HySparse2: Hybrid Sparse Attention with Two-Level KV Sharing

Anonymous authors
Paper under double-blind review

Abstract

Long-horizon and multi-turn agents typically generate short actions and process long observations from tools and environments. This growing context demands *efficient prefill*, *compact KV-cache storage*, and *accurate long-context retrieval*. To meet these demands, we introduce **HySparse2**, a hybrid sparse attention architecture with **two-level KV sharing**. At the outer level, **KV Bridging** adopts a YOCO-style self-decoder and cross-decoder structure, but bridges only full-attention layers. The self-decoder uses hybrid sliding-window attention (SWA), while the cross-decoder uses hybrid sparse attention. The KV caches for full-attention layers in the cross-decoder are generated from the hidden states of full-attention layers in the self-decoder. At the inner level, HySparse2 retains HySparse's core **KV Reuse** design with two refinements. First, it replaces block-level sparsity with *token-level* sparsity for finer long-context retrieval. Second, it removes the separate SWA branch from sparse layers and instead forces a sliding window of recent tokens into the sparse selection. This two-level KV sharing allows all cross-decoder KV caches to be constructed from self-decoder hidden states. Prefill can therefore exit after the self-decoder, skipping all cross-decoder layers. On an 80B-A3B MoE model, HySparse2 outperforms HySparse and Hybrid SWA on long-context retrieval and multi-turn agentic tasks, while substantially reducing prefill computation and KV-cache storage.

1 Introduction

Agentic inference combines long contexts with multi-turn interaction. Across interaction rounds, a short generated action or tool call can return a much longer search result, execution trace, or document that requires prefill before decoding resumes. As observations accumulate, agents must retrieve and combine evidence across an expanding history, increasing both attention computation and KV-cache storage. These workloads therefore demand efficient prefill, compact KV-cache storage, and accurate long-context retrieval.

HySparse (Gao et al., 2026) addresses long-context efficiency by interleaving full-attention layers with sparse-attention layers. Each full-attention layer supplies both selection indices and a KV cache to the following sparse layers, reducing attention computation and KV-cache storage without distilling an auxiliary indexer module (Gao et al., 2024; DeepSeek-AI et al., 2025). For agentic inference, however, HySparse still leaves substantial room to shorten prefill, further reduce KV-cache storage, and improve retrieval precision.

In this work, we introduce **HySparse2**, which extends HySparse with **two-level KV sharing**. Following YOCO (Sun et al., 2024), the backbone is divided into a self-decoder that combines full attention and sliding-window attention (SWA), and a cross-decoder that combines full attention and sparse attention. At the outer level, **KV Bridging** connects full-attention layers across the two decoders. Each cross-decoder full-attention layer applies its own K/V projections to the input hidden states of a corresponding self-decoder full-attention layer. At the inner level, **KV Reuse** retains HySparse's sharing of KV caches and selection indices within each hybrid block.

HySparse2 also makes two modifications that improve accuracy and efficiency. First, token-level selection replaces block-level selection, allocating the sparse attention budget more precisely to relevant tokens. Second, a forced window of recent tokens replaces the separate SWA branch in sparse layers, allowing local and global tokens to use the same shared KV cache. All cross-decoder

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.

KV caches can then be built from self-decoder hidden states. Prefill can therefore exit after the self-decoder.

We compare HySparse2 with HySparse and Hybrid SWA on 80B-A3B MoE models trained with the same data and schedules. After pretraining, HySparse2 retains broadly comparable general capabilities and improves long-context retrieval. After light post-training, it improves mean MRCR-v2 and RULER-v2 scores over HySparse by 11.30 and 19.81 percentage points, respectively, and achieves lower AgentPPL and LongPPL than both baselines at all evaluated lengths up to 256k. At 1M tokens, our analysis shows 2.92× and 5.02× reductions in prefill FLOPs relative to HySparse and Hybrid SWA, respectively, alongside a smaller KV cache. Ablations show that token-level selection improves retrieval at the same attention budget and that KV Bridging preserves broadly comparable quality.

2 Background and Motivation

2.1 Efficient Prefill

Long-horizon and multi-turn agents make inference increasingly input-dominated. Tool responses can add substantially more tokens than the actions that produced them. Reducing attention cost alone does not eliminate the computation needed to propagate these tokens through the backbone. Cross-layer KV-cache sharing can reduce both KV storage and prefill computation. YOCO uses a self-decoder to construct a global KV cache shared across cross-decoder layers, allowing prefill cache construction to exit after the self-decoder (Sun et al., 2024). Gemma 3n adopts related cross-layer KV sharing to improve prefill efficiency (Sanseviero & Ballantyne, 2025). HySparse applies cross-layer sharing within its hybrid sparse-attention blocks: each full-attention layer shares its global KV cache and top-k block indices with the following sparse-attention layers, though prefill still executes all layers (Gao et al., 2026). HySparse2 further incorporates a YOCO-style structure into HySparse, combining reduced attention computation and KV storage with a shorter prefill path.

2.2 KV Cache Compression

Agentic workloads require more compact KV caches. KV-cache compression spans four dimensions: head, sequence, layer, and precision. Head-level methods share KV heads through GQA/MQA (Ainslie et al., 2023; Shazeer, 2019) or compress KV representations into latent states, as in MLA (Liu et al., 2024a). Sequence-level methods compress multiple tokens into fewer cache entries (DeepSeek-AI et al., 2026). Layer-level methods share KV caches across layers (Brandon et al., 2024; Sun et al., 2024). Precision-level methods reduce the numerical precision of cached keys and values (Liu et al., 2024b; Hooper et al., 2024). Prior work focuses on intra-layer compression along the head, sequence, and precision axes. HySparse2 instead pushes cross-layer sharing through KV Bridging and KV Reuse.

2.3 Sparse Attention Granularity

Sparse attention reduces long-context attention costs by restricting each query to a subset of KV entries (Child et al., 2019). Its selection granularity creates an inherent trade-off between modeling accuracy and inference efficiency. HySparse adopted block-level sparsity as a practical compromise, since evaluations showed no substantial accuracy disadvantage and the regular block structure enabled efficient kernels. Modern agentic workloads, however, involve long-horizon multi-turn trajectories that require precise retrieval from long historical contexts. Under these patterns, block-level selection exhibits a pronounced accuracy disadvantage, while token-level selection retrieves relevant evidence more faithfully. Recent advances in sparse kernels also make token-level implementations practical (Wang et al., 2025). HySparse2 therefore adopts token-level sparsity for more precise context selection.

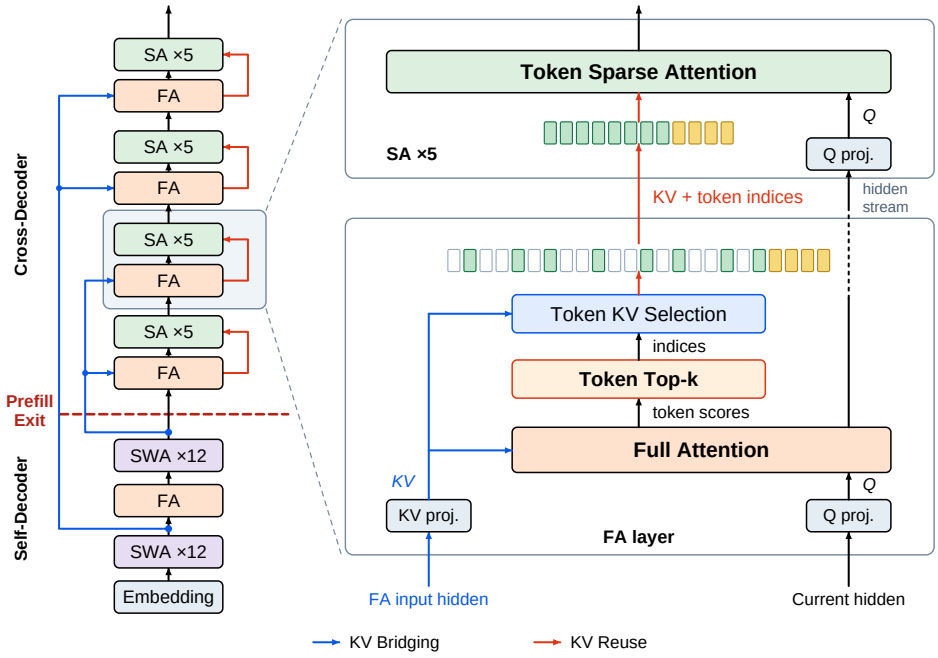


Figure 1: Two-level KV sharing in HySparse2. FA, SWA, and SA denote full attention, sliding-window attention, and sparse attention, respectively. Left: the overall architecture with KV Bridging between the self-decoder and cross-decoder. Prefill cache construction can exit after the self-decoder. Right: a HySparse2 block with token-level sparse attention and a forced local window. Through KV Reuse, SA layers reuse the block's FA KV cache and selection indices.

3 HySparse2

3.1 Overview

Figure 1 illustrates the HySparse2 architecture. Following YOCO (Sun et al., 2024), the backbone is divided into a self-decoder and a cross-decoder. Both decoders use hybrid attention. The self-decoder hybridizes full attention with sliding-window attention for local modeling. The cross-decoder hybridizes full attention with sparse attention for global retrieval. HySparse2 uses two-level KV sharing. At the outer level, **KV Bridging** constructs each cross-decoder full-attention layer's KV cache from corresponding self-decoder hidden states through layer-specific projections. At the inner level, **KV Reuse** lets sparse layers reuse the full-attention layer's KV cache and selection indices within each hybrid block.

3.2 Full-Attention-Only KV Bridging

KV Bridging operates only between full-attention layers in the self-decoder and cross-decoder. Consider a pair of full-attention layers (i, j), with layer i in the self-decoder and layer j in the cross-decoder. Let $\mathbf{H}_i^{\text{self}}$ and $\mathbf{H}_j^{\text{cross}}$ denote the corresponding input hidden states. Cross-decoder layer j computes its keys, values, and queries as

$$\mathbf{K}_j^{\text{cross}} = \text{Proj}_j^K\left(\mathbf{H}_i^{\text{self}}\right), \quad \mathbf{V}_j^{\text{cross}} = \text{Proj}_j^V\left(\mathbf{H}_i^{\text{self}}\right), \quad \mathbf{Q}_j^{\text{cross}} = \text{Proj}_j^Q\left(\mathbf{H}_j^{\text{cross}}\right). \tag{1}$$

Each cross-decoder full-attention layer has its own K/V projections, so layers that share a hidden-state source still construct distinct KV caches. Each layer also retains an independent Q projection applied to its own current hidden states. KV Bridging supports different hybrid attention ratios in the two decoders, allowing one self-decoder full-attention layer to supply multiple cross-decoder full-attention layers, as illustrated in Figure 1.

3.3 CROSS-LAYER KV REUSE

HySparse2 makes two key refinements to HySparse: adopting token-level sparse selection and removing the separate SWA branch.

Token-Level Sparse Selection. In HySparse, pretraining evaluations showed no substantial accuracy disadvantage for block-level selection. However, block-level sparsity shows a pronounced accuracy disadvantage on multi-turn, long-context agentic tasks. HySparse2 therefore applies top-k selection to individual tokens in full-attention layers, and the following sparse layers reuse the KV entries for the selected tokens. Section 4.3 evaluates selection granularity.

Removing the Separate SWA Branch. HySparse2 removes the separate SWA branch used in HySparse and supports local modeling by forcing a recent window into the sparse selection. As illustrated in Figure 1, a local window of the most recent tokens is always selected, followed by the highest-scoring tokens outside the window. Both selected sets are read from the full-attention KV cache and reused by the following sparse-attention layers.

This design also enables a complete early exit after the self-decoder during prefill. A separate SWA branch in the cross-decoder requires building the KV cache for a token suffix from projections of its own hidden states, which further depends on an expanding set of hidden states in earlier layers and grows linearly with depth. This cascading SWA dependency therefore requires the cross-decoder to process a token suffix far longer than the window size. The required states must either be computed during prefill, preventing the cross-decoder from being skipped entirely, or approximated by methods such as bounded replay (DeepSeek-AI, 2026). HySparse2 removes this dependency by construction and requires no cross-decoder computation during prefill. Removing the branch also eliminates substantial parameter overhead from the separate projections in our gated baseline.

3.4 THE ROLE OF FULL ATTENTION

Both HySparse2 and HySparse retain a small number of full-attention layers, as we believe that full attention remains important for model quality. These full-attention layers also serve as indexers, using exact attention scores to provide oracle token selection for subsequent sparse layers. This design supports native end-to-end training without a separate indexer or an auxiliary distillation objective to train one. Full attention nevertheless remains computationally expensive, so we keep the proportion of full-attention layers small. In HySparse2, prefill KV-cache construction requires only one full-attention layer. Following recent sparsification approaches (Gao et al., 2024; DeepSeek-AI et al., 2025), we could further approximate the retained full-attention layers with a lightweight indexer plus sparse attention in post-training. Future architectures could also allocate less computation to full attention and more to sparse attention, for example by reducing the number of query heads in full-attention layers while increasing it in sparse-attention layers.

3.5 INFERENCE ARCHITECTURE

Prefill–Decode Disaggregation. Under prefill–decode disaggregation, the HySparse2 prefill node hosts only the self-decoder and the KV Bridging projections. For the 49-layer model shown in Figure 1, this requires deploying only the first 25 layers, cutting the memory requirement of the prefill node by nearly half. With a high SWA-to-full-attention ratio, the self-decoder performs full attention in only one layer during the prefill stage. We also compute the cross-decoder full-attention KV caches on the prefill node and transfer them to the decode nodes, because the projected KV caches are smaller than the source hidden states.

Speculative Decoding. During pretraining, HySparse2 uses a single MTP layer conditioned on the hidden states at the self-decoder/cross-decoder boundary to aid convergence. During post-training, this MTP layer could be replaced with a larger DFlash-style drafter conditioned on the same hidden states (Chen et al., 2026). Their early availability could support asynchronous drafting and verification (Zhang et al., 2025). Training the larger drafter and coordinating asynchronous execution with verification feedback remain future work.

4 EVALUATION

We compare HySparse2 with HySparse (Gao et al., 2026) and Hybrid SWA used in MiMo-V2 Series (Xiaomi Team et al., 2026). The overall comparison covers model performance, prefill FLOPs, and KV-cache storage. We then conduct ablation studies on sparse selection granularity, the forced local window, and KV Bridging. Token-level selection targets stronger long-context accuracy in agentic scenarios without compromising efficiency, whereas the forced local window and KV Bridging target lower prefill costs while retaining model quality.

4.1 EXPERIMENTAL SETUP

Model configuration. Unless otherwise specified, experiments use 80B-A3B MoE models. Each model has 49 Transformer layers with a hidden size of 2,048 and uses a simplified mHC variant (Zhu et al., 2025; Xie et al., 2026) with the residual mixing matrix fixed to the identity.

The models only differ in their attention designs, as summarized in Table 1. Full attention in every layer is the strongest reference under our evaluation. When choosing how many full-attention layers a hybrid model uses, we avoid a substantial accuracy gap from this baseline. Hybrid SWA retains nine full-attention layers, as using fewer significantly degrades accuracy. HySparse and HySparse2 each use only five full-attention layers while remaining comparable to the full-attention baseline. Hybrid SWA and HySparse use GQA (Ainslie et al., 2023), whereas HySparse2 uses MQA (Shazeer, 2019), which yields a smaller KV cache and better efficiency for token sparse attention kernels.

Table 1: Configurations of the three attention designs.

Model	#Full	Heads (Q/KV)	Head dim. (QK/V)
Hybrid SWA	9	64/4	192/128
HySparse	5	64/4	192/128
HySparse2	5	64/1	256/256

In HySparse2, SWA layers in the self-decoder use partial rotary positional embeddings (RoPE) (Su et al., 2024) with 64 rotary dimensions and a base of 10,000, while full and sparse attention use NoPE. HySparse2 therefore requires no RoPE adjustment during long-context extension. Across the three configurations, all sparse and SWA layers use sigmoid output gates (Qiu et al., 2025) and learnable per-head sink biases (Agarwal et al., 2025). HySparse2 uses token-level selection with 128 forced local tokens and 1,024 global tokens. HySparse selects 1,024 global tokens in 64-token blocks and combines sparse attention with a separate 128-token SWA branch through gated fusion. Hybrid SWA uses a sliding window of 128 tokens.

Training. We pretrain the 80B-A3B models on approximately 500B tokens at a context length of 32k. A light post-training stage adds approximately 100B tokens, introduces agentic data into the training mixture, and extends the context length to 256k. The three models share the same data mixture and training schedule within each stage. We use the Muon optimizer (Jordan et al., 2024) with a WSD schedule and peak learning rates of 10^{-3} for pretraining and 5×10^{-5} for post-training. We report both pretraining and post-training results for overall comparisons, and only pretraining results for ablations.

Evaluation. The pretraining evaluation covers knowledge (MMLU, MMLU-Redux, MMLU-Pro, C-Eval, CMMLU, TriviaQA) (Hendrycks et al., 2020; Gema et al., 2024; Wang et al., 2024; Huang et al., 2023; Li et al., 2023; Joshi et al., 2017), reasoning (BBH, MATH, DROP, GSM8K, ARC-C, HellaSwag, WinoGrande) (Suzgun et al., 2023; Hendrycks et al., 2021; Dua et al., 2019; Cobbe et al., 2021; Clark et al., 2018; Zellers et al., 2019; Sakaguchi et al., 2020), code (HumanEval+, MBPP+, Repo Code PPL) (Chen, 2021; Austin et al., 2021; Liu et al., 2023), and long-context tasks (RULER, NoLiMa) (Hsieh et al., 2024; Modarressi et al., 2025), as listed in Table 2. Repo Code PPL is an internal benchmark that reports byte perplexity over long code repositories with cross-file dependencies, and therefore also probes long-context modeling ability.

After light post-training, we focus on multi-turn retrieval and long-context likelihood on agent trajectories. AgentPPL is an internal benchmark built from multi-turn agent trajectories that measures byte perplexity on the reasoning, tool-call, and response segments of 1,000 trajectories. LongPPL (Fang et al., 2024) measures perplexity on selected tokens that depend on long-range context, using 349 examples from agent trajectories and long-context tasks. MRCR-v2 (Vodrahalli et al., 2024) measures multi-round retrieval with two, four, or eight needles. RULER-v2 (Hsieh et al., 2025) covers 12 retrieval and question-answering subtasks. GraphWalks (OpenAI, 2025) evaluates multi-hop graph traversal.

4.2 Overall Comparison

Table 2: Pretraining performance of the 80B-A3B models. HySparse2 improves long-context performance while remaining broadly comparable to both baselines on general capabilities.

Task	Hybrid SWA	HySparse	HySparse2
Knowledge			
MMLU	61.48	**64.48**	63.92
MMLU-Redux	63.96	**68.44**	67.50
C-Eval	65.08	67.09	**67.90**
CMMLU	67.41	**69.97**	69.90
TriviaQA	60.59	**60.67**	59.67
Reasoning			
BBH	60.14	61.93	**64.29**
MMLU-Pro	36.12	35.74	**37.56**
MATH	**36.18**	36.14	34.32
DROP	60.90	**63.78**	58.99
GSM8K	**64.44**	64.14	61.94
ARC-C	75.51	**79.18**	77.82
HellaSwag	79.38	79.19	**79.54**
WinoGrande	73.72	**74.19**	71.82
Code			
HumanEval+	**35.98**	34.76	34.15
MBPP+	**55.56**	52.65	52.65
Repo Code PPL ↓	1.1578	1.1588	**1.1570**
Long context			
RULER	88.71	84.89	**90.77**
NoLiMa	30.13	40.27	**49.76**

Pretraining results. Table 2 shows that HySparse2 remains broadly comparable to both baselines on general capabilities, while its clearest gains are in long-context modeling. HySparse2 achieves the highest RULER and NoLiMa scores, improving over HySparse by 5.88 and 9.49 points, respectively. It also has the lowest Repo Code PPL at 1.1570, a small improvement over 1.1588 for HySparse and 1.1578 for Hybrid SWA. Individual general-purpose results show a mixed picture. HySparse2 is stronger on BBH and MMLU-Pro, while HySparse retains an advantage on DROP.

Agentic and long-context results. We next compare the models after the same light post-training stage of approximately 100B tokens. We evaluate retrieval with MRCR-v2 and RULER-v2 and long-context likelihood with AgentPPL and LongPPL. Retrieval averages weight the reported context lengths equally, and perplexity curves group examples by context length.

Figure 2 shows that HySparse2 leads both baselines on all four metrics at every evaluated length. The largest gains are in retrieval. Relative to HySparse, its mean MRCR-v2 and RULER-v2 scores increase by 11.30 and 19.81 points, while the corresponding gains over Hybrid SWA are 6.44 and 18.65 points. The advantage remains large at 256k, where HySparse2 reaches 58.45 on RULER-v2, compared with 32.61 for HySparse and 35.74 for Hybrid SWA. HySparse2 also achieves lower AgentPPL and LongPPL throughout the evaluated range, showing that its retrieval gains are accompanied by better likelihood on long agent trajectories and context-dependent tokens. AgentPPL increases with context length, whereas LongPPL decreases. This contrast reflects opposite effects of additional context: longer contexts expose more relevant information for LongPPL's key tokens,

but also introduce more intervening turns and tool outputs that make the relevant evidence harder to retrieve in multi-turn agent trajectories.

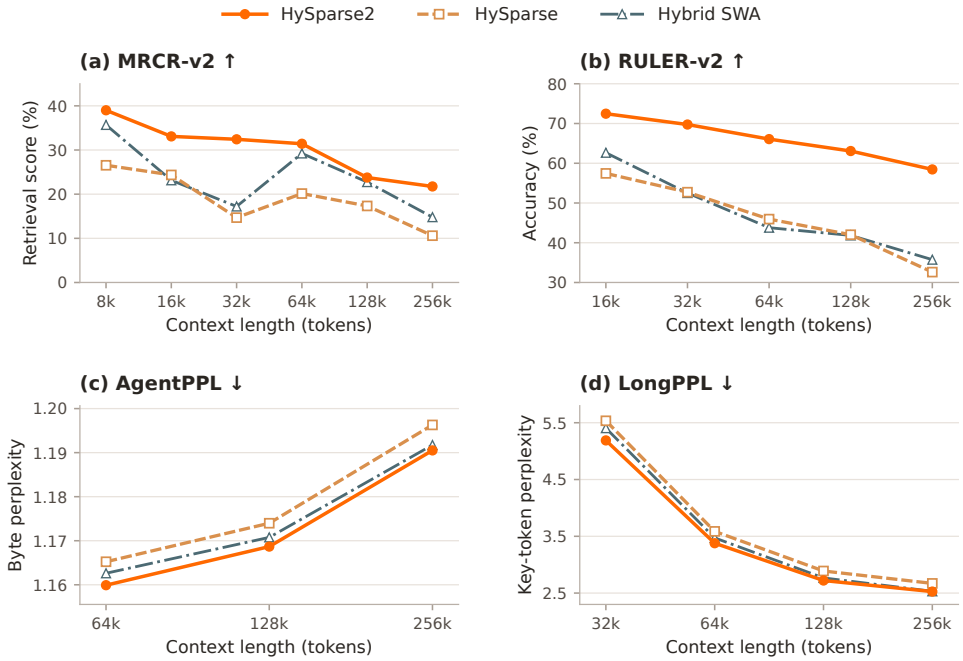


Figure 2: Long-context performance after a light post-training stage. HySparse2 achieves higher retrieval scores and lower perplexity than both baselines across all evaluated context lengths.

Prefill computation and KV-cache storage. We compare the prefill FLOPs and KV-cache size of HySparse2, HySparse, and Hybrid SWA, under the 80B-A3B configuration across context lengths from 1k to 1M tokens with FP8 KV-cache storage.

Figure 3 shows that HySparse2 requires less prefill computation and KV-cache storage than HySparse and Hybrid SWA. At 1M tokens, HySparse2 reduces prefill FLOPs by 2.92× relative to HySparse and 5.02× relative to Hybrid SWA. The KV cache of HySparse2 occupies only 2.69 GB, compared with 6.72 GB for HySparse and 12.09 GB for Hybrid SWA.

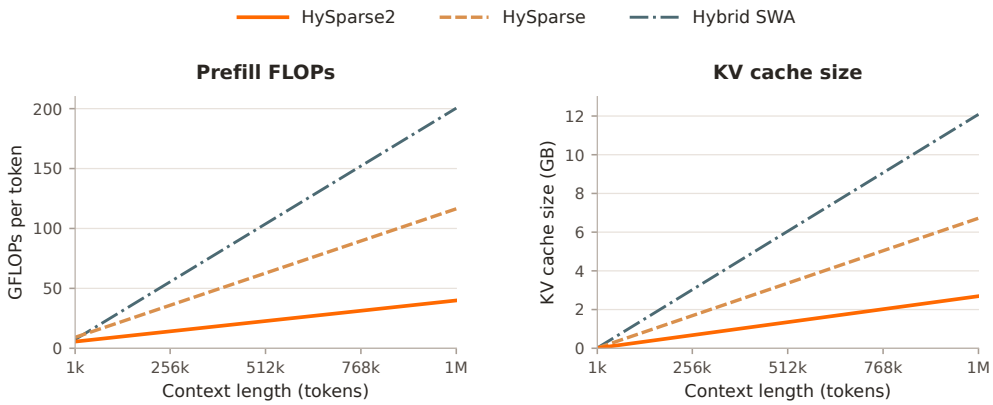


Figure 3: Prefill computation and KV-cache storage for the 80B-A3B models. HySparse2 reduces both costs relative to HySparse and Hybrid SWA.

4.3 Ablation 1: Token-Level versus Block-Level Sparsity

We compare block-level selection with token-level selection. Block-level selection uses a block size of 64. Both variants use the same backbone layout, select 1,024 global tokens, and retain a separate 128-token local window.

Table 3: Token-level versus block-level sparse selection after pretraining. Token-level selection improves long-context retrieval and graph reasoning under the same attention budget.

Task	Block	Token
Pretraining		
BBH	**61.93**	60.70
MMLU-Pro	35.74	**36.97**
NoLiMa	**40.27**	38.43
Long context ($\leq$32k)		
RULER-v2	49.56	**56.13**
MRCR-v2 (2-needle)	12.94	**21.08**
GraphWalks	29.38	**34.92**

Table 3 shows clear gains for token-level selection on long-context retrieval and graph reasoning under the same attention budget. It improves RULER-v2 by 6.57 points, two-needle MRCR-v2 by 8.14 points, and GraphWalks by 5.55 points. These gains are present within the 32k training context.

Agent trajectories repeatedly interleave reasoning, tool calls, and returned observations. The information needed for a response can be spread across several turns, along with role delimiters and other special tokens that identify the relevant boundaries. Under a fixed attention budget, selecting a block to retain one such token also spends capacity on its neighbors. Token-level selection can allocate that budget to individual positions across the context. The retrieval and graph-reasoning gains are consistent with this explanation.

4.4 Ablation 2: Local Window in Sparse Layers

We compare three ways to handle local context inside sparse layers: a separate gated 128-token SWA branch (**Gated SWA**), removing the separate SWA branch (**No SWA**), and forcing the most recent 128 tokens into the sparse selection (**Forced SWA**). All three share the same model backbone, KV Bridging, and hybrid SWA self-decoder. They differ only in how the cross-decoder's sparse attention handles local context. We evaluate pretraining results within their 32k training context.

Table 4: Local-attention ablation in sparse layers. Forced SWA remains competitive with Gated SWA.

Reasoning				*Long context ($\leq$32k)*			
Task	**Gated SWA**	**No SWA**	**Forced SWA**	**Task**	**Gated SWA**	**No SWA**	**Forced SWA**
BBH	62.23	60.11	**62.25**	NoLiMa	37.62	**38.37**	36.16
MMLU-Pro	**39.15**	37.19	38.12	RULER	88.19	84.55	**89.84**
MATH	**35.82**	33.82	33.38	RULER-v2	53.66	54.62	**55.98**
DROP	**60.06**	58.94	56.85	MRCR-v2	**27.66**	20.73	22.67
GSM8K	**64.52**	60.35	59.44	GraphWalks	35.39	36.48	**37.13**
ARC-C	**81.06**	79.01	78.41	LongPPL $\downarrow$	**6.8807**	7.1307	6.9838
HellaSwag	**79.85**	78.23	78.42				
WinoGrande	73.09	**73.95**	72.85				

Table 4 shows that local information is necessary for sparse attention. Removing the local branch (No SWA) generally degrades model performance, while Forced SWA remains competitive with Gated SWA on several tasks. Forced SWA achieves the best RULER, RULER-v2, and GraphWalks scores. Compared with Gated SWA, Forced SWA is 5.08 points lower on GSM8K and 4.99 points lower on MRCR-v2, while LongPPL is 1.50% higher. We consider this an acceptable trade-off, as Forced SWA avoids additional projection parameters and requires no local KV cache, thereby making early-exit prefill feasible.

4.5 Ablation 3: KV Bridging

With and without KV Bridging. KV Bridging is designed primarily to accelerate prefill while preserving model quality. To test whether this design scales, we train 290B-A8B models on approximately 1.8T tokens at a 32k context length, with and without KV Bridging.

Table 5: KV Bridging ablation at the 290B-A8B scale. Models with and without KV Bridging achieve broadly comparable quality.

Task	w/o bridging	w/ bridging	Task	w/o bridging	w/ bridging
MMLU	72.68	**72.80**	GSM8K	**77.63**	76.65
TriviaQA	73.32	**74.10**	Repo Code PPL ↓	**1.1351**	1.1353
BBH	**70.65**	69.57	RULER	**96.32**	96.01
DROP	**71.37**	68.17	LongPPL ↓	3.6053	**3.4202**

Table 5 shows that KV Bridging preserves quality comparable to the baseline without it. MMLU and TriviaQA improve slightly, RULER changes by only 0.31 points, and Repo Code PPL is nearly unchanged. LongPPL improves from 3.6053 to 3.4202. BBH and GSM8K decrease by about one point, and DROP falls from 71.37 to 68.17.

KV Bridging versus KV Mirror. For the connection scheme, we compare KV Bridging with KV Mirror (Liu et al., 2026) on 80B-A3B models that share the same hybrid backbone and pretraining recipe. KV Mirror uses U-shaped connections that pair early and late layers in reverse order, $1 \to n$, $2 \to n-1$, and so on, whereas KV Bridging connects only the full-attention layers in the self-decoder and the cross-decoder. Both styles form the target KV representations by re-projecting source hidden states.

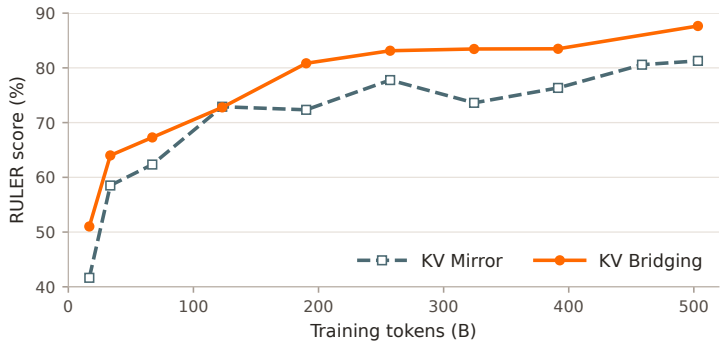


Figure 4: KV Bridging versus KV Mirror during pretraining. KV Bridging achieves higher RULER scores.

Figure 4 shows that KV Bridging is stronger for most of training and finishes at 87.65, compared with 81.28 for KV Mirror. We hypothesize that full-layer states are better sources because full attention backpropagates through scores across the entire visible context, while SWA restricts these direct connections to a local window. This denser supervision produces representations that retain more global information for projection.

5 Conclusion

We present HySparse2, a hybrid sparse attention architecture designed for long-horizon, multi-turn agentic workloads. HySparse2 is a more efficient successor to HySparse and Hybrid SWA, with better long-context and multi-turn retrieval, faster prefill, and lower KV-cache storage. Its two-level KV sharing enables prefill KV-cache construction by running only half of the model, including just one full-attention layer. Under prefill–decode disaggregation, the prefill node therefore needs to host only half of the model weights.

AI USE STATEMENT

We used OpenAI Codex to assist with adapting the LaTeX source to the conference template, checking formatting and anonymity, organizing supplementary material, and drafting the submission statements. The authors are responsible for the final content of this work, including its text, claims, and artifacts.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 86927–86957, 2024. doi: 10.52202/079017-2758. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9e23d020c18e4c40d81c6a0fc7a46f68-Abstract-Conference.html.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026. URL https://arxiv.org/abs/2602.06036.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026. Technical report.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. doi: 10.48550/arXiv.2606.19348. URL https://arxiv.org/abs/2606.19348.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 2368–2378, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? *arXiv preprint arXiv:2410.23771*, 2024.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning intrinsic sparse attention in your LLMs. *arXiv preprint arXiv:2410.13276*, 2024. URL https://arxiv.org/abs/2410.13276.

Yizhao Gao, Jianyu Wei, Qihao Zhang, Yu Cheng, Shimao Chen, Zhengju Tang, Zihan Jiang, Yifan Song, Hailin Zhang, Liang Zhao, Bo Yang, Gang Wang, Shijie Cao, and Fuli Luo. Hysparse: A hybrid sparse attention architecture with oracle token selection and kv cache sharing. *arXiv preprint arXiv:2602.03560*, 2026.

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, et al. Are we done with mmlu? *ArXiv preprint*, abs/2406.04127, 2024. URL https://arxiv.org/abs/2406.04127.

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

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

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303, 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/028fcbcf85435d39a40c4d61b42c99a4-Abstract-Conference.html.

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.

Cheng-Ping Hsieh, Faisal Ladhak, Krishna C Puvvada, and Boris Ginsburg. RULERv2: From basic retrieval to complex reasoning, a bottom-up benchmark for long-context evaluation. In *39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025. URL https://openreview.net/forum?id=ZU9tRffRSA.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c6ec1844bec96d6d32ae95ae694e23d8-Abstract-Datasets_and_Benchmarks.html.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. *ArXiv preprint*, abs/2306.09212, 2023. URL https://arxiv.org/abs/2306.09212.

Aiwei Liu, Cheng Shi, Chuhan Wu, et al. Hidden decoding at scale: Latent computation scaling for large language models. *arXiv preprint arXiv:2607.08186*, 2026. URL https://arxiv.org/abs/2607.08186.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a. URL https://arxiv.org/abs/2405.04434.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024b. URL https://proceedings.mlr.press/v235/liu24bz.html.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A Rossi, Seunghyun Yoon, and Hinrich Schütze. Nolima: Long-context evaluation beyond literal matching. *arXiv preprint arXiv:2502.05167*, 2025.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, 2025. Introduces OpenAI-MRCR and Graphwalks.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

Omar Sanseviero and Ian Ballantyne. Introducing Gemma 3n: The developer guide. https://developers.googleblog.com/en/introducing-gemma-3n-developer-guide/, June 2025.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

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

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, Orhan Firat, Angeliki Lazaridou, Jean-Baptiste Lespiau, Nithya Attaluri, and Kate Olszewska. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*, 2024. arXiv:2409.12640v2.

Lei Wang, Yu Cheng, Yining Shi, Zhengju Tang, Zhiwen Mo, Wenhao Xie, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. TileLang: A composable tiled programming model for AI systems, 2025. URL https://arxiv.org/abs/2504.17577.

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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Xiaomi Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=mDhyxu8WRb.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Ziyi Zhang, Ziheng Jiang, Chengquan Jiang, Menghan Yu, Size Zheng, Haibin Lin, Henry Hoffmann, and Xin Liu. SwiftSpec: Ultra-low latency LLM decoding by scaling asynchronous speculative decoding. *arXiv preprint arXiv:2506.11309*, 2025. URL https://arxiv.org/abs/2506.11309.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-Connections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9FqARW7dwB.