Retrieve useful depth history inside attention

Mixture-of-Depths Attention (MoDA)

Sequence KV and same-position historical KV compete in one softmax.

Lianghui Zhu, Yuxin Fang, Bencheng Liao et al.
Huazhong University of Science and Technology / ByteDance Seed

- Paper: arXiv:2603.15619v1, 16 March 2026.
- Independent paper-reading presentation; 17 September 2026.
- 20 main slides + 4 backup slides; approximately 25 minutes.

Source: author TeX and version-matched PDF, pp. 1–18. No independent model reproduction.

The question: can deep layers recover shallow features?

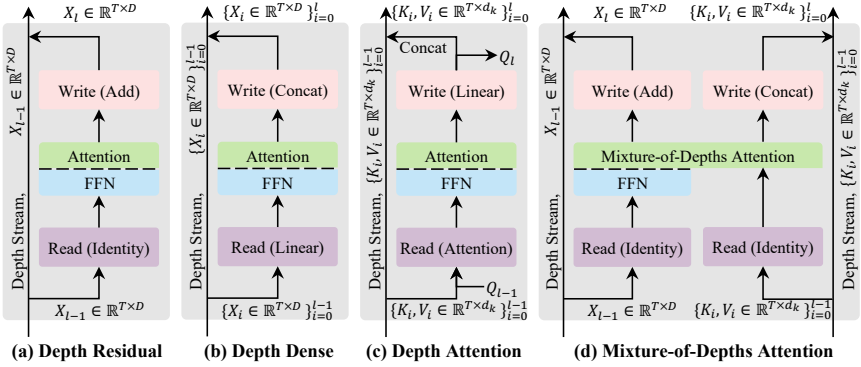
- Residual addition makes deep optimization easier, but repeatedly mixes history into a fixed-width state.
- Dense history preserves intermediate states, but can require expensive cross-layer projections.
- MoDA retains the residual stream and adds content-dependent retrieval of historical KV.

Evidence boundary

"Information dilution" motivates the architecture. The paper does not isolate it with a causal information-loss measurement.

Paper, Sec. 1, PDF pp. 2–3; Sec. 2.2, pp. 3–5.

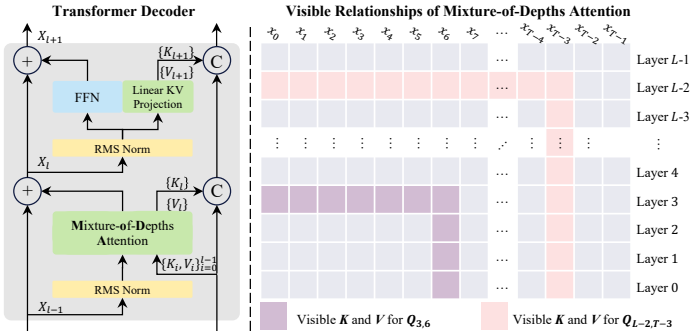
MoDA keeps a residual stream and a depth-KV stream



Attention supplies a dynamic read operation; MoDA fuses that read with sequence attention.

Paper, Fig. 3, PDF p. 4. Original figure.

Depth retrieval is restricted to the same token position



- Horizontal: causal sequence KV.
- Vertical: same-position depth KV.
- All Transformer layers still execute.

Paper, Fig. 1, PDF p. 1. Original figure; colored regions show two example queries.

One denominator allocates probability across both memories

For one query head at layer l, position t:

$$a_j = q_{l,t}^\top k_{l,j}/\sqrt{d}, \qquad b_i = q_{l,t}^\top k_{i,t}^{\mathrm{depth}}/\sqrt{d}$$

$$o_{l,t} = \frac{\sum_{j\le t} e^{a_j} v_{l,j} + \sum_{i\in\mathcal{H}_l} e^{b_i} v_{i,t}^{\mathrm{depth}}}{\sum_{j\le t} e^{a_j} + \sum_{i\in\mathcal{H}_l} e^{b_i}}$$

- d: head dimension; $\mathcal{H}_l$: previously written depth slots.
- Values and per-head output have dimension d.
- Separate softmax operations followed by addition give a different operator.

Equivalent teaching expansion of Sec. 2.2, PDF p. 5 and Algorithm 1, p. 6; not a numbered paper equation.

Reuse attention KV; add a projection for FFN inputs

1. Read the current state and previously stored depth KV.
2. Compute joint sequence/depth attention using the existing query projection.
3. Preserve attention KV for later layers.
4. Project the **FFN input** into additional depth KV slots.

Default variant

Reuse attention-side KV + extra FFN KV projection.
An independent attention-side KV projection gives little additional gain.

Paper, Sec. 2.2, p. 5; Sec. 4.2.1 and Table 3, PDF pp. 10–11.

Depth compute still accumulates quadratically in depth

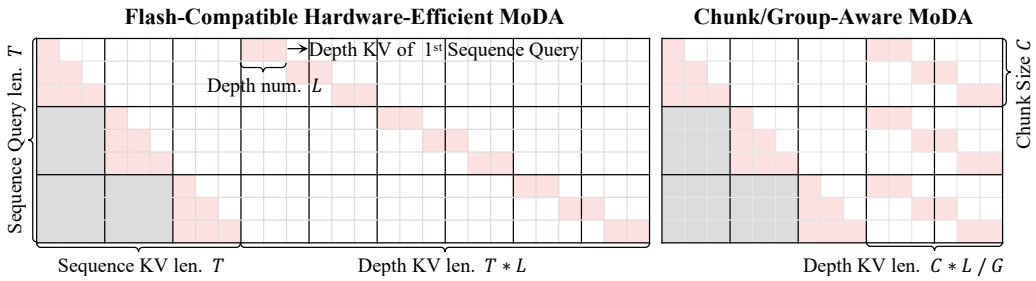
Depth-stream cost	Depth Dense	MoDA
Parameters	$O(L^2D^2)$	$O(LD^2/G)$
Decode cache	$O(LD)$	$O(LD/G)$
Prefill cache	$O(TLD)$	$O(TLD/G)$
Decode FLOPs	$O(L^2D^2)$	$O(L^2D)$
Prefill FLOPs	$O(TL^2D^2)$	$O(TL^2D)$

T: tokens; L: depth; D: width; $G = H_q/H_k$: GQA group size.

These are depth-stream terms, excluding ordinary sequence KV cache and the full attention/FFN cost.

Paper, Table 1, PDF p. 5. Attention-only KV reuse adds no projection parameters (Table 3).

Local packing avoids scanning irrelevant depth entries



- Contiguous layout: depth axis of length TL; dense utilization $1/T$.
- Chunk-aware access: local span CL; utilization $1/C$.
- GQA reuse: span $(C/G)L$; utilization G/C.

Paper, Fig. 4 and Sec. 3.2, PDF pp. 7–8. Original figure. C: expanded-query block size.

Shared online-softmax states preserve the joint operator

Maintain a running maximum m, denominator acc, and output numerator o:

$$
\begin{aligned}
m' &= \max(m, \max S), \\
acc' &= e^{m-m'} acc + \sum e^{S-m'}, \\
o' &= e^{m-m'} o + \sum e^{S-m'} V.
\end{aligned}
$$

- Process sequence blocks, then depth blocks, with the same states.
- Normalize once: o/acc. Do not materialize the full score matrix.
- Grouped masks preserve causality and same-position retrieval.

Natural-exponential teaching form of Algorithm 1, PDF p. 6; implementation details, pp. 8–9. See backup caveat.

Three evaluation settings answer different questions

Language-model quality

700M / 1.5B; 400B OLMo2 tokens; bf16; context 4096; global batch 1024. Ten downstream tasks and ten validation domains.

Depth and normalization

24 / 48 layers on FineWeb-Edu; width 384; 6 query heads and 2 KV heads. Held-out validation loss.

Kernel efficiency

A100, bf16, forward + backward. Shape sweeps and incremental kernel ablations; no full-model serving benchmark.

Paper, Sec. 4.1, p. 10; Sec. 4.3.1, p. 12; Tables 2/7, pp. 9/13.

At fixed parameter count, depth KV improves quality

700M-scale variant	Params M	FLOPs T	C4 PPL	Avg. score
OLMo2, 36 layers	669.0	8.01	18.59	56.93
+ reused depth KV	669.0	8.02	18.48	58.10
+ FFN KV (default)	705.7	8.33	18.21	58.87
+ separate attn KV	742.4	8.63	18.17	58.97

- Reused KV: +1.17 percentage points at unchanged parameters.
- FFN KV: another +0.77 points.
- Separate attention KV: only +0.10 points for +36.7M parameters.

Paper, Table 3, PDF p. 10. PPL: lower is better; score: higher is better. FLOPs units as reported.

Adding depth memory beats adding two baseline layers

Model	Layers	Params M	FLOPs T	Avg. score
OLMo2	38	700.5	8.41	57.11
MoDA default	36	705.7	8.33	58.87

- +1.76 percentage points under approximately matched resources.
- C4 PPL: 18.31 $\rightarrow$ 18.21 in this ablation table.
- Resources are close, not exactly equal; no multi-seed uncertainty is reported.

Paper, Table 3, PDF p. 10. Table 5 reports different 700M C4 values; see backup.

Average downstream gains persist at both model scales

Scale	OLMo2	MoDA	Difference
700M	57.11	58.87	+1.76 pp
1.5B	62.28	64.39	+2.11 pp

- 1.5B ARC-C: 42.47 $\rightarrow$ 46.82 (+4.35 pp).
- 1.5B COPA: 81.00 $\rightarrow$ 85.00 (+4.00 pp).
- Exceptions: 700M PIQA falls 0.33 pp; 1.5B ARC-E falls 0.17 pp.
- +2.11 pp is about +3.39% relative to 62.28.

Paper, Table 4, PDF p. 11. Averages over ten tasks; pp = percentage points.

Validation perplexity improves in all ten reported domains

Scale	OLMo2 mean PPL	MoDA mean PPL	Change
700M	15.61	15.46	−0.15
1.5B	13.67	13.47	−0.20

- 700M m2d2-s2orc: 24.37 → 23.64.
- 1.5B Reddit: 21.21 → 20.85.
- Mean of domain PPLs is not pooled-corpus PPL.
- These evaluations do not establish long-context retrieval quality.

Paper, Table 5, PDF p. 11. Domains: C4, ICE, m2d2-s2orc, Pile, Wiki-text, and five Dolma domains.

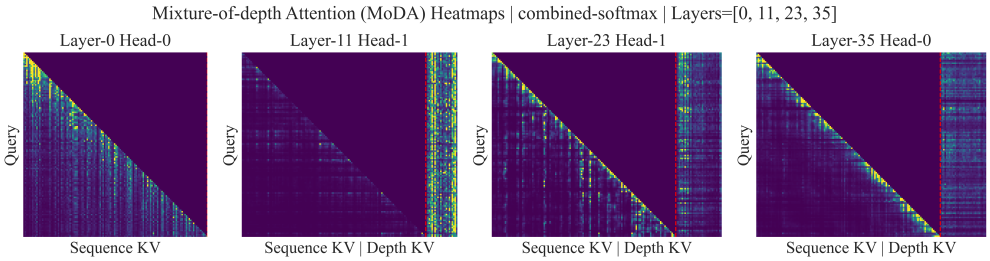
Post-norm benefits more in the 48-layer small-model study

Depth / norm	Baseline	+ Depth KV	+ FFN KV
48 / pre-norm	3.3800	3.3759	3.3656
48 / post-norm	3.4062	3.3653	3.3484
24 / post-norm	3.4740	3.4537	3.4338

- Depth KV gain at 48 layers: 0.0041 vs. 0.0409 loss reduction.
- FFN KV improves every reported setting further.
- This is FineWeb-Edu validation loss, not PPL; widths and training regime differ from the main study.

Paper, Table 6 and Sec. 4.3.1, PDF pp. 12–13. Parameters/FLOPs are not held fixed across all rows.

The visualized heads use depth memory



- Query rows attend to sequence KV, then depth KV after the red boundary.
- Columns: layers 0, 11, 23, 35; one selected head per column shown.
- Depth attention is visible, but heatmaps alone do not establish that reducing attention sinks causes the quality gains.

Paper, Fig. 5, PDF p. 14; first-row crop of the original figure. 700M model, 400B training tokens.

The 97.3% headline is a 64K kernel runtime ratio

Tokens	FA2 ms	MoDA ms	Extra time vs. FA2
4,096	7.970	10.750	34.88%
16,384	116.700	127.661	9.39%
65,536	1831.668	1883.026	2.80%

$1831.668/1883.026 = 97.27\%$: inverse runtime ratio.

The paper's "extra time" uses MoDA time as denominator (2.73% at 64K); the table above uses FA2 time.

This is an attention forward/backward benchmark, not full-model throughput.

Recomputed from Table 2, PDF p. 9. A100/bf16; B=1, d=C=L=64, G=8, Hq=64, Hk=8.

All three layout optimizations contribute to kernel speed

Implementation	Forward + backward ms
Naive PyTorch	2128.900
Flash-compatible	13.102
+ Chunk-aware	6.286
+ Group-aware	1.460

- Group-aware indexing adds a 4.31× speedup over the prior row.
- The 1458× total gain is against naive PyTorch, not FA2.
- Fixed short sequence: $T = 1024$; no model-quality comparison here.

Paper, Table 7, PDF p. 13. A100/bf16; B=1, G=8, Hq=64, Hk=8, d=L=C=64.

The main open cost is retaining and reading depth history

- More depth slots increase cache capacity, bandwidth, and attention work.
- Large-model evidence stops at the 1.5B class; no serving latency or peak-memory benchmark is reported.
- No multi-seed confidence intervals or causal test of the proposed information-dilution mechanism.

Future work in the paper

A bounded $S \ll L$ slot buffer (top-S, sliding window, or hybrid) and further CUDA/distributed optimizations are proposals, not evaluated results.

Paper, Sec. 6.1–6.2, PDF p. 15; evidence boundaries from reported experiments.

Take away the retrieval idea and keep the conditions

1. **Architecture:** retrieve same-position depth KV jointly with causal sequence KV.
2. **Design choice:** attention KV reuse is cheap; FFN KV supplies additional value.
3. **Evidence:** +2.11 pp average downstream score at the 1.5B scale; 0.20 lower mean domain PPL.
4. **Systems:** long-context kernel overhead is small under the measured configuration; real training and serving need separate validation.

Zhu et al., Mixture-of-Depths Attention, arXiv:2603.15619v1.

Primary source: version-matched author TeX and PDF. All results are paper-reported unless marked recomputed.

Backup: grouped indexing preserves the correct token map

$$T_q = GT_{kv}, \qquad t_{base}(i_q) = \lfloor i_q / G \rfloor$$

$$\text{Sequence visible: } \lfloor i_q / G \rfloor \geq i_k$$

$$\text{Depth visible: } \lfloor i_q / G \rfloor = \lfloor j_d / L \rfloor$$

- i_q: expanded query row; i_k: sequence-key position.
- j_d: flattened depth-key index; L: number of depth slots.
- For $G = 4$, rows 8–15 map to positions 2 and 3 only.
- Query block size must align with G; text length is unchanged.

Paper, Algorithm 1, p. 6 and Sec. 3.3, PDF p. 8.

Backup: depth and GQA shape sweeps have qualifications

Depth sweep: T=16,384; G=8

$L = 64, 128, 256$: MoDA times 127.661, 138.224, 167.958 ms.
FA2 remains 116.700 ms. More depth is not free.

Group sweep: T=16,384; L=64; Hk=8

$G = 2 \rightarrow 32$: utilization $3.12\% \rightarrow 50\%$.
But query heads also increase $16 \rightarrow 256$, changing workload size.

All sweeps: A100, bf16, forward + backward; $B = 1$, $d = C = 64$.

Paper, Table 2, PDF p. 9. Utilization here concerns valid depth entries within block computations.

Backup: retain the reporting discrepancies

- **700M C4 PPL:** Table 3 gives 18.31 / 18.21 for the matched baseline/default; Table 5 gives 18.32 / 18.29. The paper does not explain the difference.
- **3.7% FLOPs:** reported in the abstract alongside 1.5B gains; no corresponding detailed 1.5B cost table. Table 3's 8.01 $\rightarrow$ 8.33 is about +4.00%.
- **Evaluation details:** per-task shots and metric variants, complete 1.5B architecture/cost details, and multi-seed variability are not fully specified.

Paper, abstract p. 1; Tables 3–5 and Sec. 4.1, PDF pp. 10–11. No silent reconciliation in this reading.

Backup: pseudocode needs implementation-level checks

- Algorithm 1 writes 2^{S-m} with $S = QK^\top/\sqrt{d}$.
- To reproduce the natural-exponential softmax of Eq. (2), base-2 exponentiation requires logits scaled by $\log_2 e$.
- Block indices are compared with token ranges in the pseudocode; a kernel needs explicit block offsets and boundary handling.
- These are notation/implementation checks, not evidence that the released code is incorrect. The code was not executed in this reading.

Paper, Eq. (2), PDF p. 3; Algorithm 1, p. 6; Sec. 3.3, pp. 8–9. Reader analysis.