FOLDABLE MAPS ENABLE EXPRESSIVE AND EFFICIENT ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference is bound by the KV cache. Linear attention on its own struggles with precise recall while dense attention has higher precision but becomes extremely expensive for long sequences. Shared latent compression (MLA) improves decode bandwidth by reading once across all heads, but that leaves no benefit for sparse head access. In this paper we introduce Foldable Maps, a conditional linear map system which pairs a shared latent with per-expert latents that utilize a foldable workspace. The expert-level latent expands using 6 of 24 map projections per-expert per-token, which simplifies to one once selected. This enables the training-time workspace to fold with absorption, making the per-expert cost independent of workspace and head width for inference. At 815M parameters with token-matched arms and a 4.57x KV compression ratio over its parent model, this format matches or exceeds both uncompressed baselines on NIAH retrieval averaged over 4k-100k contexts (+20% over parent) and more than doubles the score of the next best measured compression format at equal latent width. The perplexity is the best of the measured compressed formats, but is still $\sim$5% worse than the uncompressed parent at this compression ratio. We also train a 24-byte per-token per-layer indexer which maintains the full NIAH retrieval edge (0.423 vs 0.426) with $\sim$12x less read traffic compared to the parent model at 100k context length. Partitioning by consumer makes expert count a free axis for attention capacity, aligning the training and serving benefits of attention-side MoE.

1 INTRODUCTION

Attention efficiency in long contexts is one of the main bottlenecks in Transformer language models (Vaswani et al., 2023) and many angles on how to improve it have been explored. Training and inference efficiency have very different dynamics due to differing interactions with modern GPU architecture. For decode, memory accesses are by far the largest bottleneck (Shazeer, 2019), which is why methods like Multi-Head Latent Attention (MLA) can result in dramatic speedups despite increasing floating-point operations (FLOPs) (DeepSeek-AI et al., 2024). Training sparsity requires heavy consideration for attention mixing, materialization and kernel density to fully utilize tensor cores (Yuan et al., 2025), so it is common for large disparities to exist between FLOP reductions and wall clock speedups (Dao et al., 2022). Mixture of Sparse Attention (MoSA) applies Mixture of Experts (MoE) style routing to the QKV matrices so tensor operations can be compacted, considerably improving training speed (Piękos et al., 2025). However, it is not as naturally composable with MLA unless the scheme is compute bound since fewer active heads do not reduce memory accesses at decode for a single layer-wide latent.

When it comes to long contexts in particular, the key value (KV) cache and $O(n^2)$ nature of dense attention (Bahdanau et al., 2016) cause the mixing stage to dominate total compute time (Nawrot et al., 2026). MLA massively reduces the KV cache and the total amount of memory accessed, but does not eliminate the quadratic nature of the comparison. MoSA reduces the quadratic dimension through expert selection while methods like Native Sparse Attention (NSA) create blocks of tokens at the cost of per token fidelity during selection (Yuan et al., 2025). Per token pruning granularity can usually not be taken full advantage of during training (Shi et al., 2025) but is effective in reducing the number of full fidelity memory accesses at the cost of occasional misses (Chen et al., 2024). Methods like Hash Attention with Hamming space encoding (Desai et al., 2025), Quest with KV page selection (Tang et al., 2024) and more recently the Lightning Indexer in Deepseek-V3.2

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.

(DeepSeek-AI et al., 2025) all tackle this problem from different angles, but the shared goal is to cheaply find the relevant tokens for each query. Linear attention methods on their own lack the precision of quadratic attention (Arora et al., 2025), but recent models like Kimi K3 (Team et al., 2026) have found success mixing linear and quadratic layers to reap compute and memory gains without a significant reduction in retrieval performance (Team et al., 2025).

2 Model Architectures

To enable robust comparisons, we trained three different classes of model. A Grouped Query Attention (GQA) model serves as the dense baseline (Ainslie et al., 2023) and a sparse model that alternates linear and sparse layers serves as the base for our experiments. From the sparse base, we train five models with different kv latent compressions.

2.1 Architecture Details

To maintain a fair comparison, all models in this paper are trained on 10 billion pretraining tokens at 8k context length then 1 billion at 100k context length, all in identical order. The number, size, and placement of Feed Forward Network (FFN) layers are identical as all changes occur in the attention portions of the network.

The dense GQA baseline has 25 attention layers with 6 stored kv heads of size 64 and 24 q heads of size 64 per layer. Standard Rotary Positional Embeddings (RoPE) (Su et al., 2023) are applied on all heads. The uncompressed version of our custom model alternates linear and sparse layers in a 1:1 ratio, chosen over the standard 1:3 (Team et al., 2025) due to a combination of small model size and a desire to demonstrate retrieval capabilities that are best served with the sparse layer.

The sparse layer uses 8 qkv RoPE heads with a sliding window of 256 and heads of size 64 in combination with 4/16 active routed expert No Positional Encoding (NoPe) heads (Kazemnejad et al., 2023). Router feedback is sourced from value-scaling plus a small balance term. We also apply Scalable-Softmax (SSMax) (Nakanishi, 2025) and include 4 trainable anchors per NoPe head. Although the KV side is sparse, we kept the query side dense to ensure every token is visible to four queries per-token per-layer. Since chosen experts are not expanded like GQA, effective KV is 1/3 the baseline since there are 4 instead of 6 heads at equal sizes for half as many layers. However, the attention mixing operation is meaningfully faster as effective sequence length is reduced to T * (routed / total experts) or T/4 in this setup. Training speed is only marginally faster during 8k pretraining ($\sim$10% faster) but this expands to $\sim$200% faster for the 100k finetune where attention mixing dominates.

2.2 Foldable Maps

To bridge the gap to sensible attention MoE, some modification to the standard layer-wide latent of MLA was required. By utilizing a workspace that simplifies at inference, Foldable Maps adds expressivity to expert-level latents. In our setup, we use a shared latent of size 48 and an expert-level latent of size 16 for each active kv pair. This corresponds to a single shared latent of size 112 given 4/16 active routing. Each map router receives the same value-scaling gradient for feedback as the expert router it is associated with, plus their own small balance term.

For each expert, we train 24 projections to size 128 of which 6 are active at each pass. The projections are summed, concatenated with the shared latent, then compressed down to head size. The comparison to a generic combination of shared and expert latents can be seen in Figure 1.

A set of linear projections of the same dimensionality collapses to a single linear projection. So why use this set of maps with a workspace rather than a single expansion? The key is that only a subset of the maps are active for any token, so the total number of combinations is $\binom{24}{6} = \frac{24!}{6!\,18!} = 134{,}596$. So while the resulting dimensionality in the workspace is unchanged, the variety allows different subspaces to be addressable in a learned fashion. One dimensional lines in a two dimensional space are shown in Figure 2 to illustrate this point. Tracking which maps are active requires a few extra bytes in the KV cache, but the associated compression and expansion is much more versatile as a result. It should also be noted that since the workspace folds at decode, it does not add any expressivity and is instead used solely for optimization purposes.

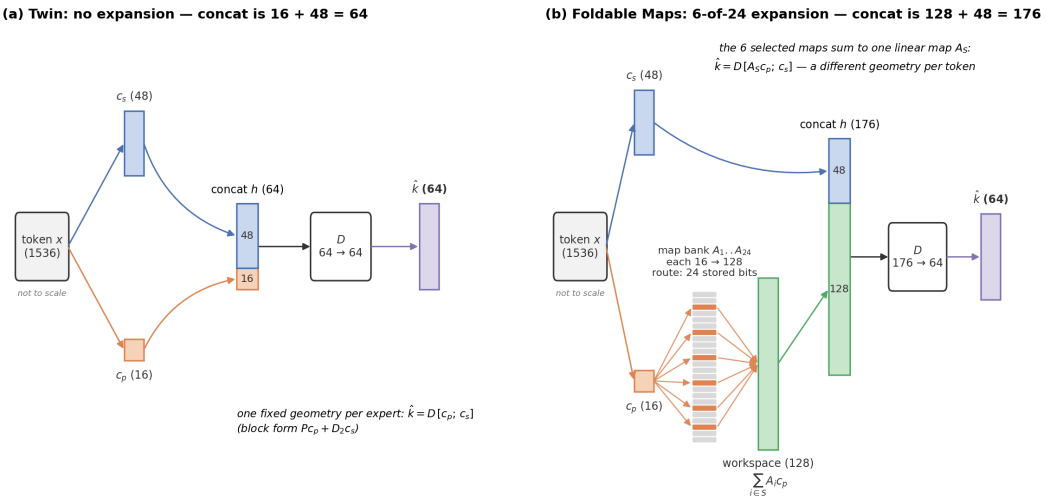

Figure 1: KV compression with and without foldable maps. Equal latent width, different usage.

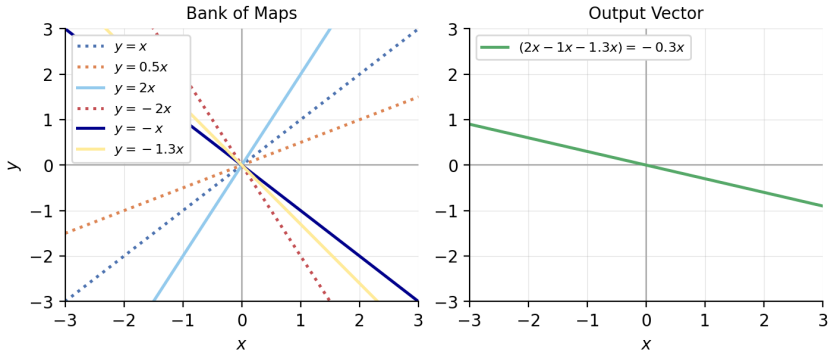

Figure 2: Collection of 1d projections allows a greater variety of representations in 2d space depending on which are active (solid lines). Actual implementation is 16 dims projecting to 128.

2.3 Map Folding

What prevents this mechanism from being costly at inference is the same absorption property that enables MLA. Although there are 134,596 combinations per head in our setup, only the total number of generators (24) have to be precomputed to enable the full range of outcomes without materialization. The math operations are shown below, descriptions for each variable can be found in the appendix.

Full operation set

$$z = \sum_{i \in S} A_i \, c_p, \qquad A_i \in \mathbb{R}^{128 \times 16},\ c_p \in \mathbb{R}^{16},\ z \in \mathbb{R}^{128} \tag{1}$$

$$h = \begin{bmatrix} z \\ c_s \end{bmatrix}, \qquad c_s \in \mathbb{R}^{48},\ h \in \mathbb{R}^{176} \tag{2}$$

$$\hat{k} = D\,h, \qquad D \in \mathbb{R}^{64 \times 176},\ \hat{k} \in \mathbb{R}^{64} \tag{3}$$

$$s = q^\top \hat{k}, \qquad q \in \mathbb{R}^{64} \tag{4}$$

Collapse the set

$$D = \begin{bmatrix} D_w & D_s \end{bmatrix}, \qquad D_w \in \mathbb{R}^{64\times 128},\ D_s \in \mathbb{R}^{64\times 48} \tag{5}$$

$$A_S := \sum_{i\in S} A_i, \qquad A_S \in \mathbb{R}^{128\times 16} \tag{6}$$

$$s = q^\top D_w A_S\, c_p \ +\ q^\top D_s\, c_s \tag{7}$$

Reorder and absorb

$$s = \sum_{i\in S} \left(A_i^\top D_w^\top q\right)^\top c_p \ +\ \left(D_s^\top q\right)^\top c_s \tag{8}$$

$$\widehat{W}_i := D_w A_i \in \mathbb{R}^{64\times 16}, \qquad \tilde{q}_i := \widehat{W}_i^\top q \in \mathbb{R}^{16}, \qquad \tilde{q}_s := D_s^\top q \in \mathbb{R}^{48} \tag{9}$$

$$s = \left(\sum_{i\in S} \tilde{q}_i\right)^\top c_p \ +\ \tilde{q}_s^\top c_s \tag{10}$$

Thus, the query interactions at inference are dependent on the private and shared latent widths, not on workspace or head size. A visual representation can be seen in Figure 3. The biggest persistent cost is the 6 bytes of routing mask per latent (24 bits each for k and v).

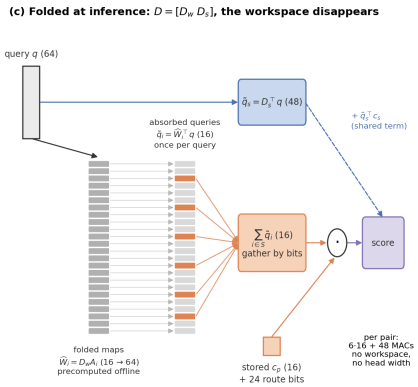

Figure 3: Workspace folds once maps are selected at decode. This allows improved optimization during training without making inference expensive.

2.4 Concentration Aux

The intended purpose of NoPe heads in this setup is content retrieval, but the dynamics of 8k pre-training can make other, more diffuse functions beneficial. However, with our intent to prune excess tokens at runtime, a diffuse structure causes even a perfect selector to make sacrifices. To encourage none of the NoPe heads in the model to develop in this manner, we apply an auxiliary loss starting 40% into training to encourage concentration into a prunable subset of tokens.

Table 1: Cost of restricting routed attention to the top-k keys, at 102400 tokens. Perplexity is measured under a small subset that is identical between models and should not be interpreted as average model performance.

		Pruned with Oracle	
Model	Full	k=1024	k=2048
sparse	31.674	+0.231	+0.180
hybrid_maps	33.834	+0.019	+0.010
mla	35.854	+2.617	+2.032

To generate the concentration aux term, we probe a random subset of 64 queries per layer per step and record their value-weighted attention mass. Any mass outside of the top 1% of tokens

visible to that slot (minimum 32) is penalized. A branch of the hybrid's training run at the point concentration aux was introduced (step 8000 of 19074) was run without concentration aux until step 12000. In this window, the branch without concentration aux had lower CE loss by a consistent 0.002 margin, suggesting the impact for a full run would be a very mild negative, though the exact difference is still uncertain. Table 1 demonstrates the different performance dropoffs between the sparse, MLA and hybrid models. When considering that MLA sees the full sequence of length T compared to T/4 for hybrid and sparse, its proportionally larger falloff is not surprising. Despite neither model ever having actually been trained on a pruned subset of sequences, the hybrid model that received concentration aux has roughly a 20x smaller penalty than the parent model without concentration aux. This shows that simulated pressure should be sufficient for pruning preparation, avoiding possible expense or complications from a direct training solution.

2.5 Importance with a Glance

To maximize decode efficiency, we trained a lightweight, per-head indexer similar to DeepSeek's Lightning Indexer (DeepSeek-AI et al., 2025). However, the differences in our latent structure allow each query to have its own subset of tokens rather than having to compete for a shared pool. For each expert selected head per token per layer, we store a 12-width int4 code which totals 24 bytes per layer. We then expand each query to size 96 to check each code with 8 different ReLU probes. As the scan is memory bound, this projection adds expressivity without a measurable increase in wall clock time. This operation was trained on 1 billion tokens to mimic the underlying model's attention distribution per head. Gradients are severed at the query expansion to prevent influence over the main model. Each query receives 2048 tokens including anchors, approximately 2% of the full 102400 trained context length. All resulting operations access only this subset of tokens.

Table 2: Glance statistics compared to oracle with 1536 tokens at 100k context, averaged over final 50 training steps. At inference the set is expanded to 2048 tokens

Layer	Mass	Max Mass	Recall	cos
L01	0.999	0.999	0.416	0.999
L05	1.000	1.000	0.701	1.000
L11	0.998	0.999	0.614	0.999
Top-1 capture (model-wide): 0.970				

With this method, the average token preserves over 99% of its attention mass. We also tested codes with a width of 8 (16 bytes per layer) and had very similar mean metrics, but chose a width of 12 to better accommodate tails. Table 2 shows various metrics at three layers for the trained glance. The concentrated mass makes recall generally lower (most tokens in the measured 1536 subset have almost no mass), showcased by the high preserved mass and output cosine alignment. The model wide Top-1 of 0.97 seems concerning on the surface, but it is important to remember that each head is selected separately for these metrics. So for a token to be truly missed, it would have to be missed for all four stored experts in that layer. Compared to other approaches like the Lightning Indexer which prunes by token, Quest that prunes chunks with pages (Tang et al., 2024), and NSA which pruned by token blocks (Yuan et al., 2025) our method's head level granularity allows greater sequence exposure at the same selection budget since heads can choose different subsets of tokens. DuoAttention (Xiao et al., 2024) operates on the head level but leaves some 'retrieval heads' with the full sequence unlike our method.

If head level detail is a must, the inclusion of private latents can substantially reduce memory accesses. With a single shared latent, if even one head is selected with a lightweight indexer, the entire latent has to be read. Since most tokens only have one selected head with these pruning settings, it becomes difficult to justify going beyond token level detail with pure MLA.

2.6 Architecture Comparison

With our custom architecture as a base, we trained several variants of the 16 NoPe heads to reveal underlying behavior. All models that incorporate latents use a fixed total width per layer.

1. **GQA** Used as control

2. **Clean Sparse** Custom architecture as described with no modifications
3. **Dense MLA** Size 112 latent. All 16 experts are active at every pass
4. **Routed MLA** Same size latent as dense, but with only 4 of 16 heads active instead of all 16. Compute for core operations is quartered, but equal memory accesses prevent any decode speedup.
5. **Hybrid Maps** size 48 shared latent, size 16 latent per kv pair that expands to foldable workspace
6. **Twin** size 48 shared latent, size 16 latent per kv pair with single expansion
7. **Expert only Maps** size 28 map for each expert (28 * 4 = 112) that expands to size 192 workspace for each head.

These arms were selected to maximize the ability to attribute results to underlying mechanisms. The hybrid maps, twin, expert only maps, and Routed MLA all received concentration aux. Any difference between the twin and hybrid would be due to the Foldable Maps system. Routed vs Dense MLA gives the cost of routing 4/16 compared to all active. Expert only Maps vs hybrid indicates how important the shared layer-wide latent is. Clean sparse vs GQA reveals the cost or benefit of KV MoE with a dense query side. And the comparison of each latent model to Clean Sparse gives the cost of the 4.57x KV compression ratio (or 18.28x in the case of Dense MLA) on the routed NoPe heads compared to each model's uncompressed parent. Expecting models to outperform uncompressed at that ratio would be a surprise, but proportional cost is key in assessing the viability of each setup.

3 Quality Comparison

3.1 RULER NIAH

Table 3: RULER S-NIAH, mean accuracy by context length. Data is sourced from the RULER Paul Graham essay haystack (Hsieh et al., 2024). RULER needle format and prompt template, partial credit.

	Context length						
Model	4K	8K	16K	32K	64K	100K	Avg.
GQA	**0.88**	0.62	0.29	0.18	0.12	0.17	0.38
sparse	0.74	0.79	0.60	0.28	0.22	0.17	0.47
hybrid_maps	0.85	0.82	0.64	0.43	**0.28**	**0.23**	0.54
glance	0.84	**0.83**	**0.68**	**0.46**	0.26	0.21	**0.55**
twin	0.36	0.39	0.16	0.11	0.08	0.08	0.20
dense_mla	0.22	0.21	0.16	0.18	0.15	0.20	0.19
routed_mla	0.25	0.24	0.14	0.12	0.05	0.07	0.14
expert_only_maps	0.24	0.18	0.10	0.09	0.09	0.09	0.13

Table 4: RULER MK-NIAH with 3 distractor keys, mean accuracy by context length. Each model was tested on an identical set of 20 documents per context length.

	Context length							
Model	4K	8K	16K	32K	64K	100K	Avg.	S + MK avg
GQA	**0.53**	**0.50**	0.32	0.21	0.14	0.15	**0.31**	0.343
sparse	0.31	0.27	0.29	0.19	0.17	0.17	0.23	0.350
hybrid_maps	0.41	0.34	**0.34**	**0.29**	**0.28**	**0.20**	**0.31**	**0.426**
glance	0.40	0.35	0.33	0.28	0.24	**0.20**	0.30	0.423
twin	0.22	0.21	0.17	0.17	0.11	0.10	0.16	0.180
dense_mla	0.19	0.20	0.18	0.17	0.17	0.17	0.18	0.183
routed_mla	0.12	0.10	0.09	0.09	0.07	0.09	0.09	0.119
expert_only_maps	0.21	0.18	0.15	0.16	0.12	0.05	0.14	0.138

To try to isolate long range retrieval in natural text, we ran RULER (Hsieh et al., 2024) single (S) and multi-key (MK) needle in a haystack (NIAH) tests seen in Tables 3 and 4 respectively. Queries are positioned only at the stated length to avoid double counting early contexts in the average column.

For these tests, the hybrid and glance models performed best across the breadth of measured contexts. GQA was the most effective at short contexts but tapered off late, and the pruned glance model outperformed its sparse counterpart at 6/6 lengths in both tests. When we compare the average column across all arms, there is a stark difference in performance between the experiment models, particularly with single key performance. Since the twin and the hybrid are identical besides the behavior of the private channel, aside from noise the difference between the 0.426 and 0.180 performance marks averaged across the twelve tests can be attributed entirely to the Foldable Maps.

The average score for the glance model over doubling dense MLA is a great result on its own, but what's more surprising is that it exceeded the baseline models as well. Compression over sparse (already 1/3 the kv of GQA) is 4.57x when considering active experts and 18.28x when considering all experts. An improvement with that level of restriction bodes well for quality oriented implementations with larger latents.

3.2 PERPLEXITY

Table 5: Perplexity by context length using the PG19 dataset, lower is better. CE results are from the final 50 training steps of the 8k pretrain and 100k finetune, glance is omitted from training columns since the indexer was trained separately.

	Context length							
Model	4K	8K	16K	32K	64K	100K	8k Training CE	100k Training CE
GQA	**29.29**	26.23	26.26	24.81	26.20	27.27	2.874	**2.488**
sparse	29.42	**26.12**	**26.04**	**24.65**	**25.52**	**26.47**	**2.836**	2.517
hybrid_maps	30.13	27.07	27.07	25.21	26.57	27.70	2.870	2.555
glance	30.13	27.07	27.10	25.29	26.79	27.98		
twin	31.03	28.12	28.11	26.10	27.70	28.83	2.880	2.577
dense_mla	31.77	28.35	28.83	27.01	28.49	29.53	2.888	2.548
routed_mla	30.72	27.54	27.49	25.54	26.95	28.13	2.878	2.583
expert_only_maps	31.60	28.52	28.67	26.69	28.41	29.43	2.894	2.649

The perplexity results in Table 5 are where all compression methods really lose ground. MLA actually scores the worst, but a large portion of that can be attributed to weaker domain transfer. Each model's final CE loss from the 100k Prolong dataset (Gao et al., 2025) finetune is listed in the final column. Although all models were trained on identical tokens in identical order in that dataset, the dropoff in performance when switching to the held out PG19 (Rae et al., 2019) test set varied wildly. MLA had a much larger falloff than the other compression methods which is important to note as there is likely more going on here than MLA simply being worse overall. The MLA and GQA models are the only ones where each query has access to the full sequence rather than just a quarter of it, and they also had the largest relative performance dropoff. This enables gist style usage that is not as viable with just a subset, so it stands to reason that those types of pathways had less carryover to the test set than retrieval heavy ones. With the current results that is mainly speculation but would be a prime candidate for a more thorough investigation. As for the 8k pretraining, relative differences are smaller and ordering varies due to different task dependencies.

Since the held out test set should usually be trusted over training performance, the 100k perplexity column is the most relevant when gauging each model's mean performance. The hybrid with Foldable Maps performed best of the compressed models, but of greater interest is the difference between the sparse, hybrid and twin perplexities. The hybrid's perplexity was 4.6% worse than sparse while the twin was 8.9% worse. If we treat this as the rent this level of compression demands, incorporating foldable maps decreased the disparity by 48% on otherwise identical models. The exact ordering of models and absolute magnitude of falloff would likely change with scaling and dataset modifications, but in this instance the maps supply an immense amount of benefit for their small byte cost.

3.3 PASSKEY

Table 6: Passkey retrieval, mean accuracy by context length. Identical prompts for each model.

	Context length						
Model	4K	8K	16K	32K	64K	100K	Avg.
GQA	**0.99**	**0.99**	**0.98**	0.70	0.31	0.22	0.70
sparse	0.98	**0.99**	**0.98**	**0.98**	0.97	0.88	0.96
hybrid_maps	0.52	0.63	0.61	0.52	0.78	0.88	0.66
glance	0.52	0.64	0.60	0.44	0.66	0.79	0.61
twin	0.98	0.97	0.97	0.94	**1.00**	**0.98**	**0.97**
dense_mla	0.35	0.35	0.27	0.50	0.32	0.23	0.34
routed_mla	0.23	0.17	0.17	0.11	0.31	0.34	0.22
expert_only_maps	0.18	0.17	0.16	0.18	0.17	0.16	0.17

For the passkey test in Table 6, a code is hidden in repeated filler text. Although less applicable to natural language than the NIAH tests, it is still useful to gauge a model's copy capabilities. The twin and sparse models performed best here, implying some significant benefit for this task with an unaltered private latent. The hybrid and GQA models had similar scores well below the best performing two, then all other models were far below. The high score of the twin could deserve further investigation as it is an interesting result in comparison with its NIAH results which were on par with MLA and well below the high performing models. Additionally, the hybrid performing better at long than short distances is unusual for long context testing. The passkey tests were run twice for each model to confirm the results. The two points together could hint at some underlying properties even if this task is less representative of retrieval in natural language where distractors are usually much less obvious. The honest takeaway is that Foldable Maps may trade away some verbatim-copy ability compared to the twin in exchange for natural language retrieval, even though it outperforms all other measured compression techniques on this task.

3.4 KERNEL PERFORMANCE

Table 7: Decode memory at $T = 102400$ on a single GH200 (94.5 GB total, 92.7 GB usable budget). Hybrid and Twin duplicate shared latent to ease kernel design which increases memory usage. Of the two, only the twin is fused.

Arm	KV/seq (MiB)	Weights (GiB)	Peak Memory (GiB)	Cache reads/seq (MiB)	Bandwidth (GB/s)
GQA	3751	1.30	31.0	3752	2219
sparse	1301	1.47	89.4	1222	**3101**
hybrid_maps	769	1.48	81.1	694	1767
glance	410	1.48	88.2	**101**	642
twin	739	1.42	78.0	697	2768
dense_mla	**317**	1.41	71.0	322	1114
routed_mla	**317**	1.41	69.9	322	1113

For each model, memory accesses can be found in Table 7 while Table 8 shows throughput at various batch sizes. The glance model performed the best at moderate and higher batch sizes, but due to inefficient kernels single sequence batches for both glance and hybrid are significantly worse than the other models. Although the kernels are immature, the supported batch sizes and non-exploding run times demonstrate that absorption holds for all models. This confirms the workspace was successfully folded as it would be immediately obvious otherwise (23,552 vs 144 Multiply-Accumulate Operations (MACs) per member). Without a fused kernel, the non duplicated hybrid was slightly slower per token due to gather overhead. As such it was only used for the glance pruned version where that is less relevant. Equal dense and routed MLA bandwidths show how reducing active heads does not improve decode speed with a single shared latent.

Table 8: Whole-model decode throughput (tokens/s) at $T = 102400$ on a single GH200, by batch size. Columns marked with – were skipped due to exceeding the 92.7 GB memory budget.

	Batch size						
Arm	1	8	32	64	96	128	192
GQA	165	540	–	–	–	–	–
sparse	247	1192	2073	2374	–	–	–
hybrid_maps	116	903	1914	2241	2375	–	–
twin	**255**	1346	2659	3200	3708	–	–
dense_mla	247	**1367**	2339	2801	2978	3360	3226
routed_mla	246	1365	2339	2803	2978	3357	3223
glance	168	1067	**2699**	**4009**	**4695**	**5086**	**5603**

4 Discussion

With the relatively small model sizes trained here (815M parameters), it is not yet possible to extrapolate if the retrieval differences would propagate to more complex language capabilities in larger models. For example, when MiniMax scaled a hybrid that interleaved linear and dense attention, they returned to pure dense as the blend caused multi-hop reasoning to suffer (Chen et al., 2026). Although not a direct analog to our model, it shows that some long context capabilities need a larger scale to properly demonstrate. That being said, there is reason to believe that the retrieval gains themselves would scale (Arora et al., 2023).

A key difference between tested models is the memory accesses in the quadratic, full sequence attention components. Going by pure byte accounting per token per layer in this configuration, our glance requires access to 24 bytes (plus 2% selected set) compared to 3072 for GQA, 1024 for sparse and 224 for MLA. When considering very long contexts such as 1 million, all costs in the model besides attention mixing are dwarfed (Yang et al., 2025). Those ranges are untested for our models and the kernels are still inefficient, but it is a promising number moving forward.

A successful recent method from DeepSeek uses Compressed Sparse Attention (DeepSeek-AI et al., 2026) to compress the KV cache across the sequence dimension then use their Lightning Indexer to select KV blocks. Even without the sequence level compression, queries still need to share a selected token subset. A huge advantage of our setup is that essential byte accesses are low while maintaining per-token per-head granularity. With pure MLA that head level detail is not helpful since all heads can be accessed with no additional memory traffic (DeepSeek-AI et al., 2024). The addition of Foldable Maps augmented private latents removes that boundary, making that higher level of selection detail and MoE expert selection more reasonable design decisions.

A core component of the Foldable Maps approach is the simplification of several linear projections into one. This property has been used for optimization purposes in compact networks via over-parameterization (Guo et al., 2021) which simplify for fast inference. Absorption makes nonlinearities particularly relevant for attention, as their specific placement can invalidate the property such as with RoPE (Xin et al., 2026). The process of careful combinations of linearities and nonlinearities seems like a prime target for training improvements that don't reduce decode throughput in the future.

5 Conclusion

If we could only rely on MLA, the usage of KV routing would be a moot point with our sparse model; it would be more reasonable to make all 16 experts active since memory accesses at decode would be the same either way. A combination of layer level and expert-level latents allows the sparsity of attention MoE to be utilized for decode and the KV cache while keeping the training benefits, but on its own lags behind MLA in several performance metrics. With the introduction of Foldable Maps, NIAH performance compared to the twin more than doubled on the average measured context, even exceeding the GQA and uncompressed sparse models. Since the mechanisms that facilitate this are absorbed at decode to avoid major expense, the path towards improved latent compression performance and common attention-side MoE in general could be much simpler.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for the following tasks:

1. **To aid or polish writing** We used LLMs to help with creation of figures, latex code, and math expressions.
2. **For retrieval and discovery** We used LLMs to find related works and other papers most associated with various concepts and mechanisms. Citations were all sourced from each paper's arXiv page not the LLM's output
3. **For research ideation and execution** We used Claude to help with the code files, brainstorming specific implementations, and for designing appropriate ablations and tests. The core ideas in the paper were not sourced in this manner, though they were communicated for purposes of code generation.
4. **For proving mathematical claims** The workspace folding proof and definition were created with the help of an LLM.
5. **help develop theoretical models or conceptual frameworks**
6. **formulate mathematical claims**
7. **provide critical ingredients for proving mathematical claims**
8. **assist in the writing of proofs**
9. **propose or refine hypotheses** Used to refine hypotheses based on experimental results
10. **design or provide feedback on research methodology or experiments**
11. **implement methods**
12. **interpret results** Some results were run by LLMs for an expanded interpretation.
13. **support qualitative and thematic data analysis** A few results were presented without context which may fall into this category
14. **suggest experimental parameters**
15. **paper review** No direct text was taken from review, but a check was run to validate result consistency and grammar

The first four points cover all our major usage. The others listed are mostly tangentially related and have descriptions if they are not. We included tasks that were loosely related to our usage unless we were absolutely certain they did not apply.

We did not use generative AI tools for the following tasks:

1. **To draft sections of the paper** Besides figure and math text, no words came from an LLM.
2. **For generating synthetic datasets** All datasets were referenced where applicable, details on exact sequences can be found in supplementary material.
3. **clean and reformat dataset**
4. **assist with translation**

We have reviewed all AI-assisted work. All LLM written code was inspected before runtime and based on outputs. Figures and math were also verified. The extent to which it was used for literature and research ideation makes plagiarism not applicable and any summaries were verified by reading the source papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Expanded ethics statement is not applicable.

Reproducibility statement

All code files essential to reproducing the results are contained in the supplementary materials zip file. Details on how to find training data, model files, and performance experiments are located in the readme file. To ensure reproducibility, all models were trained on an identical subset of tokens with identical effective batch sizes, total steps, and ordering. For kernel and long context tests, all models were run on a GH200. When applicable, all testing data (like the PG-19 and Paul Graham Essays for perplexity and NIAH respectively) was ensured to be identical for each model in each task. Although the exact results were reproduced on our end by keeping the seed constant, we did not ensure they are independent of file ordering and all other factors which remained constant.

References

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, 2023. URL https://arxiv.org/abs/2305.13245.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models, 2023. URL https://arxiv.org/abs/2312.04927.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2025. URL https://arxiv.org/abs/2402.18668.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. URL https://arxiv.org/abs/1409.0473.

Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changhao Zhang, Changqing Yu, Chao Wang, Cheng Ma, Cheng Zhong, Cheng Zhu, Chengjun Xiao, Chengyi Yang, Chengyu Du, Chenyang Zhang, Chi Zhang, Chuangyi Huang, Chunhao Zhang, Chunhui Du, Chunyu Zhao, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Dongyu Zhang, Enhui Yang, Fei Yu, Guang Zheng, Guodong Zheng, Guohong Li, Haichao Zhu, Haigang Zhou, Haimo Zhang, Han Ding, Hao Zhang, Haohai Sun, Haolin Lyu, Haonan Lu, Haoyu Wang, Huajie Shi, Huiyang Li, Jiacheng Chen, Jian Zhang, Jiaqi Zhuang, Jiaren Cai, Jiaxin Pan, Jiayao Li, Jiayuan Song, Jichuan Zhang, Jie Wang, Jihao Gu, Jin Zhu, Jingwei Dong, Jingyang Li, Jingyu Zhang, Jingze Zhuang, Jinhao Tian, Jinli Liu, Jinyi Hu, Jun Tao, Jun Zhang, Junbin Ruan, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kang Xu, Ke Ji, Ke Yang, Kecheng Xiao, Keyu Duan, Keyu Li, Le Han, Letian Ruan, Li Yuan, Lianfei Yu, Liheng Feng, Lijie Mo, Lin Li, Linge Du, Lingye Bao, Lingyu Yang, Lingyuan Zhou, Loki, Lu Chen, Lunbin Zeng, Ming Li, Ming Zhong, Mingliang Tao, Mingyuan Chi, Mujie Lin, Nan Hu, Ningxin Chen, Peiyin Zhu, Peng Gao, Pengcheng Gao, Pengfei Li, Penglin Li, Pengyu Zhao, Qibin Ren, Qibing Ren, Qidi Xu, Qihan Ren, Qile Li, Qin Wang, Quanliang Chen, Qunhong Zeng, Rong Tian, Rongxin Guo, Rui Dong, Ruitao Leng, Ruize Zhang, Shanqi Liu, Shaoxiang Chen, Shaoyu Chen, Sheng Jia, Shun Yao, Shuoran Zhao, Shuqi Yu, Sichen Li, Sicheng Pan, Songquan Zhu, Tengfei Li, Tian Xie, Tiancheng Qin, Tianle Li, Tianrun Liang, Wei Liu, Weiqi Xu, Weitao Li, Weixiang Chen, Weiyu Cheng, Weiyu Zhang, Wenhu Chen, Wenqian Zhao, Xiancai Chen, Xiangjun Song, Xiangyuan Wang, Xianzhen Luo, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xiaojie Wu, Xihao Song, Xingyi Han, Xinyu Guan, Xuan Lu, Xun Zou, Xunhao Lai, Xutong Li, Xuyang Shen, Yan Gong, Yan Ma, Yang Jiao, Yang Wang, Yang Xu, Yangsen Wang, Ye Tang, Yicheng Chen, Yihang Wang, Yinran Qiu, Yiqi Shi, Yiting Guo, Yiwen Huang, Yixuan Wang, Yongyi Hu, Yu Gao, Yu Zhang, Yuan Li, Yuanxiang Ying, Yuanzhen Zhang, Yubo Wang, Yuchen Song, Yufeng Yang, Yuhang Meng, Yuhang Miao, Yuhao Li, Yujie Liu, Yulin Hu, Yunan Huang, Yunji Li, Yunyi Huang, Yusen Zhang, Yusu Hong, Yutao Xie, Yutong Zhang, Yuwen Liao, Yuxuan Shi, Yuze Wenren, Zebin Li, Zehan Li, Zejian Luo, Zeyu Jin, Zeyuan Sun, Zhanpeng Zhou, Zhaochen Su, Zhendong Li, Zhengmao Zhu, Zhengyuan Peng, Zhenhua Fan, Zhi Zhang, Zhichao Xu, Zhiheng Lv, Zhikang Xu, Zhitao He, Zhiwei He, Zhongyuan Li, Zibo Gao, Zijia Wu, Zijian Song, Zijian Zhou, Zijun Sun, Zishan Huang, Ziying Chen, and Ziyue Ge. The minimax-m2 series: Mini activations unleashing max real-world intelligence, 2026. URL https://arxiv.org/abs/2605.26494.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: Lsh sampling for efficient llm generation, 2024. URL https://arxiv.org/abs/2410.16179.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu

Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference, 2025. URL https://arxiv.org/

abs/2412.14468.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively), 2025. URL https://arxiv.org/abs/2410.02660.

Shuxuan Guo, Jose M. Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks, 2021. URL https://arxiv.org/abs/1811.10495.

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?, 2024. URL https://arxiv.org/abs/2404.06654.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers, 2023. URL https://arxiv.org/abs/2305.19466.

Ken M. Nakanishi. Scalable-softmax is superior for attention, 2025. URL https://arxiv.org/abs/2501.19399.

Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. The sparse frontier: Sparse attention trade-offs in transformer llms, 2026. URL https://arxiv.org/abs/2504.17768.

Piotr Piękos, Róbert Csordás, and Jürgen Schmidhuber. Mixture of sparse attention: Content-based learnable sparse attention via expert-choice routing, 2025. URL https://arxiv.org/abs/2505.00315.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Jingze Shi, Yifan Wu, Yiran Peng, Bingheng Wu, Liangdong Wang, Guang Liu, and Yuyu Luo. Trainable dynamic mask sparse attention, 2025. URL https://arxiv.org/abs/2508.02124.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference, 2024. URL https://arxiv.org/abs/2406.10774.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui,

Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads, 2024. URL https://arxiv.org/abs/2410.10819.

Jihao Xin, Tian Lyu, David Keyes, Hatem Ltaief, and Marco Canini. Rap: Kv-cache compression via rope-aligned pruning, 2026. URL https://arxiv.org/abs/2602.02599.

Amy Yang, Jingyi Yang, Aya Ibrahim, Xinfeng Xie, Bangsheng Tang, Grigory Sizov, Jeremy Reizenstein, Jongsoo Park, and Jianyu Huang. Context parallelism for scalable million-token inference, 2025. URL https://arxiv.org/abs/2411.01783.

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, 2025. URL https://arxiv.org/abs/2502.11089.

A Expanded Proof

All quantities are for a single attention head and a single cached member (token–head pair).

Symbol	Shape	Definition
c_p	$\mathbb{R}^{16}$	Private latent: the per-member code stored in the cache for each active expert.
c_s	$\mathbb{R}^{48}$	Shared latent: the per-token code stored once and read by every head the token is routed to.
S	$\subset \{1, \dots, 24\}$	Active Foldable Map set for this member, $\lvert S\rvert = 6$; chosen per member and stored as a 24-bit mask.
A_i	$\mathbb{R}^{128\times 16}$	Map i: lifts the private latent into the 128-dimensional workspace. One of 24 learned maps per head.
z	$\mathbb{R}^{128}$	Workspace vector: the sum of the active maps applied to c_p.
h	$\mathbb{R}^{176}$	Decoder input: the workspace vector concatenated with the shared latent.
D	$\mathbb{R}^{64\times 176}$	Key decoder: maps h to a head-dimension key.
$\hat{k}$	$\mathbb{R}^{64}$	Reconstructed key for this member. Never formed at inference.
q	$\mathbb{R}^{64}$	Query for this head at the current decode step.
s	$\mathbb{R}$	Attention logit of the query against this member.
$D_w,\ D_s$	$\mathbb{R}^{64\times 128},\ \mathbb{R}^{64\times 48}$	Column blocks of D acting on the workspace and shared parts of h.
A_S	$\mathbb{R}^{128\times 16}$	Combined map of the active set, $\sum_{i\in S} A_i$.
$\widehat{W}_i$	$\mathbb{R}^{64\times 16}$	Folded map $D_w A_i$: map i composed with the workspace decoder. Fixed per head; precomputable offline.
$\tilde{q}_i$	$\mathbb{R}^{16}$	Absorbed private query for map i, $\widehat{W}_i^\top q$. Computed once per decode step for all 24 maps.
$\tilde{q}_s$	$\mathbb{R}^{48}$	Absorbed shared query, $D_s^\top q$. Computed once per decode step.

Full operation set

$$z = \sum_{i\in S} A_i\, c_p, \qquad A_i \in \mathbb{R}^{128\times 16},\ c_p \in \mathbb{R}^{16},\ z \in \mathbb{R}^{128} \tag{11}$$

$$h = \begin{bmatrix} z \\ c_s \end{bmatrix}, \qquad c_s \in \mathbb{R}^{48},\ h \in \mathbb{R}^{176} \tag{12}$$

$$\hat{k} = D\, h, \qquad D \in \mathbb{R}^{64\times 176},\ \hat{k} \in \mathbb{R}^{64} \tag{13}$$

$$s = q^\top \hat{k}, \qquad q \in \mathbb{R}^{64} \tag{14}$$

Collapse the set

$$D = \begin{bmatrix} D_w & D_s \end{bmatrix}, \qquad D_w \in \mathbb{R}^{64\times 128},\ D_s \in \mathbb{R}^{64\times 48} \tag{15}$$

$$A_S := \sum_{i\in S} A_i, \qquad A_S \in \mathbb{R}^{128\times 16} \tag{16}$$

$$s = q^\top D_w A_S\, c_p \ +\ q^\top D_s\, c_s \tag{17}$$

Reorder and absorb

$$s = \sum_{i\in S} \big(A_i^\top D_w^\top q\big)^\top c_p \ +\ \big(D_s^\top q\big)^\top c_s \tag{18}$$

$$\widehat{W}_i := D_w A_i \in \mathbb{R}^{64\times 16}, \qquad \tilde{q}_i := \widehat{W}_i^\top q \in \mathbb{R}^{16}, \qquad \tilde{q}_s := D_s^\top q \in \mathbb{R}^{48} \tag{19}$$

$$s = \Big(\sum_{i\in S} \tilde{q}_i\Big)^\top c_p \ +\ \tilde{q}_s^\top c_s \tag{20}$$

The value side differs and requires its own proof for completeness. Quantities are now summed over the T cached members visible to each query at each decode step. Since the value decoder mixes across members and those members' values are conditional on their active maps, MLA's standard absorption does not apply. However, the property still holds by exchanging summation order:

Symbol	Shape	Definition
S_t^v	$\subset \{1,\dots,24\}$	Active value-map set for member t, $\lvert S_t^v\rvert = 6$; stored as the 24-bit v mask.
B_i	$\mathbb{R}^{128\times 16}$	Value map i: lifts the private latent into the workspace. One of 24 learned maps per head.
z_t^v	$\mathbb{R}^{128}$	Value workspace vector of member t. Never formed at inference.
E	$\mathbb{R}^{64\times 176}$	Value decoder; $E = \begin{bmatrix} E_w & E_s \end{bmatrix}$, $E_w \in \mathbb{R}^{64\times 128}$, $E_s \in \mathbb{R}^{64\times 48}$.
$\hat{v}_t$	$\mathbb{R}^{64}$	Reconstructed value of member t. Never formed at inference.
α_t	$\mathbb{R}$	Attention weight of member t, from the folded score path.
a_i	$\mathbb{R}^{16}$	Accumulator of map i, over members whose mask contains i.
a_s	$\mathbb{R}^{48}$	Shared accumulator, over all members.
$\widehat{V}_i$	$\mathbb{R}^{64\times 16}$	Folded value map $E_w B_i$. Fixed per head; precomputable offline.
u	$\mathbb{R}^{64}$	Head output at this decode step.

Full operation set

$$z_t^v = \sum_{i\in S_t^v} B_i\, c_{p,t}, \qquad B_i \in \mathbb{R}^{128\times 16},\ z_t^v \in \mathbb{R}^{128} \tag{21}$$

$$\hat{v}_t = E \begin{bmatrix} z_t^v \\ c_{s,t} \end{bmatrix} = E_w z_t^v + E_s c_{s,t}, \qquad \hat{v}_t \in \mathbb{R}^{64} \tag{22}$$

$$u = \sum_{t=1}^{T} \alpha_t\, \hat{v}_t, \qquad u \in \mathbb{R}^{64} \tag{23}$$

Regroup by map

$$u = E_w \sum_{t=1}^{T} \alpha_t \sum_{i\in S_t^v} B_i\, c_{p,t} \ +\ E_s \sum_{t=1}^{T} \alpha_t\, c_{s,t} \tag{24}$$

$$= E_w \sum_{i=1}^{24} B_i \underbrace{\sum_{t\,:\, i\in S_t^v} \alpha_t\, c_{p,t}}_{=:\, a_i} + E_s \underbrace{\sum_{t=1}^{T} \alpha_t\, c_{s,t}}_{=:\, a_s} \tag{25}$$

Absorb

$$\widehat{V}_i := E_w B_i \in \mathbb{R}^{64\times 16} \tag{26}$$

$$u = \sum_{i=1}^{24} \widehat{V}_i\, a_i \ +\ E_s\, a_s \tag{27}$$

For a single member at decode these operations cost 6*16 + 48 = 144 MACs per side if folded and 6(128*16) + 64*176 = 23,552 per side if unfolded. Each side incurs an additional 27,648 MAC cost per-query per-head, independent of sequence length.