MoNe: Modular Neural Memory for Efficient Long Context Inference

Paper authors: Wonguk Cho, Kyubyung Chae, Tribhuvanesh Orekondy, Sunghyun Park, Hyoungwoo Park, Jeongho Kim, Arash Behboodi, Kyuwoong Hwang, and Sungrack Yun

Qualcomm AI Research

Core idea

Write context into small neural memories; let a frozen Transformer read them through its existing attention interface.

Research discussion, 9 October 2026
arXiv:2608.17616v1 AdaptFM Workshop at ICML 2026

Can a frozen model use a fixed-size context state?

Full-context attention

Store token-level keys and values.

- Attention prefill cost: $O(N^2)$.
- Context KV storage: $O(N)$.
- Small models may struggle even within their native window.

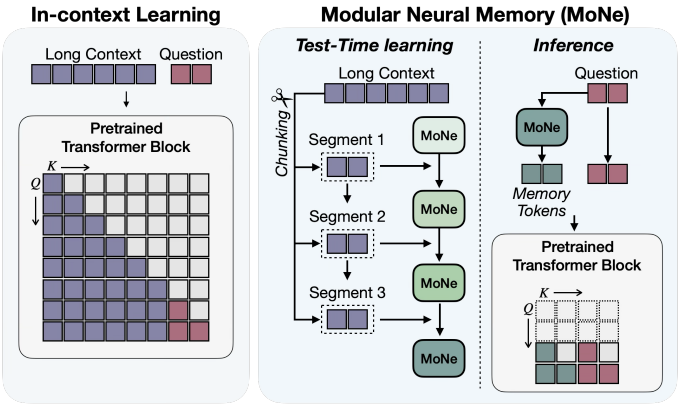
Paper, Secs. 1–2, PDF pp. 1–3. N: context tokens.

MoNe's proposal

Store context in adaptable weights.

- Keep pretrained backbone weights frozen.
- Learn a task-relevant write rule offline.
- Pay for context ingestion once, then reuse the state.

Separate context writing from question answering



Paper, Fig. 1, PDF p. 1; Sec. 3, PDF pp. 3–4.

Write

Consume 512-token segments and update each layer's memory.

Read

Generate memory KV from the question and final state.

Original context tokens are absent at answering time.

The backbone is frozen; the memory system is trained

Stage	Objective / input	Parameters that change
Offline learning	Answer generation after context writing	LoRA and update/read meta-parameters
Context writing	Local key–value association	Per-layer fast weights and update state
Answering	Question and generated tokens	Memory weights remain fixed

No backbone retraining still involves offline adaptation and gradient-based context ingestion at deployment.

Paper, Secs. 3.3–3.4, PDF p. 4; Appendix A, PDF p. 10.

A nonlinear fast-weight network carries the history

$$\mathcal{M}(\mathbf{x}; \mathbf{W}^{(l)}) = \mathbf{W}_{\text{out}}^{\top}\left[\text{SiLU}(\mathbf{W}_{\text{in}}\mathbf{x}) \odot (\mathbf{W}_{\text{gate}}\mathbf{x})\right]$$

- Three $d_h \times d_h$ matrices per memory head.
- Experimental configuration: $H = 4$, $d_h = 224$, all 24 layers.
- Fixed-size state; authors report 6.4% total parameter overhead.
- The memory output becomes an additional attention input.

Capacity implication

A fixed state can learn useful associations, but information from different context segments must share that state.

Paper, Eq. (3), PDF p. 3; Appendix A.2, PDF p. 10. Head dimensions refer to memory.

Writing maximizes agreement with layer-local values

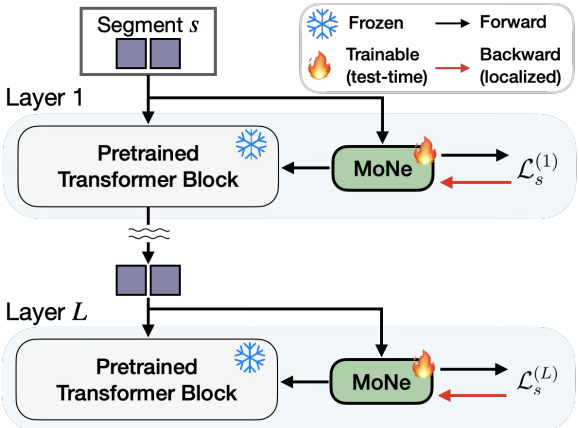
$$\ell_{s,j}^{(l)}(\mathbf{W}) = -(\mathbf{v}_{s,j}^{(l)})^{\top}\mathcal{M}(\mathbf{k}_{s,j}^{(l)};\mathbf{W}), \qquad \mathcal{L}_s^{(l)} = \frac{1}{T}\sum_{j=1}^{T}\ell_{s,j}^{(l)}$$

$$\mathbf{W}_s^{(l)} = \mathbf{W}_{s-1}^{(l)} - \boldsymbol{\mu}_s^{(l)}$$

- $\mathbf{k}, \mathbf{v}$ come from the current layer's segment activations.
- $\boldsymbol{\mu}$: gradients with learned per-token rates and momentum.
- Key/query: affine rescaling, SiLU, token-wise L2 normalization.
- Value: SiLU; weights: channel-wise L2 renormalization after updates.

Paper, Eqs. (6)–(8), PDF p. 4; Appendix A.2, PDF p. 10. The loss uses a negative inner product.

Local write losses avoid a full backward pass



Paper, Fig. 3, PDF p. 3; Eqs. (10)–(11), PDF p. 4.

- Each write loss updates only its layer's memory.
- No answer-logit gradient must traverse subsequent layers.
- Forward activations still flow through the backbone.

This boundary describes test-time writing; offline training uses answer supervision.

The question reads memory through extra attention KV

$$\mathbf{h}_j = \mathcal{M}(\mathbf{W}_q \mathbf{x}_j; \mathbf{W}_S^{(l)})$$

$$\text{out} = \text{Attention}(\mathbf{Q}, [\mathbf{K} \,\|\, \mathbf{W}_k \mathbf{h}], [\mathbf{V} \,\|\, \mathbf{W}_v \mathbf{h}])$$

- Read the state after all S context segments.
- Apply RMSNorm and a learned gate to the raw memory output.
- Reuse backbone projections, with offline-trained LoRA adapters.
- Attention combines ordinary query entries with memory entries.

Paper, Eqs. (4)–(5), PDF p. 3; Sec. 3.2, PDF pp. 3–4. Equations omit normalization details.

Local positions stay in range as context grows

$$\underbrace{\mathbf{s}_1}_{[0,T)} \longrightarrow \underbrace{\mathbf{s}_2}_{[0,T)} \longrightarrow \cdots \longrightarrow \underbrace{\mathbf{s}_S}_{[0,T)}$$

- Segment-local RoPE reuses positions in $[0, T)$; $T = 512$.
- Query positions lie in $[0, Q)$, assuming $Q \ll T$.
- Segment order enters through the sequence of memory updates.

Two different extrapolation factors

Offline training: up to 4K $\rightarrow$ evaluation: 128K $\quad(32\times)$.
Backbone native window: 32K $\rightarrow$ evaluation: 128K $\quad(4\times)$.

Bounded positions remove a positional obstacle; memory capacity remains finite.

Paper, Sec. 3.3, PDF p. 4; Table 1, PDF p. 5; Appendix A.1, PDF p. 10.

The constant query cost is relative to context length

$$F_{\text{write}}(N) = \left\lceil \frac{N}{T} \right\rceil c(T) = O(N)$$

$$F_{\text{total}} = F_{\text{write}}(N) + \sum_{i=1}^{m} F_{\text{query}}(Q_i, A_i)$$

- Fix segment size, backbone, memory size, and answer length.
- Query cost does not grow with the original context length N.
- Q_i and A_i still affect compute and generation storage.
- Multi-query reuse amortizes writing; ICL can also reuse prefix KV.

Paper, Sec. 1, PDF p. 2; Secs. 3–4.3, PDF pp. 3–6. Cost decomposition is an explanatory derivation.

The tasks test retrieval and frequency aggregation

Task	What the model must recover	Metric ↑
S-NIAH	One planted key–value pair	Substring EM
MK-NIAH	One queried value among four planted pairs; others are distractors	Substring EM
FWE	Top three non-noise words from a synthetic Zipfian stream	Target-word recall

MK-NIAH does not require chaining the four facts.
FWE measures recall, despite the Sub-EM wording in Sec. 4.2.

Paper, Sec. 4.1, PDF p. 4; task definitions in Appendix A.1, PDF p. 10.

Shared backbone, different task-adaptation budgets

- **Backbone:** Qwen2.5-0.5B-Instruct; native window 32K.
- **MoNe training:** 1K–4K contexts; 30K samples per segment count $S \in \{2, \ldots, 8\}$, with 512-token segments.
- **Evaluation:** 100 samples per length setting; 4K–128K.
- **ICL:** use the full context as a prompt.
- **RAG:** 128-token chunks, BGE-Large, cosine retrieval; report the best $K \in \{1, 4, 8\}$ at each length.

Comparison to keep in mind

MoNe receives offline task supervision for its adapters and update rule.
The baselines are described as direct ICL and retrieval prompting.

Paper, Sec. 4.1, PDF pp. 4–5; Sec. 3.3, PDF p. 4; Appendix A.1, PDF p. 10.

Gains already appear within the native window

Task	Length	ICL	RAG	MoNe
S-NIAH	4K	0.95	0.94	**1.00**
	32K	0.94	0.93	**1.00**
MK-NIAH	4K	0.89	0.92	**1.00**
	32K	0.93	0.79	**0.99**
FWE	4K	0.59	0.58	**1.00**
	32K	0.41	0.61	**1.00**

Largest gap: global frequency information

At 32K, FWE recall improves by 59 percentage points over ICL and 39 points over RAG.

Paper, Table 1, PDF p. 5. NIAH: Sub-EM; FWE: variable recall. Differences calculated from table.

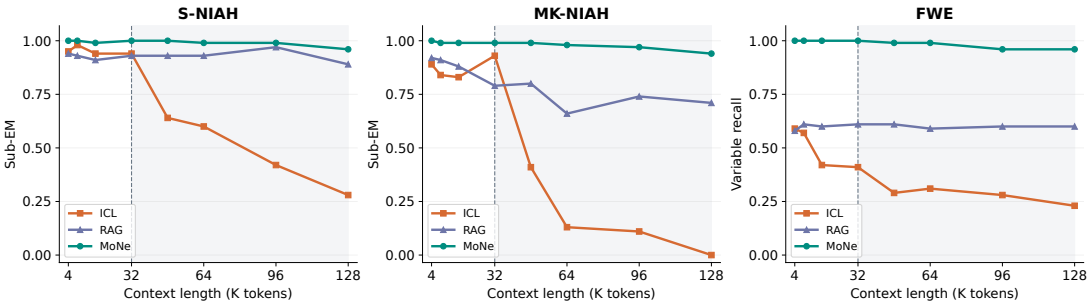
At 128K, MoNe retains high scores on all three tasks

Task at 128K	ICL	RAG	MoNe	Gain vs. RAG
S-NIAH	0.28	0.89	**0.96**	+7 pp
MK-NIAH	0.00	0.71	**0.94**	+23 pp
FWE	0.23	0.60	**0.96**	+36 pp

- Training contexts stop at 4K.
- ICL is evaluated beyond the backbone's native 32K window.
- Evidence covers controlled retrieval and aggregation workloads.

Paper, Table 1, PDF p. 5. NIAH: Sub-EM; FWE: variable recall. pp = percentage points.

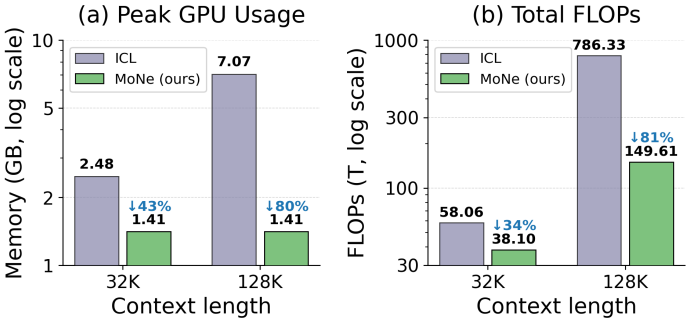
Length scaling separates the methods clearly



The dashed boundary is 32K; shading marks the region beyond it.
MoNe degrades gradually, while ICL retrieval falls sharply.

Redrawn from the exact values in Table 1, PDF p. 5. The right panel uses variable recall.

At 128K, total FLOPs and peak memory fall by about 80%



Peak GPU memory
7.07 → 1.41 GB
80.06% lower

Total compute
786.33 → 149.61 T
80.97% lower

Includes context writing and answering.

Logarithmic axes. FLOPs are operation counts, not measured latency.

Paper, Fig. 2, PDF p. 1; Table 2, PDF p. 5. T FLOPs = 10^{12} operations. Offline training excluded.

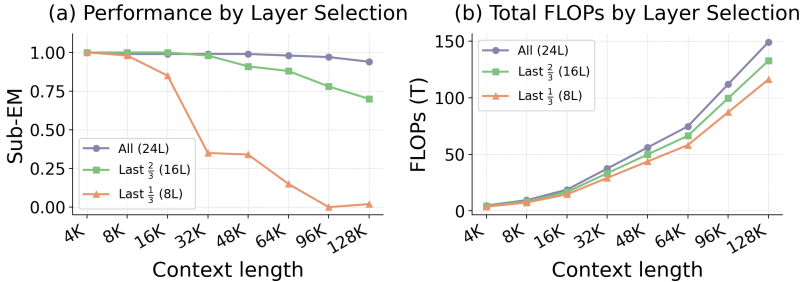
Writing accounts for almost all single-query compute

FLOPs (T)	32K	128K
ICL	58.06	786.33
MoNe: context writing	37.24	148.97
MoNe: answering	0.86	0.64
MoNe total	**38.10**	**149.61**

- Writing grows about 4× as context grows from 32K to 128K.
- Peak memory stays at 1.41 GB for writing and 1.29 GB for reading.
- At 128K, writing is 99.6% of the reported total compute.
- Multi-query amortization is proposed, without a dedicated timing study.

Paper, Table 2, PDF p. 5; multi-query reuse in Sec. 1, PDF p. 2. Peak memory uses max, not sum.

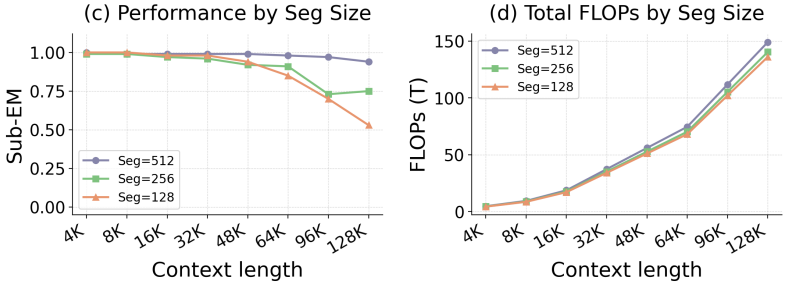
Removing memory from lower layers is costly in accuracy



At 128K	All 24 layers	Last 16 layers	Last 8 layers
MK-NIAH Sub-EM	0.94	0.70	0.02
FLOPs (T)	149.1	132.6	116.1

Paper, Fig. 4(a–b) and Sec. 4.4, PDF p. 6. The ablation reports 149.1 T; Table 2 reports 149.61 T.

Larger segments give a better tested trade-off



Segment size	512	256	128
64K MK-NIAH	0.98	0.91	0.85
128K MK-NIAH	0.94	0.75	0.53

Paper, Fig. 4(c–d) and Sec. 4.4, PDF p. 6. Smaller segments save little compute in these experiments.

The evidence is strongest for trained retrieval memory

- **Demonstrated:** one small backbone, three controlled tasks, length extrapolation, and lower deployment FLOPs / peak memory.
- **Comparison scope:** MoNe has offline task supervision; ICL beyond 32K also faces native-window extrapolation.
- **Task scope:** MK-NIAH queries one of four keys; natural document reasoning remains future work.
- **System scope:** no measured latency or energy result; RAG is absent from the computational cost table.

Paper, Secs. 4–5, PDF pp. 4–6; Appendix A.1, PDF p. 10. Interpretation based on the reported protocol.

Three experiments would clarify practical usefulness

1. **Measure actual memory capacity.**
   Increase independent facts, temporal dependencies, and continued writes; track old-fact retention as well as new-fact accuracy.

2. **Match task-adaptation budgets.**
   Compare trained baselines and isolate memory, position, and update choices.

3. **Measure a complete serving workload.**
   Fix query/output lengths; include ingestion, multiple queries, and RAG costs; allow ICL prefix-cache reuse.

Discussion proposals. Paper's stated future directions: larger backbones and natural workloads, Sec. 5, PDF p. 6.

A useful interface between learned memory and attention

Method

Local associative updates compress context into nonlinear fast weights.
Query-conditioned reads inject that information as extra attention KV.

Result

4K-trained memory reaches 128K on controlled tasks, with
0.96 / 0.94 / 0.96 scores and about 80% lower FLOPs / peak memory.

Practical interpretation

Offline-trained memory can reduce dependence on retained context tokens;
natural workloads and measured serving speed are the next tests.

Cho et al., MoNe: Modular Neural Memory for Efficient Long Context Inference.
arXiv:2608.17616v1; Tables 1–2, PDF p. 5; Sec. 5, PDF p. 6. FWE score is recall.

Backup: a linear example explains associative writing

$$\mathcal{M}(\mathbf{k}; A) = A\mathbf{k}, \qquad \ell = -\mathbf{v}^\top A\mathbf{k}$$
$$\nabla_A \ell = -\mathbf{v}\mathbf{k}^\top, \qquad A' = A + \eta \mathbf{v}\mathbf{k}^\top$$
$$A'\mathbf{u} = A\mathbf{u} + \eta \mathbf{v}(\mathbf{k}^\top \mathbf{u})$$

- Matching the key increases the contribution of its associated value.
- Repeated related updates can accumulate signal or interference.
- Actual MoNe uses nonlinear memory, momentum, and renormalization.

Teaching derivation from the form of Eq. (7), PDF p. 4. This linear example is not the implemented MoNe update.

Backup: the SwiGLU write gradient stays inside the MLP

Define $a = \mathbf{W}_{\text{in}} k$, $b = \mathbf{W}_{\text{gate}} k$,
$u = \text{SiLU}(a) \odot b$, and $z = \mathbf{W}_{\text{out}} v$.

$$\ell = -v^\top \mathbf{W}_{\text{out}}^\top u = -z^\top u$$

$$\begin{aligned}
\nabla_{\mathbf{W}_{\text{out}}} \ell &= -uv^\top, \\
\nabla_{\mathbf{W}_{\text{in}}} \ell &= -[z \odot b \odot \text{SiLU}'(a)] k^\top, \\
\nabla_{\mathbf{W}_{\text{gate}}} \ell &= -[z \odot \text{SiLU}(a)] k^\top.
\end{aligned}$$

The segment's keys and values are treated as fixed local inputs.
Learned rates, momentum, and weight normalization act around this gradient.

Derivation for one head/token from Eqs. (3) and (7), PDF pp. 3–4; update modifiers in Appendix A.2, PDF p. 10.

Backup: offline training configuration

Component	Reported value
Backbone	Qwen2.5-0.5B-Instruct, frozen
Memory	SwiGLU; 4 heads; 224 dimensions/head
LoRA	Rank 128; $\alpha = 128$
Optimizer	AdamW; $\beta_1 = 0.9$, $\beta_2 = 0.95$; decay 0.1
Learning rate	10^{-3} non-LoRA; 5×10^{-5} LoRA
Training	1 epoch; batch 16; bf16
Schedule	200 warmup steps; cosine to 10^{-5}
Gradient clipping	1.0

Paper, Appendix A.2, PDF p. 10. These are offline optimizer settings, not the test-time memory rule.

Backup: details that require clarification for reproduction

- **Memory entry count.**
  Eq. (4) generates a read per query token; Sec. 3.2 also states a fixed T-entry memory KV. The slot-construction step is unspecified.
- **Update recurrence and tensor layout.**
  Appendix A.2 describes rates and momentum activations, but does not fully spell out their recurrence or all head/projection reshapes.
- **Cost protocol.**
  Answering costs differ (0.86 T vs. 0.64 T); output-length and hardware details are insufficient to infer a latency speedup.

Paper, Sec. 3.2, PDF pp. 3–4; Table 2, PDF p. 5; Appendix A.2, PDF p. 10.