned on privileged context, or feedback on student-generated trajectories (Hinton et al., 2015; Kim & Rush, 2016; Snell et al., 2022; Agarwal et al., 2024). Self-training can also bootstrap from a model's own candidates, including for long-context reasoning (Li et al., 2024a). The closest conceptual precursor to our method is Cartridges, whose Self-Study procedure generates synthetic conversations about a corpus and distills full-context behavior into a compact, corpus-specific KV-cache representation (Eyuboglu et al., 2025). It motivates using task-oriented synthetic supervision to learn compressed representations that preserve useful document information. LDIS brings this idea to shared model-parameter training: prefix-grounded QA is interleaved with the original document, and answer supervision is normalized separately from document next-token prediction. Interleaving changes the distribution of prefixes and targets encountered during training, encouraging finite-state recurrent layers to retain earlier information for use at multiple positions. Separate normalization gives these targets explicit weight despite their small share of the sequence. The method keeps the student training-token budget fixed and can use either an external teacher or an internally trained checkpoint.

Efficient long-context architectures. State-space and linear-recurrent sequence mixers compress preceding tokens into a fixed-size state, avoiding the sequence-length-dependent KV cache of full attention. Mamba-2 connects selective SSMs with structured masked attention through state-space duality, while Gated DeltaNet augments its decayed recurrence with delta-rule memory edits (Dao & Gu, 2024; Yang et al., 2025). Sliding-window attention offers a complementary mechanism that preserves exact softmax interactions within a bounded local window (Beltagy et al., 2020). Griffin and Samba combine gated recurrence or Mamba with local attention, while Jamba and Hymba explore layer-wise and intra-layer mixtures of SSM and attention components (De et al., 2024; Ren et al., 2025; Lieber et al., 2024; Dong et al., 2025). Recent large-scale hybrids include Nemotron-H, Kimi Linear, and Qwen3.5 (NVIDIA, 2025b; Kimi Team, 2025; Qwen Team, 2026). The bounded memory of their recurrent or local-attention components motivates our architecture-aware data design. A detailed comparison of recent adaptation approaches that form the closest architectural context for our work is provided in Table 9.

Adapting pretrained models to efficient architectures. Earlier work converts pretrained Transformers into RNNs by replacing attention and then fine-tuning the modified network (Kasai et al., 2021). Recent conversion methods use cross-architecture distillation, attention transfer with low-rank adaptation, or direct gated-recurrent linearization (Wang et al., 2024a; Zhang et al., 2025; Lan et al., 2025). Recent adaptation methods replace or sparsify attention and repair the resulting mismatch through lightweight fine-tuning, distillation, or continued training (Yu et al., 2025; MiniCPM Team et al., 2026; Pan et al., 2026; Zhou et al., 2026). Unlike these works, we co-design data and objectives for bounded recurrent and local memory across training regimes.

Table 9 compares attention-only sparsification and Transformer-to-hybrid conversion.

Table 9: Adaptation of pretrained models for efficient long-context inference, distinguished by architectural changes and training strategy. SWA denotes sliding-window attention; KV denotes key–value.

Work Starting model	**Adaptation**	**Attention / memory**	**Training strategy**
SWAA (Yu et al., 2025) Full-attention Transformers	Attention-only sparsification No recurrent layers.	SWA with optional sink tokens; full attention retained in selected layers or during decoding.	SWA-aware long-context fine-tuning; self-distillation for thinking models.
MiniCPM-SALA (MiniCPM Team et al., 2026) MiniCPM-4.0 intermediate checkpoint	Transformer-to-hybrid conversion	Layer-selected mix of InfLLM-V2 sparse attention and Lightning Attention (linear).	Conversion, continual pretraining with long-context extension, then supervised fine-tuning.
SpikingBrain2.0 (Pan et al., 2026) Qwen3-4B-Instruct-2507; Qwen3-VL-4B-Instruct	Transformer-to-hybrid conversion	MoBA over the uncompressed KV cache; SSE-GDN over compressed recurrent states with an auxiliary SWA branch; final layer retains full attention.	Language model: distillation, progressive context extension, then general and reasoning fine-tuning. Vision–language model: distillation and supervised fine-tuning.
RTPurbo *Full Attention Strikes Back* (Zhou et al., 2026) Qwen3-Coder-30B-A3B; Qwen3-30B-A3B-Think	Attention-only sparsification No recurrent layers.	Local heads: sink tokens and SWA. Retrieval heads: full attention at prefill; indexed dynamic top-p token selection at decoding.	Indexer training, then self-distillation from the full-attention model.

B Interleaved Study Pipeline in Practice

We describe our implementation of LDIS (Section 2.2) on ProLong (Gao et al., 2025b). Appendix E provides additional details on the ProLong corpus and its role in our experiments. Figure 2 summarizes the four-stage pipeline.

Stage 1: Raw documents to packed long-context sequences. We start from the original ProLong corpus and pack it into fixed-length rows of length L with right padding. Each raw pack stores four columns: `input_ids`, `labels`, `attention_mask`, and `position_ids`. The `attention_mask` and `position_ids` columns are used to construct cross-document attention masks during training, following Gao et al. (2025b). The raw packs provide `labels` for natural-text prediction, with padding positions set to -100.

Stage 2: Batch prefix-grounded QA generation. As described in Section 2.2, we augment a subset $\mathcal{D}_{\text{studied}}$ of the packed rows with synthesized QA pairs. Our final recipe augments the rows from three single-document domains in ProLong: `book`, `textbooks`, and `thestackv1_concat_by_repo`. These domains provide native long-context data of up to 512K tokens. We find them more suitable for augmentation than domains that concatenate multiple short documents, whose rows are more prone to hallucinated content during QA generation.

For each row $\mathcal{C} \in \mathcal{D}_{\text{studied}}$, we split it into n equal-length chunks of size $\ell := L/n$ and construct the growing causal prefix $x_{1:i\ell}$ for the i-th insertion point. The teacher π_t generates questions and answers according to (2.1) and (2.2). For each $(\mathcal{C}, i)$, we randomly sample the seed prompt $p_{\mathcal{Q}}$ from a pool and generate $m = 4$ candidate QA pairs. Appendix F presents the full seed-prompt pool, and Figure 8 gives three examples.

LONG-DISTANCE SINGLE-FACT RECALL
Seed prompt. Scan the WHOLE text above, from its very beginning to its end, and pick ONE specific factual detail (a name, date, number, identifier, equation, code symbol, place, or technical term) that was introduced EARLY ON and is easy to overlook. Write a single question whose answer is that detail. Prefer a fact stated long before the end of the text rather than something near the end.
IMPORTANT output rules: appended below.

MULTI-HOP ACROSS DISTANT PARTS
Seed prompt. Find TWO related facts that appear in DIFFERENT, well-separated parts of the text above (ideally one early and one later). Write a single question that can only be answered by combining both facts—a two-hop question. Name the entities involved so the question stands on its own. Example shape: 'Who <relation-from-fact-1> the <thing> that <property-from-fact-2>?'
IMPORTANT output rules: appended below.

STATE TRACKING
Seed prompt. Find a quantity, status, score, balance, position, or assignment in the text above that CHANGES or is UPDATED more than once as the text progresses. Write a single question asking for its FINAL or current value (or who/what holds it now), such that answering correctly requires following the updates across the whole text. Name the entity being tracked.
IMPORTANT output rules: appended below.

IMPORTANT output rules:
– The question must be a standalone natural-language question that stands alone OUTSIDE the document.
– The question must NOT mention or reference 'chunk', 'section number', 'passage', 'the document', 'the text above', 'recently read', 'earlier in the text', 'later in the text', or any similar meta-phrase about the surrounding text or about how it was presented.
– You MAY connect facts that are far apart, but always identify each fact by the NAMES of the people, places, terms, equations, code symbols, or events involved—never by where it sits in the text.
– If you mention any chapter / section / part / figure / table label (e.g. 'Chapter 5', 'Section 3.2', 'Part II', 'Figure 4'), that EXACT label MUST already appear verbatim in the text above. Otherwise refer to the content by NAMES of characters, places, terms, equations, or events.
– Refer to people, places, terms, and concepts by their NAMES from the text, not by their position in the text.
– Output the question only—no preamble (no 'Based on the text...', no '**Item:**', no 'Question:'), no quotation, no labels.

Figure 8: **Seed prompt examples.** Three representative prompts for prefix-grounded QA generation and the shared output rules appended to each prompt.

Stage 3: Quality filtering and QA insertion. Synthesized QA pairs can be noisy, so we use both prompt-level and assembly-time safeguards. First, the prompt-level safeguards are applied during Stage 2: the prompts instruct the teacher to avoid meta-references such as "the above text" and forbid fabricated position labels unless they appear verbatim in the source. Figure 8 illustrates these

prompt requirements. Second, after generation, a rule-based filter rejects candidates that introduce unidentifiable position labels, leak the answer into the question, give a refusal-style answer, or substantially duplicate a previous answer in the same row. Rejected pairs are omitted. Finally, we optionally ask a document-grounded LLM to judge whether each remaining QA pair is supported by the source prefix. We retain at most one eligible QA pair per insertion slot.

After filtering, the retained QA pairs are inserted at their corresponding chunk boundaries, and the augmented sequence $\widetilde{\mathcal{C}}$ is truncated to the original length L. For adapted Qwen models, the inserted QA pairs use the Qwen ChatML format; for Nemotron base models without a trained chat template, we use a natural `Question:/Answer:` format. We extend the original data columns with `loss_mask`, `distill_positions`, `distill_target_ids`, and `distill_target_logprobs`. The `loss_mask` and `distill_positions` columns identify answer-token positions.

Stage 4: Blending and consumption. The final training set retains the same weighted blend of domains as the original ProLong. To keep the data schema uniform, unaugmented rows $\mathcal{C} \in \mathcal{D}_{\text{un}}$ receive zero-filled distillation columns, except that `distill_positions` is set to -1; these rows contribute only to $\mathcal{L}_{\text{NTP}}$. For augmented rows, the collator masks QA tokens from the next-token-prediction labels while keeping them visible in the input sequence. Question tokens are therefore never directly supervised, and answer tokens are supervised only through $\mathcal{L}_{\text{QA}}$. This implements the intended separation between $\mathcal{L}_{\text{NTP}}$ and $\mathcal{L}_{\text{QA}}$ in each training batch.

C General Capability Results

Tables 10 and 11 report the full standard-capability evaluations for Qwen3.5-4B-SWA adaptation and Nemotron-H-4B length extension, respectively. We include individual benchmark scores and the knowledge, instruction-following, and overall averages.

Table 10: Standard capability benchmark evaluation for Qwen3.5-4B-SWA adaptation.

Model / Recipe	Stage	MMLU Redux	MMLU Pro	Super GPQA	GPQA Dia.	IF Eval	IF Bench	Multi Chal.	Know. Avg.	Instr. Avg.	Avg.
Reference Model											
Qwen3.5-4B	N/A	84.4	69.2	36.7	42.4	82.3	33.0	40.6	58.2	52.0	55.5
Qwen3.5-4B-SWA-2FA											
ProLong	256K	**82.4**	**66.6**	35.2	**45.5**	47.0	**29.3**	30.5	**57.4**	35.6	**48.1**
ProLong-IS-GPT	256K	82.0	65.8	**35.6**	40.4	47.5	23.0	25.6	56.0	32.0	45.7
ProLong-IS-Qwen	256K	81.7	66.1	34.2	37.9	**51.8**	28.0	**31.6**	55.0	**37.1**	47.3
Qwen3.5-4B-SWA-3FA											
ProLong	256K	**82.6**	66.8	**36.0**	39.4	47.0	25.3	**29.7**	56.2	34.0	46.7
ProLong-IS-GPT	256K	81.6	**66.9**	35.9	**48.0**	49.7	23.7	22.9	**58.1**	32.1	47.0
ProLong-IS-Qwen	256K	81.9	66.4	35.8	36.4	**51.4**	**31.3**	27.1	55.1	**36.6**	**47.2**

Qwen3.5-4B-SWA adaptation. Table 10 shows that the effect on general capability depends on the teacher and architecture. Relative to ProLong, knowledge averages change by -2.4 to $+1.9$ points, while instruction-following averages improve with the Qwen teacher and decrease with GPT-5-Nano. The largest overall decrease occurs for the 2FA variant with GPT-5-Nano (48.1 to 45.7); the remaining recipes stay within 0.8 points of their respective ProLong baselines.

Table 11: Standard capability benchmark evaluation for Nemotron-H-4B length extension.

Model / Recipe	Stage	MMLU Redux	MMLU Pro	Super GPQA	GPQA Dia.	IF Eval	IF Bench	Multi Chal.	Know. Avg.	Instr. Avg.	Avg.
Nemotron-H-4B-64K											
ProLong	64K CPT + SFT	64.1	37.7	**18.6**	19.2	42.0	18.0	15.4	34.9	25.1	30.7
ProLong-IS-GPT	64K CPT + SFT	**64.6**	**37.8**	**18.6**	**20.2**	**44.4**	19.0	15.0	**35.3**	26.1	**31.4**
ProLong-IS-Nemotron	64K CPT + SFT	61.2	33.8	18.1	12.1	42.9	**21.3**	**17.7**	31.3	**27.3**	29.6

Nemotron-H-4B length extension. Table 11 reports the corresponding standard-capability results. ProLong-IS-GPT slightly improves both the knowledge and instruction-following averages over ProLong, raising the overall average from 30.7 to 31.4. ProLong-IS-Nemotron increases the instruction-following average from 25.1 to 27.3, while its knowledge average decreases from 34.9 to 31.3, yielding an overall average of 29.6. Together, these results show that long-context gains need not come at the expense of general capability, while also revealing a teacher-dependent trade-off between knowledge and instruction following.

D Additional Ablation Details

Section 3.3 reports the main ablations of QA placement and loss normalization. Here we give the comparison protocols and numerical learning-rate results.

QA placement protocol. Both QA-placement variants use Qwen3.5-4B-SWA-2FA, with Qwen3.5-4B as teacher, and are evaluated after the 64K adaptation stage. ProLong-IS-Qwen interleaves the generated QA pairs at their corresponding chunk boundaries. ProLong-IS-Qwen-End instead places them together at the end of each document. Both variants use the same training recipe and separate answer-loss normalization. Tables 7 and 8 report the QA-placement and loss-normalization results, respectively.

Global token averaging. The global-averaging variant uses a single next-token-prediction objective over both original-document and inserted answer tokens, masking only question tokens from the loss. For unaugmented documents, we take $\widetilde{x}_t^{(n)} = x_t^{(n)}$ and $\mathbf{1}_{t,\mathrm{ans}}^{(n)} = 0$. Equation (3.1) gives the objective used in Section 3.3. The denominator counts all supervised tokens, so original-document and answer tokens receive equal per-token weight, without a separate $\mathcal{L}_{\mathrm{QA}}$ loss term or a weighting coefficient λ. The question tokens remain visible in the causal input but receive no direct supervision.

Learning-rate sweep. We sweep $\eta \in \{1, 2, 3, 4\} \times 10^{-5}$ under three recipes: original ProLong, ProLong-IS-Qwen with global token averaging, and ProLong-IS-Qwen with separate answer normalization (QA averaging). Table 12 gives the scores plotted in Figure 7. Each entry is the unweighted mean RULER accuracy over 4K, 8K, 16K, and 32K contexts. These four-length averages differ from the five-benchmark averages in the main long-context evaluation tables.

Table 12: Learning-rate sweep for Qwen3.5-4B-SWA-2FA. Bold entries mark the best tested learning rate for each recipe.

Training recipe	1×10^{-5}	2×10^{-5}	3×10^{-5}	4×10^{-5}
ProLong	86.8	88.9	**90.1**	87.8
ProLong-IS-Qwen (global avg.)	88.1	89.5	**90.5**	88.5
ProLong-IS-Qwen (QA avg.)	92.5	92.8	**93.8**	93.1

Global averaging improves over ProLong by 0.4–1.3 points across the tested learning rates, while QA averaging improves over ProLong by 3.7–5.7 points. These results support the use of separate answer normalization across the tested settings.

E THE PROLONG DATASET

Release and original training recipe. The ProLong data release accompanies the long-context continued-pretraining study of Gao et al. (2025b) and provides separately packed 64K and 512K corpora.[6] The records are tokenized with the Llama-3 tokenizer and packed to 65,536 or 524,288 tokens; document-boundary indices permit cross-document attention masking. In the original recipe, Llama-3-8B-Instruct is continually trained for 20B tokens at 64K and then for another 20B tokens at 512K before supervised fine-tuning on UltraChat (Gao et al., 2025b). We use the released corpora, rather than the ProLong model weights, as the common data source for our ProLong and ProLong-IS recipes.

Domain mixture and length curriculum. ProLong combines native long-context sources with a curated short-context mixture. Its data ablations identify books and repository-level code as complementary long-document sources and show that retaining high-quality short-context data is important for both long- and short-context downstream performance (Gao et al., 2025b). The two training stages use the same global domain mixture: 30% code repositories, 30% books, 3% textbooks, and 37% ShortMix. Only the sequence-length composition within each domain changes between stages, as summarized in Table 13. Within ShortMix, FineWeb-Edu and FineWeb each account for 27%, Wikipedia and StackExchange each account for 11%, and Tulu-v2, OpenWebMath, and ArXiv each account for 8% (Gao et al., 2025b).

Table 13: The final ProLong mixture and length curriculum (Gao et al., 2025b). Global shares apply to both stages; Stage 2 percentages are conditional within each domain.

Component	Global share	Stage 1: 64K	Stage 2: up to 512K
Code repositories	30%	Native 64K rows	50% at 64K; 50% at 512K
Books	30%	Native 64K rows	83% at 64K; 17% at 512K
Textbooks	3%	Packed at 64K	Packed at 512K
ShortMix	37%	Short documents packed to 64K	Short documents packed to 512K

In this work. We retokenize ProLong for each student and retain the global shares in Table 13. The three QA-augmented domains are `book`, `textbooks`, and `thestackv1_concat_by_repo`.

Table 14: ProLong usage in our experiments. Token budgets are measured after augmentation and matched between ProLong and ProLong-IS; unlisted rows remain unchanged. The two Qwen3.5-4B-SWA stages use 14B tokens in total.

Experiment	Ctx.	Tokens	Source	QA augmentation scope
Qwen3.5-SWA (0.8B / 2B)	64K	1.5B / 4B	64K split	All three single-doc domains
Qwen3.5-4B-SWA, Stage 1	64K	7B	64K split	All three single-doc domains
Qwen3.5-4B-SWA, Stage 2	256K	7B	512K split $\rightarrow$ 256K	Native 512K row from single-doc domains
Nemotron-H	64K	7B	64K split	All three single-doc domains

[6] `princeton-nlp/prolong-data-64K` and `princeton-nlp/prolong-data-512K`.

F FULL SEED-PROMPT POOL

Our seed-prompt pool contains 29 prompts organized into ten categories that target retrieval, integration, tracking, aggregation, and synthesis over long contexts. For each insertion point, we sample one prompt uniformly at random and append it to the accumulated causal prefix. The same output-rule tail is attached to every prompt. Figure 8 visualizes three representative prompts; the complete pool is reproduced below.

Prompt-level safeguards. The following output-rule tail is appended to every sampled seed prompt.

IMPORTANT output rules:
– The question must be a standalone natural-language question that stands alone OUTSIDE the document.
– The question must NOT mention or reference "chunk," "section number," "passage," "the document," "the text above," "recently read," "earlier in the text," "later in the text," or any similar meta-phrase about the surrounding text or about how it was presented.
– You MAY connect facts that are far apart, but always identify each fact by the NAMES of the people, places, terms, equations, code symbols, or events involved—never by where it sits in the text.
– If you mention any chapter / section / part / figure / table label (e.g. "Chapter 5," "Section 3.2," "Part II," "Figure 4"), that EXACT label MUST already appear verbatim in the text above. Otherwise refer to the content by NAMES of characters, places, terms, equations, or events.
– Refer to people, places, terms, and concepts by their NAMES from the text, not by their position in the text.
– Output the question only—no preamble (no "Based on the text...," no "**Item:**," no "Question:"), no quotation, no labels.

Seed prompts. Each box below gives one complete seed prompt.

A1 LONG-DISTANCE SINGLE-FACT RECALL
Seed prompt. Scan the WHOLE text above, from its very beginning to its end, and pick ONE specific factual detail (a name, date, number, identifier, equation, code symbol, place, or technical term) that was introduced EARLY ON and is easy to overlook. Write a single question whose answer is that detail. Prefer a fact stated long before the end of the text rather than something near the end.
IMPORTANT output rules: shared across the pool.

A2 LONG-DISTANCE SINGLE-FACT RECALL
Seed prompt. Choose a concrete value that appears exactly once anywhere in the text above—an exact figure, a coined term, a proper name, an account/ID number, a function name, or a constant. Write a single retrieval question that asks the reader to recover that precise value. The value may have been introduced anywhere in the text, near the start or the middle; favour ones that are far from where the reading currently ends.
IMPORTANT output rules: shared across the pool.

A3 LONG-DISTANCE SINGLE-FACT RECALL
Seed prompt. Identify a specific attribute attached to a named entity somewhere in the text above (e.g. a character's title, a city's population, a variable's assigned value, a parameter's default, a date associated with an event). Write one question asking for that attribute, naming the entity explicitly. The entity may have been introduced at any point in the text.
IMPORTANT output rules: shared across the pool.

B1 MULTI-HOP ACROSS DISTANT PARTS
Seed prompt. Find TWO related facts that appear in DIFFERENT, well-separated parts of the text above (ideally one early and one later). Write a single question that can only be answered by combining both facts—a two-hop question. Name the entities involved so the question stands on its own. Example shape: "Who <relation-from-fact-1> the <thing> that <property-from-fact-2>?"
IMPORTANT output rules: shared across the pool.

B2 MULTI-HOP ACROSS DISTANT PARTS
Seed prompt. Locate a chain of references in the text above where one named entity is described in one place and connected to another named entity elsewhere. Write a single multi-hop question that forces the reader to follow that connection from one named entity to the other to reach the answer.
IMPORTANT output rules: shared across the pool.

B3 MULTI-HOP ACROSS DISTANT PARTS
Seed prompt. Write ONE bridging question whose answer depends on linking an early statement to a later one—for instance, a cause introduced earlier with an effect mentioned later, or a definition given once and then used again under a different name. Identify both endpoints by name, not by position.
IMPORTANT output rules: shared across the pool.

C1 VARIABLE / STATE TRACKING
Seed prompt. Find a quantity, status, score, balance, position, or assignment in the text above that CHANGES or is UPDATED more than once as the text progresses. Write a single question asking for its FINAL or current value (or who/what holds it now), such that answering correctly requires following the updates across the whole text. Name the entity being tracked.
IMPORTANT output rules: shared across the pool.

C2 VARIABLE / STATE TRACKING
Seed prompt. Identify a named variable, character, object, or account whose state is established early and then modified later in the text above. Write one question that can only be answered correctly by tracking that entity through its changes from where it was introduced to the latest mention.
IMPORTANT output rules: shared across the pool.

C3 VARIABLE / STATE TRACKING
Seed prompt. Pick a value that is defined once and then re-bound or reassigned through one or more intermediate references in the text above (e.g. A is set, B is set from A, the answer depends on resolving B back to its root). Write a single question that requires resolving that chain of bindings. Refer to each link by its name.
IMPORTANT output rules: shared across the pool.

D1 AGGREGATION / COUNTING / LISTING
Seed prompt. Identify a category of items that recurs throughout the text above (e.g. all the cities visited, all the function names defined, all the people who spoke, all the dates mentioned, all the error codes). Write a single question asking the reader to LIST every member of that category that appears anywhere in the text, naming the category precisely.
IMPORTANT output rules: shared across the pool.

D2 AGGREGATION / COUNTING / LISTING
Seed prompt. Find something that occurs MULTIPLE times across the text above. Write a single counting question—"How many times does <named thing> happen?" or "How many distinct <named category> are mentioned?"—whose answer requires scanning the whole text, not just the latest part.
IMPORTANT output rules: shared across the pool.

D3 AGGREGATION / COUNTING / LISTING
Seed prompt. Pick a named entity (a person, place, term, or symbol) that appears in several different parts of the text above. Write a single question that asks the reader to gather and report ALL the distinct values, roles, or facts associated with that entity across the entire text (multi-value retrieval). Name the entity.
IMPORTANT output rules: shared across the pool.

D4 AGGREGATION / COUNTING / LISTING
Seed prompt. Determine which named item is the most frequent / most prominent / most repeated of its kind across the whole text above (e.g. the most mentioned character, the most-called function, the most-cited source). Write a single question asking for that superlative, phrased so the reader must aggregate over the entire text.
IMPORTANT output rules: shared across the pool.

E1 COMPARISON / CONTRAST
Seed prompt. Find two named things in the text above that can be meaningfully compared and that are introduced FAR APART (one earlier, one later)—two characters, two methods, two figures, two configurations, two events. Write a single comparison question (which is larger / earlier / faster / different how?) that names both and requires recalling both.
IMPORTANT output rules: shared across the pool.

E2 COMPARISON / CONTRAST
Seed prompt. Identify a property of a named entity stated early in the text above and a corresponding property of a related entity stated later. Write one question asking how the two differ or which one satisfies some condition, naming both entities.
IMPORTANT output rules: shared across the pool.

E3 COMPARISON / CONTRAST
Seed prompt. Write a single question that asks whether a claim, value, or description given early in the text above is consistent with—or contradicted by—something said later. Identify both statements by the names of the entities they concern.
IMPORTANT output rules: shared across the pool.

F1 ORDERING / TEMPORAL RELATIONS
Seed prompt. Pick a clearly identifiable event, step, or state in the text above. Write a single question asking what happened (or what was true) IMMEDIATELY BEFORE it, or what led to it—anchoring the named event but requiring the reader to recall earlier material to answer.
IMPORTANT output rules: shared across the pool.

F2 ORDERING / TEMPORAL RELATIONS
Seed prompt. Find a sequence of events, steps, or stages described across the text above. Write a single ordering question—"Which happened first, X or Y?" or "What came after X?"—naming the events involved, where X and Y are separated by a long span of text.
IMPORTANT output rules: shared across the pool.

F3 ORDERING / TEMPORAL RELATIONS
Seed prompt. Identify the FIRST occurrence or the ORIGIN of some named thing in the text above (the first time a character appears, where a term is first defined, the initial value before any changes). Write a single question asking for that earliest state, naming the thing.
IMPORTANT output rules: shared across the pool.

G1 WHOLE-PREFIX SUMMARIZATION
Seed prompt. Write a single instruction asking the reader to summarise the MAIN developments, arguments, or plot across EVERYTHING in the text above, from beginning to end, in a few sentences. Anchor it on the central named subject so the instruction stands alone.
IMPORTANT output rules: shared across the pool.

G2 WHOLE-PREFIX SUMMARIZATION
Seed prompt. Write a single instruction asking the reader to give the overall outcome, conclusion, or current situation that the WHOLE text above has built up to—requiring synthesis of early and late material. Name the central subject.
IMPORTANT output rules: shared across the pool.

G3 WHOLE-PREFIX SUMMARIZATION
Seed prompt. Write a single instruction asking the reader to trace how a named entity, theme, or quantity EVOLVES over the full span of the text above—what it was at the start versus what it is by the end.
IMPORTANT output rules: shared across the pool.

H1 ENTITY / COREFERENCE RESOLUTION
Seed prompt. Find a named entity in the text above that is later referred to by a different name, alias, title, role, or pronoun-heavy description far from where it was introduced. Write a single question that uses one of those references and asks the reader to identify WHO or WHAT it is, resolvable only by recalling the original introduction.
IMPORTANT output rules: shared across the pool.

H2 ENTITY / COREFERENCE RESOLUTION
Seed prompt. Identify a role, position, or description in the text above (e.g. "the person who founded the company," "the function that validates input," "the city where the treaty was signed") whose referent is established early. Write a single question asking the reader to name that referent.
IMPORTANT output rules: shared across the pool.

H3 ENTITY / COREFERENCE RESOLUTION
Seed prompt. Pick an entity introduced early in the text above and described again much later under additional detail. Write one question that requires matching the two mentions to answer—confirming they are the same entity and reporting a fact that needs both mentions.
IMPORTANT output rules: shared across the pool.

I1 MULTI-PASSAGE SYNTHESIS
Seed prompt. Write a single, natural open-domain-style question whose answer is stated somewhere in the text above but NOT near its end—the kind of factual question a reader might ask, answerable from one specific spot in the earlier or middle portion of the text. Anchor it on named content so it stands alone.
IMPORTANT output rules: shared across the pool.

I2 MULTI-PASSAGE SYNTHESIS
Seed prompt. Pose a single question that a reader could only answer by locating the one relevant fact among everything in the text above and ignoring unrelated material. Anchor it on a specific named subject, and prefer evidence that is not in the most recent part.
IMPORTANT output rules: shared across the pool.

J1 CHUNK-LOCAL RECALL
Seed prompt. Generate ONE question that tests knowledge of a specific factual detail introduced in the MOST RECENTLY ADDED portion of the text above (the latest part the reader has just finished). Anchor the question on a concrete surface element (name, date, number, identifier, equation, code symbol, place, or technical term) from that latest portion. The reader may consult any part of the document to answer.
IMPORTANT output rules: shared across the pool.

J2 CHUNK-LOCAL RECALL
Seed prompt. Find a term, concept, function, character, variable, or entity that is NEWLY introduced in the MOST RECENTLY ADDED part of the text above and not mentioned earlier. Write a single question asking the reader to define or describe this newly introduced item, referring to it by its EXACT name from the text.
IMPORTANT output rules: shared across the pool.

G SELECTION OF RETAINED FULL-ATTENTION LAYERS

Pairwise screening. The 0.8B and 2B models have six candidate FA layers at zero-indexed layers 3, 7, 11, 15, 19, and 23. The 4B model additionally includes layers 27 and 31. For each candidate pair $\{a, b\}$, we retain these two layers as FA, replace the other FA layers with SWA using a 512-token window, and evaluate RULER at 4K, 8K, 16K, and 32K. Figure 9 reports the RULER score mean over these four context lengths. Off-diagonal cells correspond to 2FA pairs, while diagonal cells retain only one FA layer and serve as controls.

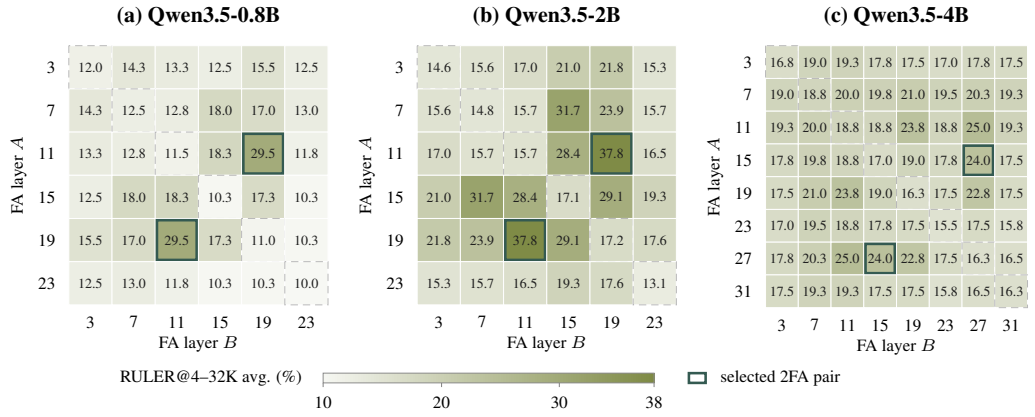


Figure 9: Pairwise screen for retained FA layers. Each cell reports the mean RULER accuracy over 4K, 8K, 16K, and 32K contexts when the indexed FA pair is retained and all other original FA layers use SWA-512. Diagonal cells retain a single FA layer. The three panels share one color scale, and outlined cells mark the 2FA choices used in our scaling experiments.

For both 0.8B and 2B, $\{11, 19\}$ is the strongest pair. This agreement suggests a stable relative-depth pattern: the third and fifth FA slots are preferred among the six candidates. Mapping these relative positions to the eight FA slots of the 4B model gives the fourth and seventh slots, or $\{15, 27\}$. This pair scores within 1.0 point of the best 4B pair $\{11, 27\}$, and is used for Qwen3.5-4B-SWA-2FA. The screen is used only to choose layer locations; the final models use a 4K SWA window and undergo the full adaptation recipe described in Section 3.1.

Extension to 3FA. For the Qwen3.5-4B-SWA-3FA configuration, we conduct an analogous three-dimensional sweep over candidate FA-layer triplets using the same evaluation protocol. We omit the full sweep results for brevity; the sweep selects layers $\{11, 19, 27\}$, which we use for Qwen3.5-4B-SWA-3FA.

H INFERENCE EFFICIENCY OF QWEN3.5-SWA

This appendix details the inference-efficiency measurements. We compare Qwen3.5-4B with Qwen3.5-4B-SWA before adaptation, separating steady-state decoding, prefill, per-sequence cached state, and end-to-end serving performance.

Table 15: Comparison of Qwen3.5 (FA*8) and Qwen3.5-SWA (FA*2) (*before* the adaptation fine-tuning stage)[7].

Model	Throughput ↑ (tok/s)	Memory ↓ (GiB/seq)	Max. batch ↑ (1×H100)	RULER ↑ ≤64K
Qwen3.5-4B	968	2.05	27	**96.1**
Qwen3.5-4B-SWA	**1,774** (1.8×)	**0.64** (3.2×)	**68** (2.5×)	52.7

H.1 SETUP AND IMPLEMENTATION

Models and attention configuration. The two models share the same weights. Qwen3.5-4B has 24 Gated DeltaNet layers and eight FA layers at zero-indexed positions $\{3, 7, 11, 15, 19, 23, 27, 31\}$. Qwen3.5-4B-SWA retains FA at $\{15, 27\}$ and replaces the other six FA layers with SWA of window size $w = 4096$; the Gated DeltaNet layers are unchanged. This is a configuration-only modification, without parameter updates.

Hardware and serving stack. All measurements use one NVIDIA H100 80GB HBM3 GPU, vLLM 0.16.1, and BF16 model weights and attention KV caches, without quantization or tensor parallelism. CUDA graphs are enabled and prefix caching is disabled. We set `gpu_memory_utilization` to 0.9 and enable chunked prefill with `max_num_batched_tokens`= 8192. A vLLM plugin implements the configured SWA layers using the standard Qwen3.5 attention block with a per-layer sliding window. The hybrid KV-cache manager bounds the retained KV state for these layers, while the two FA layers retain the full context. As an implementation check, the two models give identical token-wise prompt log-probabilities on a 4,032-token sequence, with maximum absolute difference zero and 100% argmax agreement. An 8,192-token control agrees within the initial window and diverges beyond it, as expected from the changed attention mask.

Workload. We use random-token prompts to control input length and greedy decoding with `ignore_eos` so that requests remain active during timing. The prompt lengths are $L \in \{4, 16, 32, 64, 128, 256\}$K, where $\mathrm{K} = 1024$ tokens, and the requested batch sizes are shown in Table 16. These fixed-shape workloads isolate inference costs; they do not measure task accuracy or represent a natural distribution of request lengths and stopping times.

H.2 MEASUREMENT PROTOCOL

Steady-state decode throughput. We advance the engine through `LLMEngine.step()` after submitting all requests together. Timing begins only after every request has produced its first token and the waiting queue is empty. We then time 256 engine steps, recording how many sequences produce a token at each step to verify that the full batch remains active. Decode throughput is the total number of tokens generated during this interval divided by its duration; average step latency is the duration divided by 256. Each reported cell is the median of three repeats after a warm-up run. This excludes the initial prefill period, including any decoding that chunked prefill interleaves with the admission of later requests.

[7] Efficiency is evaluated at 64K on one H100: throughput is steady-state decoding at batch 16, memory is analytic per-sequence cached state in GiB, and maximum batch is allocator-reported concurrent-request capacity. Parentheses report throughput and capacity gains or memory reduction factors. RULER≤64K averages RULER@$\{4, 8, 16, 32, 64\}$K.

Infeasible batches. If the KV budget cannot hold the requested batch, vLLM preempts some requests. With `ignore_eos`, the active requests do not finish and release their caches, so the remaining requests can starve. The benchmark detects this condition when no additional request produces its first token for 256 steps. We mark these configurations as infeasible rather than reporting preemption-and-recomputation rates as batch decode throughput. Speedups are computed only when both models sustain the requested batch.

Prefill throughput and time to first token. We measure the interval from request submission until all requests have produced their first token and divide the total prompt-token count by this interval. For batches larger than one, the interval also includes decoding interleaved with other requests' prefill. We therefore report batch-one measurements in Table 18, where the interval is the request's time to first token (TTFT). These are request-level measurements, not isolated attention-kernel timings.

Cached-state memory and capacity. Memory denotes the analytic per-sequence attention KV cache plus fixed-size Gated DeltaNet convolution and recurrent states, not total GPU memory or model weights. With four KV heads, head dimension 256, and two bytes per BF16 element, each attention layer stores $c_{\mathrm{KV}} = 2 \cdot 4 \cdot 256 \cdot 2 = 4096$ bytes per cached token. Let $M_{\mathrm{base}}(L)$ and $M_{\mathrm{SWA}}(L)$ denote the cached-state footprints of Qwen3.5-4B and Qwen3.5-4B-SWA, respectively. Writing M_{GDN} for the context-independent state of all 24 Gated DeltaNet layers, these footprints in bytes are

$$M_{\mathrm{base}}(L) = 8c_{\mathrm{KV}}L + M_{\mathrm{GDN}}, \qquad M_{\mathrm{SWA}}(L) = c_{\mathrm{KV}}\big(2L + 6\min\{L, w\}\big) + M_{\mathrm{GDN}}.$$

The analytic accounting uses FP32 Gated DeltaNet states, contributing approximately 0.05 GiB in total. We report memory in GiB (2^{30} bytes). Separately, we report the whole-request capacities obtained from the vLLM allocator logs under the same 90% GPU-memory budget. These capacities are not throughput-optimal batch sizes or guarantees of a serving latency target. The allocator reserves $w - 1 + 8192$ token positions for each windowed layer, rather than just w, and also accounts for cache grouping and allocation granularity; its capacity need not scale exactly as the analytic state size.

H.3 Results

Decode scaling. Table 16 reports the full decode sweep. At 64K, batch 16 is the largest tested batch that fits both models, with throughput increasing from 968 to approximately 1.77×10^3 tokens/s ($1.83\times$). At 256K, the corresponding common batch is four, with throughput increasing from 248 to 518 tokens/s ($2.09\times$). At 4K, throughput differs by at most about 3%; the larger gains emerge as context length and batch size increase. This trend is consistent with reduced attention-cache traffic, although the sweep alone does not identify the hardware bottleneck or attribute small differences to a particular scheduler component.

Table 16: Steady-state decode throughput on one H100. Ratios are Qwen3.5-4B-SWA / Qwen3.5-4B, computed from the displayed throughputs.

Context	Model	Decode throughput (tokens/s) ↑				
		$B = 1$	$B = 8$	$B = 16$	$B = 32$	$B = 64$
4K	Qwen3.5-4B	219	1,461	2,550	4,151	6,238
	Qwen3.5-4B-SWA	213	1,427	2,503	4,098	6,259
	Ratio	0.97×	0.98×	0.98×	0.99×	1.00×
16K	Qwen3.5-4B	213	1,227	1,912	2,700	3,432
	Qwen3.5-4B-SWA	211	1,361	2,311	3,667	5,187
	Ratio	0.99×	1.11×	1.21×	1.36×	1.51×
32K	Qwen3.5-4B	212	1,041	1,466	1,854	†
	Qwen3.5-4B-SWA	209	1,311	2,134	3,157	4,226
	Ratio	0.99×	1.26×	1.46×	1.70×	–
64K	Qwen3.5-4B	189	764	968	†	†
	Qwen3.5-4B-SWA	205	1,155	1,775	2,455	3,072
	Ratio	1.08×	1.51×	1.83×	–	–
128K	Qwen3.5-4B	167	491	†	†	–
	Qwen3.5-4B-SWA	196	962	1,371	1,728	–
	Ratio	1.17×	1.96×	–	–	–
256K	Qwen3.5-4B	134	†	†	–	–
	Qwen3.5-4B-SWA	184	738	933	–	–
	Ratio	1.37×	–	–	–	–

256K, $B = 4$, Qwen3.5-4B = 248, Qwen3.5-4B-SWA = 518 tokens/s (2.09×).

†: the requested batch does not fit; –: not measured or ratio unavailable.

Memory and concurrent capacity. Table 17 separates the analytic state size from allocator-reported capacity. At 64K, per-sequence state decreases from 2.05 to 0.64 GiB, while the concurrent-request capacity increases from 27 to 68. At 256K, the corresponding values are 8.05 versus 2.14 GiB and seven versus 26 requests. The analytic reduction approaches $4\times$ as L grows, because the leading full-context KV term is proportional to eight versus two FA layers. This asymptotic statement concerns cached-state memory, not an equivalent speedup for the entire model.

Table 17: Per-sequence cached state and allocator-reported capacity at 90% GPU-memory utilization. State excludes model weights and workspaces. Factors use the displayed values.

Model	4K	16K	32K	64K	128K	256K
Cached state (GiB/seq) ↓						
Qwen3.5-4B	0.17	0.55	1.05	2.05	4.05	8.05
Qwen3.5-4B-SWA	0.17	0.27	0.39	0.64	1.14	2.14
Reduction	1.00×	2.04×	2.69×	3.20×	3.55×	3.76×
Concurrent requests ↑						
Qwen3.5-4B	143	77	48	27	14	7
Qwen3.5-4B-SWA	139	115	93	68	44	26
Gain	0.97×	1.49×	1.94×	2.52×	3.14×	3.71×

Prefill and serving. Batch-one prefill throughput increases from 47,900 to 57,900 tokens/s at 64K, and from 25,200 to 46,500 tokens/s at 256K (Table 18). The corresponding TTFT decreases from 1.37 to 1.13 seconds and from 10.41 to 5.64 seconds. Replacing six FA layers reduces their attention cost from $\mathcal{O}(L^2)$ to $\mathcal{O}(Lw)$; however, the two retained FA layers still have quadratic prefill cost, so the complete model is not linear in context length.

Table 18: Batch-one prefill throughput and time to first token. Throughput is prompt tokens divided by TTFT; speedup and latency-reduction factors use the displayed values.

Model	4K	16K	32K	64K	128K	256K
Prefill (10^3 **tokens/s**) ↑						
Qwen3.5-4B	59.4	62.4	57.9	47.9	37.5	25.2
Qwen3.5-4B-SWA	59.6	63.6	62.4	57.9	54.4	46.5
Speedup	1.00×	1.02×	1.08×	1.21×	1.45×	1.85×
TTFT (s) ↓						
Qwen3.5-4B	0.07	0.26	0.57	1.37	3.49	10.41
Qwen3.5-4B-SWA	0.07	0.26	0.53	1.13	2.41	5.64
Reduction	1.00×	1.00×	1.08×	1.21×	1.45×	1.85×

For an end-to-end check, we run `vllm serve` with `vllm bench serve` using random 64K-token inputs, 256 output tokens, and concurrency limits of 8, 16, or 32. Each run contains twice as many requests as its concurrency limit. Table 19 reports total output throughput, mean time per output token (TPOT), and mean TTFT. Output throughput improves by 1.27–1.34×, and TPOT decreases by approximately 29–30% at concurrency 16 and 32. These gains are smaller than the decode-only speedups because the end-to-end workload also includes long-prompt prefill, and a request's inter-token latency can include prefill chunks scheduled for other requests. Unlike the nonterminating decode stress test, this workload allows requests to finish and release their caches; a concurrency limit of 32 therefore does not imply that all 32 requests decode simultaneously. The results characterize this hardware, software stack, and workload, rather than establishing hardware-independent speedup ratios.

Table 19: Online serving on one H100 with 64K input and 256 output tokens per request. Output throughput includes the full benchmark wall time, unlike the decode-only throughput in Table 16.

Concurrency	Model	Throughput ↑ (tokens/s)	Mean TPOT ↓ (ms)	Mean TTFT ↓ (s)
8	Qwen3.5-4B	145.0	30.8	6.1
	Qwen3.5-4B-SWA	184.5	23.0	5.1
16	Qwen3.5-4B	147.6	65.4	10.6
	Qwen3.5-4B-SWA	192.5	46.5	9.0
32	Qwen3.5-4B	143.8	144.2	19.1
	Qwen3.5-4B-SWA	193.1	102.9	15.6