MASK-ROUTED SPARSE ATTENTION FOR LONG-CONTEXT DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models apply bidirectional attention over the whole context at every denoising step, so a longer context makes each step substantially more expensive. Sparsifying only at inference leaves the pretrained weights unchanged. We propose Mask-Routed Sparse Attention (MRSA), which continued-trains a diffusion language model to route with its native mask token. One mask is appended at the end of each chunk, and a learned role offset marks it as a routing slot. On every fourth layer, a query keeps a bidirectional local window and the top-k chunks scored from that mask; the other layers use a bidirectional sliding window. Starting from Dream-v0-Base-7B extrapolated to 16K with YaRN, we compare MRSA with bidirectional NSA, DSA, and SWA under the same layout, key budget, and dense initialization, using the same Fast-dLLM decoder. Figure 1 shows that, with the YaRN factor scaled to the evaluation length, MRSA leads the other sparse methods on all three RULER probes from 16K to 128K. Its mean accuracy at 128K is 18.4% above NSA, the next sparse model, while DSA and SWA stay near the floor. At the same length, decode is $172\times$ faster than full dense.

1 INTRODUCTION

Diffusion language models (dLLMs) generate text by iteratively unmasking tokens under bidirectional attention, rather than by left-to-right next-token prediction (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2026; Ye et al., 2025). At every denoising step, each query can attend to the whole context, so a longer context makes that step substantially more expensive.

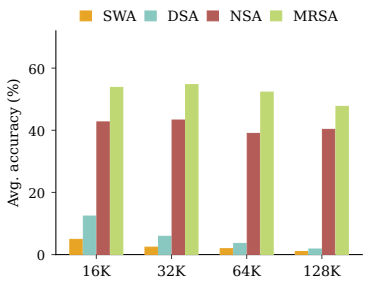

Figure 1: Mean accuracy of the four sparse methods on the three RULER probes. The YaRN factor scales with the evaluation length.

Sparse attention in autoregressive models allocates a limited budget with a local window, a learned token-level route, or a mixture of compressed and selected blocks (Jiang et al., 2023; Yuan et al., 2025; DeepSeek-AI, 2025). These designs focus on causal attention, whereas efficient dLLM inference is studied mainly in a *training-free regime*. Fast-dLLM reuses keys and values across denoising steps and decodes blocks in parallel; it accelerates serving but does not sparsify attention (Wu et al., 2025b). Sparse-dLLM evicts cache entries (Song et al., 2025), SparseD builds a head-specific block-sparse MASK at inference time (Wang et al., 2025), and Prefilling-dLLM scores prefix chunks and reuses a static top-K cache (Xiong et al., 2026). Sparsifying only at inference leaves the pretrained dense attention weights unchanged and restricts which tokens are visible. Training a dLLM under a sparse attention budget has received comparatively less study.

Our key insight is that a mask token is trained to absorb the context needed to predict an unknown token. Fixed at the end of a chunk, it can summarize that chunk and indicate which remote chunks a query should read. Thus we propose **M**ask-**R**outed **S**parse **A**ttention (MRSA), a *continued-training* sparse attention method that selects remote chunks with these summaries. Figure 2 contrasts the resulting visibility with bidirectional adaptations of Native Sparse Attention (NSA), DeepSeek Sparse Attention (DSA), and sliding-window attention (SWA).

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

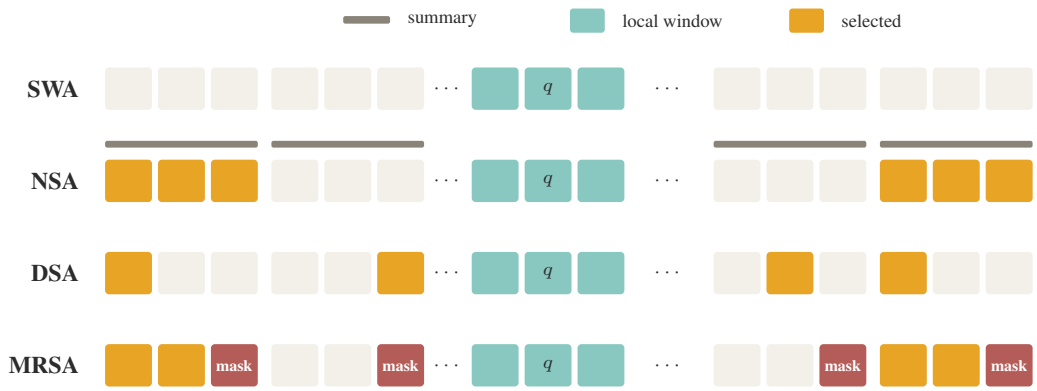

Figure 2: What query q can see. Each chunk is three cells, and q sits in the middle so the local window extends both ways. Teal cells are that window and orange cells are selected. The bar above each NSA chunk is its summary. DSA reads scattered tokens, and MRSA marks the last cell of every chunk as a mask.

Specifically, MRSA appends the model's native mask token at the end of each chunk. A binary type mask marks that position as routing rather than generation, and a learned role offset is added only there. The shared query and key projections turn this offset into a shift of the routing mask's state, while a mask marked for generation remains a token to be predicted. The routing mask's query attends over the tokens in its chunk and pools their keys into a summary. That pool is a softmax over the content keys, and a chunk score adds the entropy of those weights to the agreement with the summary. Each query keeps the highest-scoring chunks. The same scores are then normalized together with the local branch, and the resulting weights assign the attention mass on each selected chunk and on the bidirectional local window. Each of the other layers applies a bidirectional sliding window around the query. Routing positions are omitted from the language-model targets and from the value stream, so the summary is used to choose tokens rather than being predicted or copied into the answer.

Starting from a Dream-v0-Base-7B checkpoint extrapolated to 16K with YaRN, we compare MRSA with NSA, DSA, and SWA under the same layer layout, key budget, and dense initialization, using the same Fast-dLLM decoder at evaluation (Wu et al., 2025b). The benchmarks are LongBench-v1 (Bai et al., 2023) and three RULER probes: single-needle retrieval, multi-key multi-query, and variable tracking (Hsieh et al., 2024). With the YaRN factor scaled to the evaluation length, MRSA's mean accuracy on the three probes is 53.8 at 16K and 47.7 at 128K. NSA, the next sparse model, scores 42.7 and 40.3, so the margin is 11.1 and 7.4 points. DSA and SWA stay near the floor at both lengths. Single-needle accuracy is 100.0 at 16K and 95.0 at 128K, against 81.0 and 83.0 for NSA. Holding the training factor fixed, that single-needle score is 100.0 at 16K and 64K, and 99.0 at 32K and 128K. Component and routing-token ablations are reported in Appendix B and stay out of this comparison. Our contributions are as follows:

- Our core contribution is **MRSA**, a *continued-training sparse attention* method for dLLMs that summarizes each chunk with a native mask token and trains the model to select remote chunks from those summaries.

- We introduce *mask-summary scoring*, which marks that mask as routing rather than generation with a binary type mask and a learned role offset, pools the chunk keys into a summary, and normalizes the scores with the local branch to assign attention mass.

- On three RULER probes, with the YaRN factor scaled to the evaluation length, MRSA leads the other sparse methods on every probe from 16K to 128K. The three-task mean is 11.1 points above NSA at 16K and 7.4 points above NSA at 128K.

2 Related Work

Diffusion language models. Diffusion was first applied to text by corrupting continuous word embeddings and learning to reverse that process (Li et al., 2022; Gong et al., 2023). A second line corrupts discrete tokens. Absorbing-state and masked diffusion replace selected tokens with a mask and train a model to reconstruct them under bidirectional context (Austin et al., 2021; He et al., 2023; Sahoo et al., 2024). This masked formulation has been scaled to LLM size along two routes. One converts a pretrained autoregressive model into a diffusion model by continued training (Gong et al., 2025; Ye et al., 2025). The other trains from scratch: LLaDA is competitive with autoregressive models at the 8B scale (Nie et al., 2026). Later work aligns these models with preferences (Zhu et al., 2025), adds visual instruction tuning (You et al., 2025), and extends the context window (Liu et al., 2025; Peng et al., 2023). Across these extensions, every denoising step still uses dense bidirectional attention over the whole sequence.

Efficient diffusion language models. One line reduces attention cost by training under a restricted mask. Block diffusion denoises one block at a time: tokens inside a block attend bidirectionally, while each block attends only to earlier blocks (Arriola et al., 2025). Fast-dLLM v2 continues training from a pretrained autoregressive model under this mask, and caches completed blocks at inference (Wu et al., 2025a). A second line leaves the dense bidirectional backbone unchanged and accelerates serving. Fast-dLLM reuses key and value states across denoising steps and decodes a block in parallel (Wu et al., 2025b). SparseD builds a head-specific block-sparse mask at inference (Wang et al., 2025), Sparse-dLLM evicts cache entries (Song et al., 2025), and Prefilling-dLLM prefills prefix chunks once, retrieves a top-K subset, and reuses that cache for the remaining steps (Xiong et al., 2026). The trained masks above are fixed in advance, and the serving methods decide visibility only after dense training. Training a bidirectional dLLM to select remote context under a limited attention budget has received less study.

Sparse attention in autoregressive LMs. Autoregressive sparse attention splits into a fixed pattern, a pattern learned in training, and a pattern applied only at inference. Fixed patterns keep each query inside a local window, a stride, or a few global tokens (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Jiang et al., 2023). A distant token is read only when it falls inside that pattern. Learned patterns train the model under a limited budget: NSA mixes compressed, selected, and sliding branches (Yuan et al., 2025), DSA selects individual tokens with a lightweight indexer (DeepSeek-AI, 2025), and MoBA and HiLS route each query to a few chunks (Lu et al., 2025; Hu et al., 2026). Serving-time patterns leave the causal weights unchanged and drop tokens through streaming caches, eviction, or dynamic block masks (Xiao et al., 2024; Zhang et al., 2023; Jiang et al., 2024). These designs assume causal attention and a cache that grows from left to right. Whether the same biases remain useful once attention is bidirectional, with the budget imposed while training a dLLM, is still open.

3 Method

Mask-Routed Sparse Attention (MRSA) is a continued-training sparse attention method for a masked diffusion language model. At one denoising step the model sees a physical sequence $x_{0:L-1}$. A query at position i may attend to valid keys on either side of i. Figure 3 follows the computation on a route layer in three parts: the routing mask that represents a chunk (Section 3.1), the top-K selection of remote chunks (Section 3.2), and the attention mass those scores assign (Section 3.3). The other layers are non-route layers and use sliding-window attention. A local window in this paper is taken around a query and extends to both sides of that query. Its size is a radius: ρ on a non-route layer, and w for the local branch of a route layer. The repeated $3\mathrm{SWA} + 1\mathrm{route}$ layout follows from the bidirectional information flow of a dLLM. A query at an early position can attend to tokens that occur later in the sequence. If the layers between route layers were dense, that remote evidence could be mixed into the hidden states before the next route decision, and the learned route would no longer be the only long-range path. Sliding-window layers keep bidirectional mixing inside the local radius and block direct attention beyond that radius, so long-range evidence enters at the route layers. It can then be carried by the hidden states through the following local layers. The window limits distance rather than imposing a causal direction. The layout is thus both the efficiency constraint and the setting in which the learned route is the long-range path being measured. The layer

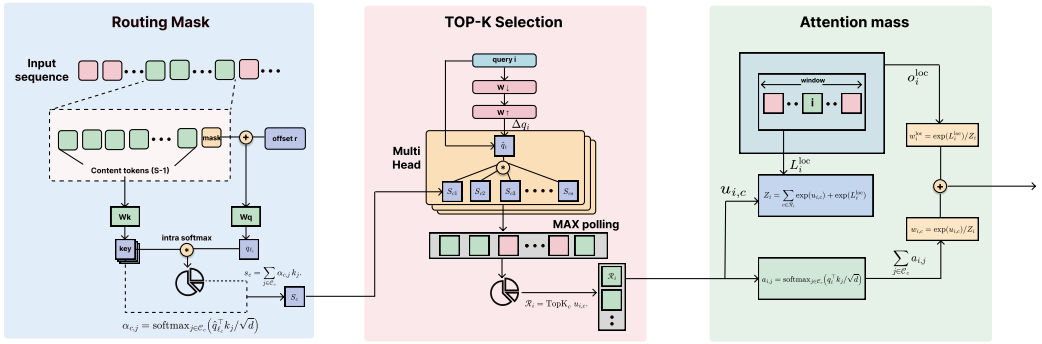

Figure 3: Computation on a route layer. A chunk is $S-1$ content tokens followed by one routing mask. The role offset r is added to that mask at the input. Its query pools the content keys with an intra-chunk softmax, and the weighted sum is the summary key s_c. A content query adds a low-rank residual and scores these summaries. Inside a GQA group the scores are reduced by a maximum, and the top K chunks are shared by every head in the group. Attention mass puts the selected chunk scores and the local-window log-sum-exp in one partition, then a second softmax distributes each chunk's mass over its content tokens.

schedule, the radii, the chunk size, and the number of selected chunks are fixed for the comparison and listed in Appendix A.1.

3.1 Routing mask

Chunk splitting. The sequence is partitioned into contiguous chunks of S positions. Chunk c holds $S-1$ content tokens, followed by one routing position ℓ_c at the end of the chunk. The routing position is filled with the model's native mask id, so the vocabulary is unchanged and the embedding E of the mask is the one the model was already trained with. The same id is also written over content positions that the loss will predict. A binary type mask $m_i \in \{0, 1\}$ separates the two uses. It is stored with the validity mask, and $m_i = 1$ only at the appended routing positions. A mask that occupies a position to be denoised has $m_i = 0$.

Mask role slot. The role offset r is one learned vector in the model hidden dimension, shared by every chunk and every denoising step. It is added once, before the first layer, and only on a routing position: that position still stores the native mask id, and the input of the stack is the mask embedding plus r,

$$e_i = E(x_i) + m_i r. \tag{1}$$

A content position has $m_i = 0$ and keeps $E(x_i)$. No layer adds r again. The first three layers are sliding-window layers. The first route layer is the fourth layer, and the remaining route layers follow every four layers. A route layer does not read e_i. Let h_i be the hidden state that reaches that layer, and let R_i be the rotary map at position i. The backbone query is that state under the shared projection,

$$q_i = R_i(W_Q h_i)\,. \tag{2}$$

The routing position is one index of this same state. Write h_{ℓ_c} for h_i at $i = \ell_c$. It is the state produced by the preceding layers, not the input $E(x_{\ell_c}) + r$. The routing mask's query is q_i at that index, so the substitution $h_i = h_{\ell_c}$ gives

$$q_{\ell_c} = q_i\big|_{i=\ell_c} = R_{\ell_c}(W_Q h_{\ell_c})\,. \tag{3}$$

Content keys are the k_j from the same hidden state. The value path and the token attention inside a selected chunk use these backbone queries and keys. The query that pools the chunk is q_{ℓ_c} after the residual in Section 3.2.

Chunk summary. Let $\mathcal{C}_c$ be the content positions of chunk c. Write $\hat{q}_{\ell_c}$ for that residual query at the routing position. It attends over the content keys and pools them into one key for the chunk,

$$\alpha_{c,j} = \mathrm{softmax}_{j \in \mathcal{C}_c}\left(\hat{q}_{\ell_c}^{\top} k_j / \sqrt{d}\right), \qquad s_c = \sum_{j \in \mathcal{C}_c} \alpha_{c,j}\, k_j. \tag{4}$$

The routing position's own key is excluded from this softmax. s_c is the key of the routing mask. It is a weighted sum of the content keys, not the projection $k_{\ell_c} = R_{\ell_c}(W_K h_{\ell_c})$. Tokens that agree with $\hat{q}_{\ell_c}$ contribute more to s_c, so the summary can concentrate on a few positions inside the chunk. The role offset changes $\hat{q}_{\ell_c}$ through the hidden state at ℓ_c. It is not itself the summary. A generation mask keeps $m_i = 0$, and therefore keeps the mask embedding the denoising loss was trained to read. Routing positions are omitted from the language-model targets. The denoising loss is computed on content positions whose tokens were masked for prediction. The appended mask is present so that $\hat{q}_{\ell_c}$ can pool the chunk. It is not itself a token the model is asked to emit.

3.2 TOP-K SELECTION

Residual query. A low-rank residual is added to the backbone query, including the routing position and every content position,

$$\Delta q_i = W_{\uparrow} W_{\downarrow} h_i, \qquad \hat{q}_i = R_i(\mathrm{RMSNorm}(W_Q h_i + \Delta q_i)). \tag{5}$$

$W_Q h_i$ is the backbone query before the rotary map. $W_{\downarrow}$ maps the model dimension down to a small rank, and $W_{\uparrow}$ maps that rank back to the query dimension. At the routing position this residual is $\hat{q}_{\ell_c}$, the query that forms s_c in Section 3.1. At a content position it is $\hat{q}_i$, and that is the query used in the score below.

Chunk score. Chunk scores do not pool the chunk again. A content position scores the summary s_c of each chunk outside its window of radius w by the inner product of $\hat{q}_i$ with that summary. The score adds a prior, the entropy of the pooling weights inside that chunk,

$$u_{i,c} = \hat{q}_i^{\top} s_c / \sqrt{d} + H(\alpha_c), \qquad H(\alpha_c) = -\sum_{j \in \mathcal{C}_c} \alpha_{c,j} \log \alpha_{c,j}, \tag{6}$$

and there are $(L - 2w)/S$ such summaries. The route keeps the K highest-scoring chunks among them,

$$\mathcal{R}_i = \mathrm{TopK}_c\ u_{i,c}. \tag{7}$$

Chunks inside the local window, chunks without a valid routing mask, and chunks the query is not allowed to read receive no score. The selected set $\mathcal{R}_i$ is the remote half of the route-layer contract. Together with the local window of radius w in Section 3.3, it is the budget under which the route layer is trained. Each query scores $(L - 2w)/S$ summaries and then reads the content tokens of K chunks, so the remote cost scales with the number of chunks outside the window rather than with every token pair.

Grouped max pooling. Dream uses grouped-query attention (GQA): several query heads share one key and one value. Those heads can prefer different chunks, but the selected tokens have to be gathered once because the key is shared. Each query head pools its own weights $\alpha_{c,j}$ and its own summary s_c from the keys of the shared KV head, so the summary is formed once per query head. Inside a GQA group, MRSA takes the maximum of $u_{i,c}$ over those query heads, chooses $\mathcal{R}_i$ from that maximum, and broadcasts the chosen chunks to every head in the group. The gathered values are then read once for the shared key. Each head still receives its own weights on that shared set in Section 3.3.

3.3 ATTENTION MASS

Remote and local mass. Selecting $\mathcal{R}_i$ decides which chunks are visible. It does not yet decide how much attention each visible token receives. The local branch is a bidirectional window of radius w around the query, restricted to content positions. Its attention output is o_i^{loc} and its log-sum-exp is L_i^{loc}. The selected chunk scores and this local log-sum-exp are normalized together,

$$Z_i = \sum_{c \in \mathcal{R}_i} \exp(u_{i,c}) + \exp(L_i^{\mathrm{loc}}), \tag{8}$$

$$w_{i,c} = \exp(u_{i,c})/Z_i, \qquad w_i^{\mathrm{loc}} = \exp(L_i^{\mathrm{loc}})/Z_i. \tag{9}$$

Thus $w_{i,c}$ is the attention mass of chunk c, and w_i^{loc} is the attention mass of the local window. A high summary score increases the share of the budget that chunk receives, and the local window takes the remaining share. The chunk score $u_{i,c}$ is computed with the residual query $\hat{q}_i$. The local log-sum-exp L_i^{loc} is computed with the backbone query q_i, as is the within-chunk softmax below. Putting $u_{i,c}$ and L_i^{loc} in one partition assigns each selected chunk and the local window a share of the attention mass. This is the allocation used by the route. It is not the softmax of one query over every visible token.

Within-chunk attention. Inside a selected chunk, token attention is a separate softmax over the content positions,

$$a_{i,j} = \mathrm{softmax}_{j \in \mathcal{C}_c}\left(q_i^\top k_j/\sqrt{d}\right), \qquad j \in \mathcal{C}_c. \tag{10}$$

The routing position is excluded from this softmax and from the value stream. The summary s_c is used to choose the chunk. It is not a value that can be copied into the output. The route-layer output assembles the two stages,

$$o_i = \sum_{c \in \mathcal{R}_i} w_{i,c} \sum_{j \in \mathcal{C}_c} a_{i,j}\, v_j + w_i^{\mathrm{loc}}\, o_i^{\mathrm{loc}}. \tag{11}$$

For a token j in selected chunk c, the product $w_{i,c}\, a_{i,j}$ is its attention weight. The first factor is the mass of the chunk, computed from the routing-mask summary. The second factor renormalizes the query–key logits inside the chunk. equation 11 puts the scores of Section 3.2 on the forward path. $\mathcal{R}_i$ is a hard top-K, so membership in that set is discrete and receives no pathwise gradient. The denoising loss trains the role offset, the summary, and the low-rank query residual through the masses $w_{i,c}$ of the chunks that were selected.

Non-route layers. On a non-route layer the same query uses only its sliding window of radius ρ, with no chunk summary and no top-K selection. The route layers are where MRSA spends its remote budget. The non-route layers carry local context on both sides of each query.

4 Experiments

4.1 Setup

Continued training starts from Dream-v0-Base-7B. Its original context is 2K, and we extrapolate the checkpoint to 16K with YaRN at a factor of 8. We compare MRSA with bidirectional NSA, DSA, and SWA. MRSA, NSA, and DSA use the same layout, three bidirectional sliding-window layers and one sparse layer, the same key budget, and the same dense initialization. SWA applies a bidirectional sliding window in every layer. The hyperparameters of this comparison are fixed across variants. Appendix A.1 lists the layer schedule, the window radii, and the key budget. Appendix A.2 records the dense extrapolation, and Appendix A.3 records the continued-training objective, the GPUs, and the settings that differ across variants. Appendix A.4 records the Fast-dLLM decoder and the prompts. At evaluation, every variant uses the same Fast-dLLM decoder (Wu et al., 2025b).

Benchmarks. We report LongBench-v1 and three RULER probes (Bai et al., 2023; Hsieh et al., 2024). LongBench keeps each task's official score, and we report those scores and their mean at the 16K training length. RULER inserts the required evidence into a distractor and varies the length. Of its 13 tasks we use single-needle retrieval, multi-key multi-query, and variable tracking, the three families supervised by the RULER term in equation 13. Training and evaluation draw them from that synthesizer. The prompts are in Appendix A.4, lengths above 16K are outside the training length, and the other ten tasks are not reported. An example is correct only when every gold string appears in the prediction. We report each task and their mean at 16K, 32K, 64K, and 128K, with the training YaRN factor held fixed and with the factor scaled to the evaluation length. Each number is one continued-training run. Component and routing-token ablations are in Appendix B.

Table 1: Three RULER probes at 16K, 32K, 64K, and 128K. The upper block is dense attention. Full dense keeps dense attention in every layer through the same number of continued-training steps as the other variants. Hybrid dense uses three sliding-window layers and one dense layer. The lower block is sparse attention. Fixed uses YaRN factor 8. Unfixed sets the factor to the evaluation length divided by 2048. MK is multi-key multi-query. Avg. is the mean of S-N, MK, and VT. In the dense block, $*$ marks the best score in each column. In the sparse block, the best score is bold and the second best is underlined.

		16K				**32K**				**64K**				**128K**			
Model	YaRN	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.
Dense																	
Full dense	*Fixed*	100.0*	100.0*	91.7*	97.2*	100.0*	96.5*	75.7*	90.7*	94.0*	68.5	40.7	67.7*	85.0*	90.0*	34.1*	69.7*
	Unfixed	100.0*	100.0*	91.7*	97.2*	100.0*	99.5*	80.0*	93.2*	100.0*	97.5*	69.3*	88.9*	100.0*	100.0*	62.3*	87.4*
Hybrid dense	*Fixed*	99.0	87.5	67.3	84.6	100.0*	81.5	67.7	83.1	70.0	73.5*	59.3*	67.6	25.0	25.0	27.0	25.7
	Unfixed	99.0	87.5	67.3	84.6	99.0	80.0	69.0	82.7	98.0	81.5	62.7	80.7	85.0	83.5	53.7	74.1
Sparse																	
SWA	*Fixed*	5.0	5.0	4.7	4.9	2.0	1.0	4.3	2.4	3.0	2.5	1.0	2.2	1.0	0.0	0.0	0.3
	Unfixed	5.0	5.0	4.7	4.9	2.0	1.0	4.3	2.4	3.0	2.0	0.7	1.9	2.0	0.0	1.0	1.0
NSA	*Fixed*	81.0	32.0	15.0	42.7	84.0	32.0	10.0	42.0	79.0	16.0	9.3	34.8	79.0	12.0	5.0	32.0
	Unfixed	81.0	32.0	15.0	42.7	87.0	30.5	12.3	43.3	86.0	22.0	9.0	39.0	83.0	29.0	9.0	40.3
DSA	*Fixed*	9.0	15.5	12.7	12.4	6.0	7.0	4.0	5.7	5.0	4.0	2.0	3.7	3.0	0.5	1.3	1.6
	Unfixed	9.0	15.5	12.7	12.4	7.0	6.0	4.7	5.9	3.0	5.0	2.7	3.6	2.0	1.5	2.0	1.8
MRSA	*Fixed*	**100.0**	**42.0**	**19.3**	**53.8**	**99.0**	35.0	**18.0**	50.7	**100.0**	27.0	**17.0**	48.0	**99.0**	32.5	8.0	46.5
	Unfixed	**100.0**	**42.0**	**19.3**	**53.8**	**99.0**	**47.5**	17.7	**54.7**	98.0	**45.0**	14.0	**52.3**	95.0	**37.5**	**10.7**	**47.7**

4.2 Main results

RULER. Table 1 is the length sweep. Among the sparse models, MRSA has the highest three-task mean at every length and under both YaRN settings. With the factor scaled to the evaluation length, the means are 53.8, 54.7, 52.3, and 47.7 at 16K, 32K, 64K, and 128K, against 42.7, 43.3, 39.0, and 40.3 for NSA, a margin of 11.1, 11.4, 13.3, and 7.4 points. DSA and SWA stay near the floor and finish at 1.8 and 1.0. The margin is carried by single-needle retrieval: MRSA scores 100.0, 99.0, 98.0, and 95.0, against 81.0, 87.0, 86.0, and 83.0 for NSA. Multi-key multi-query and variable tracking remain far from full dense, 42.0 and 19.3 against 100.0 and 91.7 at 16K, because those probes scatter the evidence and a top-K route can miss one piece. Holding the factor at 8, MRSA single-needle accuracy stays at 100.0 at 16K and 64K and at 99.0 at 32K and 128K, while full dense falls to 94.0 at 64K. That fixed factor costs multi-key accuracy, from 47.5 to 35.0 at 32K and from 45.0 to 27.0 at 64K, but the 64K mean still favors MRSA, 48.0 against 34.8 for NSA. Hybrid dense replaces the route with a dense layer and scores 84.6 at 16K, against 97.2 for full dense. At 128K its scaled single-needle score is 85.0, below MRSA at 95.0, and the fixed factor drops that score to 25.0.

LongBench. On LongBench the order reverses (Table 2). The task means are 30.9 for SWA, 26.6 for NSA, 25.8 for DSA, and 25.4 for MRSA, against 44.3 for full dense. Most of these tasks need coverage of a long document. A route that keeps a few remote chunks drops most of that document, while SWA applies a wide window in every layer. The same window cannot retrieve a needle, which is why the RULER ranking runs the other way.

4.3 Routing-mask probes

Figure 5 tests the routed mask on 16K single-needle retrieval, 100 examples, under the same Fast-dLLM decoder. Beside the native mask we evaluate two slots trained the same way: the end-of-sequence token, and a new learned token. In the first probe we permute the summary each routing slot computed, across chunks, and leave the text in place. Accuracy falls from 100.0, 94.0, and 99.0 to 12.0 for the mask, the new token, and EOS. Figure 4 shows why. On the clean probe the cosine between a chunk and a routing slot concentrates on the diagonal, so the routed mask agrees with its own chunk. The new token and EOS show the same diagonal. Permuting the summaries removes

Table 2: LongBench-v1. The upper block is dense attention. Full dense keeps dense attention in every layer through the same number of continued-training steps as the other variants. Hybrid dense uses three sliding-window layers and one dense layer. The lower block is sparse attention. Cells without a finished run are left blank. In the dense block, $*$ marks the best score in each column. In the sparse block, the best score is bold and the second best is underlined.

Model	Single-Doc. QA			Multi-Doc. QA			Summarization				Few-shot		Synthetic		Code		Avg.
	NarrativeQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	SAMSum	TREC	TriviaQA	PCount	PRe	LCC	RB-P	
Dense																	
Full dense	29.1*	36.0	53.7*	60.1*	47.1*	38.8	5.1	22.8	4.7*	42.9*	73.5*	91.2*	2.0*	99.5*	47.9*	53.8	44.3*
Hybrid dense	25.4	37.1*	48.5	59.3	40.6	39.2*	8.7*	24.9*	1.5	35.7	72.0	88.7	2.0*	96.3	46.2	55.7*	42.6
Sparse																	
SWA	18.0	**20.8**	26.4	**28.3**	**28.1**	12.8	**19.8**	**20.7**	13.7	**41.9**	**64.0**	**86.2**	1.7	**14.5**	**45.8**	**50.8**	**30.9**
NSA	13.1	16.2	**28.3**	22.6	20.6	10.8	16.2	20.6	17.0	38.9	60.0	79.5	0.3	0.5	38.6	41.9	26.6
DSA	**18.1**	17.3	**28.3**	20.4	22.7	**13.3**	15.2	16.5	14.0	34.2	61.0	74.8	**2.1**	4.4	33.4	36.9	25.8
MRSA	10.4	14.8	17.7	24.3	26.3	10.1	13.4	18.4	**17.2**	35.0	62.0	70.6	0.8	5.6	37.6	42.6	25.4

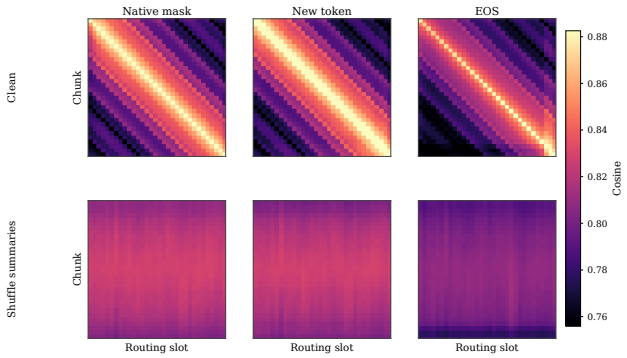

Figure 4: Cosine between each chunk and each routing slot on the 16K single-needle probe, averaged over route layers and 100 examples. Each panel is binned to 32×32. On the clean probe the diagonal pairs a slot with its own chunk. Shuffle summaries permutes those slots across chunks.

it. The needle is still in the text, and breaking the binding between that summary and its chunk is enough to lose it.

Finding. The routed mask stores a chunk summary the model reads. Permuting those summaries across chunks drops 16K single-needle accuracy from 100.0 to 12.0.

In the second probe we move the intact needle into another chunk. The routing slots stay where they are, and only that span is exchanged with the tokens at the destination. Accuracy remains 68.0 for the native mask and falls to 0.0 for the new token and for EOS. The mask summary is formed again from the chunk that now holds the needle, and the route still returns it. A new token and EOS match the mask when the needle stays put, and lose the needle once it moves.

Finding. The routed mask follows a needle after the needle moves to another chunk, at 68.0 accuracy. A new token and EOS fall to 0.0 on the same move.

4.4 Latency

Figure 6 reports Fast-dLLM prefill and decode for full dense, hybrid dense, and MRSA. Each bar is the median of ten trials on one synthetic sequence with 128 masked answer tokens, and the YaRN

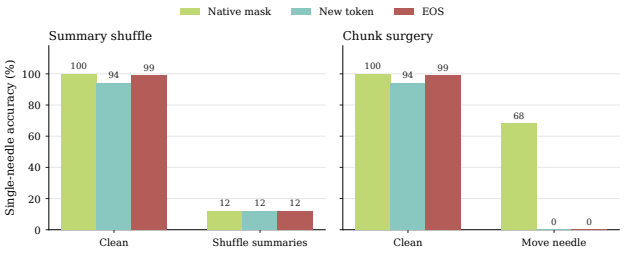

Figure 5: 16K single-needle accuracy for the routed mask, a new learned token, and EOS. Shuffle summaries permutes the summary of each chunk across chunks and leaves the text in place. Move needle places the intact needle in another chunk. Each bar is 100 examples.

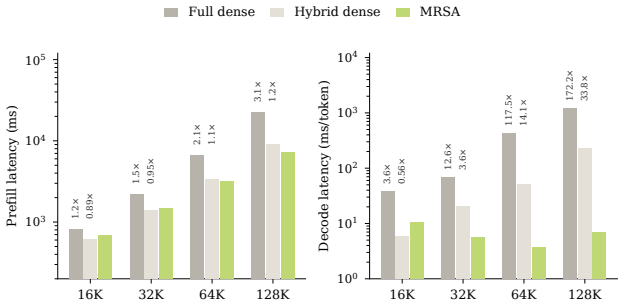

Figure 6: Prefill time and decode time per generated token under the same Fast-dLLM decoder. Both vertical axes are logarithmic. The number above each full-dense and hybrid-dense bar is that latency divided by MRSA. The YaRN factor is the evaluation length divided by 2048.

factor is the evaluation length divided by 2048. NSA, DSA, and SWA are not timed. At 128K, prefill is 22338 ms for full dense, 9034 ms for hybrid dense, and 7280 ms for MRSA, so MRSA is 3.1× below full dense: this pass still reads every token. Decode per token is where the gap opens. Full dense rises from 37.66 ms at 16K to 1190 ms at 128K. Hybrid dense is faster than MRSA at 16K, 5.90 ms against 10.55 ms, then reaches 233.4 ms at 128K, 5.1× below full dense. MRSA stays between 3.69 ms and 10.55 ms per token and is 6.91 ms at 128K, 172× below full dense, because the remote budget is a fixed number of chunks.

Finding. At long context the prefill gain is much smaller than the decode gain. At 128K, MRSA prefill is 3.1× below full dense, while decode per token is 172× below it.

5 Conclusion

MRSA continued-trains a diffusion language model to route with its native mask. A mask at the end of each chunk summarizes that chunk, and every fourth layer keeps a local window plus the top-k chunks scored from that summary. Under a matched budget and the same Fast-dLLM decoder, MRSA leads NSA, DSA, and SWA on all three RULER probes from 16K to 128K when the YaRN factor scales with length. At 128K, decode is 172× faster than full dense, because the remote budget is a fixed number of chunks.

Those three probes are the task families in the continued-training loss, and lengths past 16K are the extrapolation. Multi-key retrieval and variable tracking still trail full dense, because one missed chunk fails an example whose evidence is scattered. On LongBench, where most of a document is useful, a wide window outranks a few selected chunks. The route is the budget that retrieves a localized needle and keeps decode cheap as the context grows.

AI USE

This statement follows the ICLR 2027 AI policy. We used AI writing assistants solely for language polishing and proofreading. All research ideas, experimental design, implementation, and scientific conclusions are entirely the authors' own work.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations*, 2023.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, Kewei Tu, Haitao Mi, and Leo Liang. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, 2024.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Xiaoran Liu, Yuerong Song, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. LongLLaDA: Unlocking long context capabilities in diffusion LLMs. *arXiv preprint arXiv:2506.14429*, 2025.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dLLM: Accelerating diffusion LLMs with dynamic cache eviction. *arXiv preprint arXiv:2508.02558*, 2025.

Zeqing Wang, Gongfan Fang, Xinyin Ma, Xingyi Yang, and Xinchao Wang. SparseD: Sparse attention for diffusion language models. *arXiv preprint arXiv:2509.24014*, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Jing Xiong, Qi Han, Shansan Gong, Yunta Hsieh, Chengyue Wu, Chaofan Tao, Chenyang Zhao, and Ngai Wong. Prefilling-dLLM: Predictive prefilling for long-context inference in diffusion language models. *arXiv preprint arXiv:2606.10537*, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. LLaDA-V: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. BigBird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, 2020.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

Table 3: Layer contracts. Each variant occupies two rows. The first row is every fourth layer, and the second row is the other 21 layers. Local window is a bidirectional radius. Remote budget is the set of keys gathered beyond that window. Total budget is the number of key positions on that row. The sparse rows share a budget of about 2560. A dense layer reads every key.

Variant	Layers	Local window	Remote budget	Total budget
MRSA	every 4th	radius 256	32 chunks of 64	≈ 2560
	other 21	radius 1280	–	≈ 2560
NSA	every 4th	radius 128	32×64 selected, one token per 64	≈ 2560
	other 21	radius 1280	–	≈ 2560
DSA	every 4th	–	token top-k 2560	2560
	other 21	radius 1280	–	≈ 2560
SWA	every 4th	radius 1280	–	≈ 2560
	other 21	radius 1280	–	≈ 2560
Hybrid dense	every 4th	all keys	–	all keys
	other 21	radius 1280	–	≈ 2560
Full dense	every 4th	all keys	–	all keys
	other 21	all keys	–	all keys

A Experimental setup

This appendix records the values used in Section 4.

A.1 Basic setup

These experiments use NVIDIA H20 GPUs. Training and evaluation run in PyTorch 2.9.1 built with CUDA 12.8. The model is Dream-v0-Base-7B, with 28 transformer layers. Its original context is 2048 tokens, and YaRN with factor 8 extrapolates the checkpoint to 16K.

Table 3 records the attention contract. Layers $\{3, 7, 11, 15, 19, 23, 27\}$, every fourth layer, are route layers for MRSA, NSA, and DSA, and dense layers for hybrid dense. Each of the other 21 layers applies its own bidirectional sliding window of radius $\rho = 1280$ around every query, about 2561 positions. Full dense keeps dense attention in every layer, and SWA uses radius 1280 in every layer. On an MRSA route layer the nominal budget is about 2560 keys: a bidirectional window of radius $w = 256$ together with $K = 32$ selected chunks of $S - 1 = 63$ content tokens, before overlap between the window and the selected chunks. Overlap at chunk boundaries can make the number of distinct keys slightly smaller.

A.2 Dense extrapolation

This stage continues Dream-v0-Base-7B with dense attention in every layer. YaRN with factor 8 extends the original context of 2048 to 16384 positions. The run uses two GPUs.

The language-model data are document windows from Dolma 3 Longmino. Each window is one contiguous span of a single document, 16128 tokens long. That length is the text that fills a 16384-position window after one position in every block of 64 is left unused. The corpus has 3200 windows. 128 windows are held out for validation, and the remaining windows form the training set. Documents shorter than one window are dropped.

A training example is a language-model window with probability 0.95 and a RULER window with probability 0.05. On a language-model window, the mask rate t is drawn uniformly from $[0.2, 0.8]$. One view masks eligible tokens at that rate, and the second view masks the remaining eligible tokens, so the two views supervise each of those tokens once. An eligible token is a token whose preceding position is also inside the document and is not an end-of-sequence token. On a RULER window, the same document is the filler. The task is drawn uniformly from single-needle retrieval, multi-key multi-query, and variable tracking, and the loss supervises the answer tokens. The objective is the

Table 4: Dense extrapolation. The run continues Dream-v0-Base-7B under dense attention. The base weights stay frozen, and the listed low-rank adapters are the trainable parameters.

Hyperparameter	Value
GPUs	2
Steps	500
Text tokens per sequence	16128
YaRN factor	8
Optimizer	AdamW
Learning rate	2×10^{-4}
Schedule	cosine
Warmup steps	25
Weight decay	0
Gradient clipping	1.0
Micro-batch size	1
Gradient accumulation	1
LoRA rank	16
LoRA scale	32
LoRA dropout	0.05
LoRA modules	query, key, value, output
Mask rate	$[0.2, 0.8]$
RULER probability	0.05
Seed	7

mean cross-entropy over supervised tokens,

$$\mathcal{L} = \mathrm{mean}[\mathrm{CE}]. \tag{12}$$

The base weights stay frozen. Low-rank adapters of rank 16, scale 32, and dropout 0.05 train the query, key, value, and output projections. Optimization runs for 500 cosine steps at learning rate 2×10^{-4}, with 25 warmup steps, zero weight decay, and gradient clipping at 1.0. Each GPU holds one sequence per step. Table 4 collects these settings.

A.3 Continued training

The six variants in the main comparison continue from the step-500 checkpoint of Section A.2. They use the same Dolma 3 Longmino windows. Every step contains both a language-model view and a RULER view. The language-model view keeps the two complementary masks of that section, with the mask rate drawn uniformly from $[0.2, 0.8]$. The RULER view draws one task uniformly from single-needle retrieval, multi-key multi-query, and variable tracking, and supervises the answer tokens. The two views are averaged as token means,

$$\mathcal{L} = \tfrac{1}{2}\ \mathrm{mean}[\mathrm{CE}_{\mathrm{Dolma}}] + \tfrac{1}{2}\ \mathrm{mean}[\mathrm{CE}_{\mathrm{RULER}}]\,. \tag{13}$$

Each run uses four GPUs, one sequence per GPU, for 1000 cosine steps at learning rate 10^{-4} with 50 warmup steps. The base weights stay frozen, and low-rank adapters of rank 16, scale 32, and dropout 0.05 train the query, key, value, and output projections. Table 5 records the parameters that differ.

NSA. Every fourth layer mixes three branches. The compressed branch maps each non-overlapping block of 64 tokens to one key and one value. The selected branch gathers 32 of those blocks, 64 tokens each, using the attention weights of the compressed branch. The local branch is a bidirectional window of radius 128. A learned gate gives each head three independent sigmoids and mixes the three outputs with those weights. The gate and the compression parameters train together with the low-rank adapters. The other 21 layers use a bidirectional window of radius 1280.

DSA. Every fourth layer retrieves individual tokens. The first 100 steps keep those layers dense. The objective on these steps is a KL divergence that matches the indexer to the dense attention on the tokens in its top 2560, evaluated at every query. Only the indexer is updated, at learning rate 10^{-3}. The indexer has 8 heads of dimension 128, and its weights are trained in full. The remaining

Table 5: Continued training. Each variant is initialized from the dense step-500 checkpoint and trains for 1000 cosine steps on four GPUs, at learning rate 10^{-4} with 50 warmup steps. The base weights stay frozen. Low-rank adapters of rank 16, scale 32, and dropout 0.05 train the query, key, value, and output projections. The rows below are the parameters that differ across variants.

Variant	Hyperparameter	Value
NSA	Route layers	every 4th
	Compressed branch	block 64, stride 64
	Selected branch	32 blocks of 64
	Local branch	radius 128
	Gate	three sigmoids per head
	Other layers	SWA radius 1280
DSA	Route layers	every 4th
	Warmup	100 steps, dense attention
	Warmup objective	indexer KL on every query
	Indexer learning rate	10^{-3}
	Indexer	8 heads, dimension 128
	Sparse phase	900 steps, token top-k 2560
	Sparse KL	weight 1, 64 queries, selected tokens
	Other layers	SWA radius 1280
SWA	Layers	all 28
	Window	radius 1280
MRSA	Route layers	every 4th
	Local window	radius 256
	Remote budget	32 chunks of 64
	Selection	hard top-k
	Role offset	one learned vector, added at the input
	Query residual	rank 64
	Other layers	SWA radius 1280
Hybrid dense	Every 4th layer	dense, all keys
	Other layers	SWA radius 1280
Full dense	Layers	all 28
	Attention	dense, all keys

900 steps switch those layers to token top-k 2560. The language-model term returns, and the KL stays at weight 1, now on 64 queries per sequence and still on the selected tokens. The indexer and the low-rank adapters then share a cosine schedule at 10^{-4} with 50 warmup steps. The other 21 layers use a bidirectional window of radius 1280.

SWA. Every layer uses a bidirectional window of radius 1280. The trainable parameters are the low-rank adapters.

MRSA. Every fourth layer is a route layer, and the other 21 layers use a bidirectional window of radius 1280. Each chunk appends the native mask, and a learned role offset is added to that mask once, at the model input. A rank-64 residual is added to the backbone query before the rotary map. Chunk selection is hard top-k: radius 256 together with 32 chunks of 64. The denoising loss trains the role offset and the residual together with the low-rank adapters. Routing positions are left out of the language-model targets.

Hybrid dense. Every fourth layer reads every key. The other 21 layers use a bidirectional window of radius 1280. The trainable parameters are the low-rank adapters.

Full dense. Every layer reads every key. The trainable parameters are the low-rank adapters.

A.4 EVALUATION

Every variant uses the same Fast-dLLM decoder (Wu et al., 2025b). Key and value states are cached and reused across denoising steps. A route layer also keeps its chunk summaries and refreshes a summary when its chunk changes. Generation fills blocks of 32 positions. Inside a block, every token whose confidence is at least 0.9 is committed, and each step commits at least the most confident token. On RULER the masked span equals the gold answer. On LongBench each task keeps its official generation length.

The two benchmarks use different prompts. A RULER prompt repeats the filler sentence to the target length, inserts the task's needles into that filler, and ends with the question. The single-needle question is below. Multi-key multi-query asks for the codes of two named items, and variable tracking asks for every variable assigned a given value.

RULER.

```
The grass is green.  The sky is blue.  The sun is yellow.  Here we go.
There and back again.
...
|One of the special magic numbers for long-context is:  <code>|
...
What is the special magic number for long-context mentioned in the
provided text?  Answer:
```

LongBench fills the official template of each task. The multi-field English question-answering template is below.

LongBench.

```
Read the following text and answer briefly.

<context>

Now, answer the following question based on the above text, only give me
the answer and do not output any other words.

Question:  <question>
Answer:
```

A.5 TRAINING CURVES

Figure 7 plots the continued-training objective in equation 13 for the six variants on one set of axes. Each line is a trailing average of 25 steps. DSA trains only the indexer for the first 100 steps, so its language-model loss stays near zero until step 101, where the vertical line is drawn.

B ABLATION STUDY

The component ablations keep the native mask and remove one piece of the mainline. No residual drops the rank-64 query residual. No role offset keeps the mask embedding and does not add the learned role offset. Mean summary replaces the attention pool in equation 4 with the mean of the content keys. No entropy removes $H(\alpha_c)$ from the chunk score. These runs use the same continued-training recipe as MRSA. Every row in Table 6 is below MRSA. No role offset is the largest drop. Its means are 11.5, 9.0, 6.0, and 4.8, and single-needle falls to 21.0, 17.0, 15.0, and 13.5. The learned offset is what marks the mask as a routing slot. No residual means are 27.2, 19.1, 12.9, and 6.3. Single-needle falls from 63.0 at 16K to 13.0 at 128K. The rank-64 residual is the query that scores the chunks. No entropy means are 40.7, 38.8, 37.3, and 27.1. Single-needle holds through 64K and falls to 58.0 at 128K, while multi-key stays near 7–13. The entropy term separates chunks that would otherwise look alike. Mean summary is the smallest drop, with means 49.9, 43.3, 40.3, and 35.0 against 53.8, 54.7, 52.3, and 47.7. Single-needle stays at 98.0 through 64K and at 90.0 at 128K. Multi-key falls from 33.5 to 10.9, and variable tracking from 18.2 to 4.2. Averaging the content keys keeps a single needle visible and dilutes the summary once several positions matter.

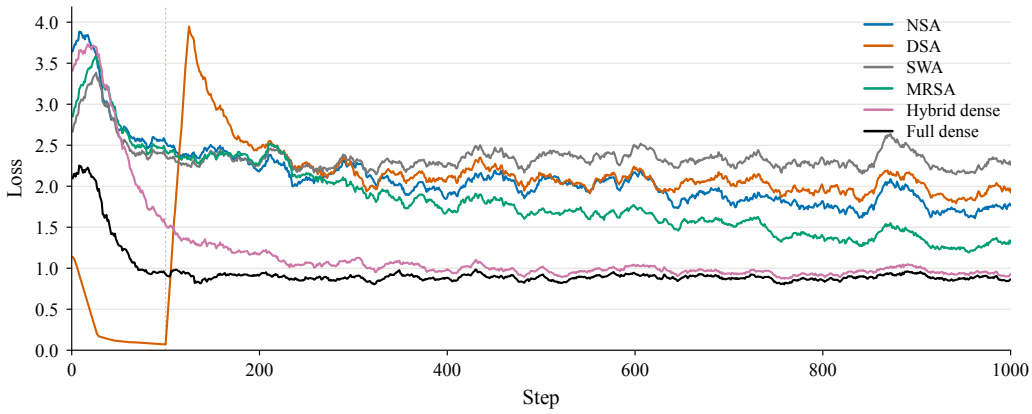

Figure 7: Training loss of the continued-training runs. Each line averages the previous 25 steps. The dashed vertical line marks the end of the DSA indexer warmup.

Table 6: Component ablations on the three RULER probes. The YaRN factor is the evaluation length divided by 2048. MK is multi-key multi-query. Avg. is the mean of S-N, MK, and VT. MRSA is the mainline. Each other row removes one component.

	16K				**32K**				**64K**				**128K**			
Variant	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.
MRSA	100.0	42.0	19.3	53.8	99.0	47.5	17.7	54.7	98.0	45.0	14.0	52.3	95.0	37.5	10.7	47.7
No residual	63.0	12.0	6.6	27.2	47.0	5.5	4.7	19.1	27.0	5.5	6.3	12.9	13.0	1.5	4.3	6.3
No role offset	21.0	11.0	2.4	11.5	17.0	8.3	1.6	9.0	15.0	1.7	1.2	6.0	13.5	0.8	0.1	4.8
Mean summary	98.0	33.5	18.2	49.9	98.0	21.7	10.1	43.3	98.0	16.4	6.5	40.3	90.0	10.9	4.2	35.0
No entropy	98.0	13.0	11.0	40.7	97.0	7.0	12.3	38.8	90.0	10.0	12.0	37.3	58.0	12.5	10.8	27.1

The token ablations replace the routing mask and keep the rest of MRSA. EOS writes the end-of-sequence token into the routing slot. New token uses a learned embedding in place of the native mask. Table 7 is complete for EOS through 64K and for the new token at every length. The 128K EOS run has not started. EOS means are 10.0, 4.6, and 4.1, against 53.8, 54.7, and 52.3 for the native mask. Its single-needle scores are 10.0, 1.0, and 0.0, and its multi-key scores are 9.0, 2.5, and 0.5. Variable tracking stays near 11. New token means are 41.3, 40.1, 40.1, and 31.0. Single-needle stays at 99.0, 95.0, and 92.0 through 64K, then falls to 72.0 at 128K, against 95.0. Multi-key collapses to 9.5, 6.0, 7.5, and 7.0. Variable tracking stays near the mask and is higher at 64K and 128K. An end-of-sequence token does not carry the chunk. A learned token keeps variable tracking and the shorter needle, and still loses multi-key retrieval and the 128K needle. The native mask is the slot that holds both.

Table 7: Routing-token ablations on the three RULER probes. The YaRN factor is the evaluation length divided by 2048. MK is multi-key multi-query. Avg. is the mean of S-N, MK, and VT. Mask is the native mask used by MRSA. EOS and New token replace that mask in the routing slot.

	16K				**32K**				**64K**				**128K**			
Variant	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.	S-N	MK	VT	Avg.
Mask	100.0	42.0	19.3	53.8	99.0	47.5	17.7	54.7	98.0	45.0	14.0	52.3	95.0	37.5	10.7	47.7
EOS	10.0	9.0	11.0	10.0	1.0	2.5	10.3	4.6	0.0	0.5	11.7	4.1	–	–	–	–
New token	99.0	9.5	15.3	41.3	95.0	6.0	19.3	40.1	92.0	7.5	20.7	40.1	72.0	7.0	14.0	31.0