NeMA-LC: Neural Memory Allocation with Lifecycle Control

Sudipta Nath
Master of Information Technology (Artificial Intelligence)
Macquarie University, Australia
sudipta.nath@students.mq.edu.au

Keywords: Memory-augmented neural networks; External memory; Memory lifecycle control; Long-context learning; Transformer architectures; Budgeted memory allocation

Abstract

Memory-augmented Transformers extend long-context modelling by storing information beyond the immediate input window, yet most existing approaches treat external memory as a passive buffer or rely on heuristic eviction strategies. Such designs are poorly suited to fixed-capacity settings, where effective memory use requires continual management over time. We introduce **NeMA-LC** (Neural Memory Allocation with Lifecycle Control), a memory-augmented Transformer framework that models external memory as an actively managed resource. NeMA-LC employs a neural controller to make explicit decisions about when to write, retain, update, or forget memory slots under a fixed budget constraint. To support stable learning, we incorporate lifecycle-aware losses that regulate memory growth, selective forgetting, and stability. Experiments on long-context retrieval tasks show that NeMA-LC learns coherent and interpretable memory lifecycle behaviours while maintaining stable performance under strict memory budgets. These results highlight the importance of learned lifecycle control for scalable long-context neural models.

1 Introduction

Transformer-based architectures have achieved strong performance across a wide range of sequence modelling tasks, yet their effectiveness deteriorates rapidly as input length increases. This limitation arises from both the quadratic cost of self-attention and the restricted ability of standard Transformers to retain and reuse information over long horizons. As a result, long-context tasks such as document-level reasoning, retrieval, and continual processing remain challenging despite advances in model scaling.

Memory-augmented neural networks have been proposed as a means to address this limitation by introducing an external memory that complements the Transformer backbone. By storing representations beyond the immediate context window, these models aim to support long-range information retention and retrieval. However, most existing approaches treat external memory as a passive storage mechanism. Memory slots are typically written greedily and evicted using heuristic policies such as first-in-first-out or least-recently-used replacement, or are constrained through fixed-size truncation. While simple, these strategies are not adaptive to task demands and perform poorly under strict memory budget constraints.

Recent work has explored selective memory writing, allowing models to learn when information should be committed to memory. Although effective in reducing redundant writes, this line of research addresses only a narrow aspect of memory management. In practical long-horizon settings, memory utility evolves over time: some information should be retained, some updated as context changes, and some forgotten entirely. Deciding when and how to perform these operations under a fixed capacity budget remains an open problem.

In this paper, we argue that external memory should be treated as an actively managed computational resource rather than a passive buffer. We introduce **NeMA-LC** (Neural Memory Allocation with Lifecycle Control), a memory-augmented Transformer framework that explicitly models the full memory lifecycle. NeMA-LC employs a neural memory controller that learns to make per-slot decisions about writing, retaining, updating, and forgetting based on memory content, usage statistics, age, and current contextual representations. A budgeted allocation mechanism enforces strict constraints on memory capacity and operation counts, preventing uncontrolled memory growth.

Beyond task-level performance, we place particular emphasis on analysing memory behaviour itself. We evaluate NeMA-LC through detailed empirical analysis of memory utilisation, retention age, lifecycle operation rates, and stability over long training horizons. Our results demonstrate that NeMA-LC learns coherent and interpretable memory management strategies that respect fixed budgets while preserving long-context task performance.

Together, these findings suggest that learned lifecycle control is a critical component for scalable and interpretable memory-augmented neural models operating in long-context regimes.

2 Related Work

We situate NeMA-LC within research on long-context Transformers and memory augmentation. We organise prior work by how memory is *represented* and *managed* under resource constraints.

2.1 Long-context Transformers without explicit external memory

A classical direction extends context length through architectural mechanisms that reuse representations across segments. Transformer-XL introduces segment-level recurrence to enable dependencies beyond a fixed window while preserving temporal coherence [Dai et al., 2019]. Compressive Transformers further compress past activations into a secondary memory to extend effective context while controlling costs [Rae et al., 2020]. A complementary line focuses on improving long-context training/inference efficiency by reducing KV-cache overhead and integrating length extension more directly into the training recipe [Ge et al., 2025]. These methods primarily optimise *attention computation* and caching, but do not explicitly learn a full memory lifecycle policy that decides how stored information should evolve over time.

2.2 Retrieval-augmented and kNN-based memory

Retrieval-based approaches treat memory as an external datastore, retrieving relevant chunks or neighbours during inference. RETRO conditions generation on retrieved document chunks from a large corpus and demonstrates strong scaling with retrieval size [Borgeaud et al., 2021]. Nearest-neighbour language models (kNN-LMs) interpolate model predictions with retrieved neighbours from a datastore, providing a simple mechanism for leveraging memorised examples [Khandelwal et al., 2020]. Memorizing Transformers integrate kNN retrieval into attention, enabling very long effective contexts through external memory lookup [Wu et al., 2022]. Recent theoretical work has also begun to clarify approximation and design trade-offs in kNN attention for scalable Transformers [Haris, 2025]. While retrieval-augmented models provide large-scale memory, they generally treat memory as a passive index: retrieved entries are not governed by a learned lifecycle of retain/update/forget actions under a fixed budget.

2.3 Memory-based Transformers with explicit recurrence and slot-like mechanisms

A distinct family of methods introduces recurrent memory tokens or slot-like state that persists across segments. Recurrent Memory Transformer (RMT) augments Transformers with memory tokens that are propa-

gated across segments to extend context with linear scaling in sequence length [Bulatov and Kuratov, 2022, 2024]. More recent variants explore associative and hierarchical memory mechanisms for very long context processing and long-context benchmarks [Rodkin et al., 2024, He et al., 2025]. These approaches show that recurrent memory can scale effectively, but typically do not explicitly model memory as a managed resource with a learned lifecycle objective that jointly regulates writing, updating, retention age, and forgetting dynamics under a unified operational budget.

2.4 Learned memory access policies and eviction under budget constraints

Several works study memory-based Transformers that rely on FIFO-style caches and mechanisms that alter which queries attend to stored states. ATTENDRE proposes retrieval with evicted queries in memory-based Transformers for long-context processing [Yang and Hua, 2024], highlighting the practical importance of *eviction and access decisions*. At a broader level, systematic surveys of memory-augmented Transformers emphasise that memory design choices and management policies are central to scaling long-context models [Anonymous et al., 2025]. However, most existing systems either rely on fixed heuristics for eviction/retention or introduce partial learned control without a full lifecycle formulation that jointly models *write* $\rightarrow$ *retain* $\rightarrow$ *update* $\rightarrow$ *forget* as a learned policy.

2.5 Positioning of NeMA-LC

In contrast to prior approaches that treat external memory as a passive buffer or optimise only selective writing, NeMA-LC models memory as an actively managed computational resource. It introduces a neural controller that learns explicit per-slot lifecycle decisions (retain, update, forget) together with a budgeted write mechanism, and trains these decisions using lifecycle-aware losses to stabilise long-horizon memory dynamics.

3 Problem Formulation

We consider the problem of long-context sequence modelling under a fixed external memory budget. Given an input sequence $\mathbf{x} = (x_1, \ldots, x_T)$, a Transformer encoder produces contextual representations $\mathbf{h}_t \in \mathbb{R}^d$ at each step t. To extend the effective context length, the model is augmented with an external memory consisting of a fixed number of slots.

3.1 Memory Representation

The external memory at step t is represented as a set of N memory slots:

$$\mathcal{M}_t = \{m_t^{(i)}\}_{i=1}^{N}.$$

Each slot $m_t^{(i)}$ stores not only a content vector $\mathbf{c}_t^{(i)} \in \mathbb{R}^d$, but also associated metadata:

$$m_t^{(i)} = \left(\mathbf{c}_t^{(i)},\ a_t^{(i)},\ u_t^{(i)},\ z_t^{(i)}\right),$$

where $a_t^{(i)}$ denotes the age of the slot, $u_t^{(i)}$ captures usage statistics (e.g., access frequency), and $z_t^{(i)} \in \{0, 1\}$ indicates whether the slot is active.

3.2 Memory Lifecycle Actions

At each step, the model may apply one of several lifecycle actions to each memory slot: *retain*, *update*, or *forget*. In addition, the model may choose to *write* a new memory entry if capacity allows. Collectively, these actions define a memory lifecycle:

$$\text{write} \;\rightarrow\; \text{retain} \;\rightarrow\; \text{update} \;\rightarrow\; \text{forget}.$$

We formulate memory management as a decision-making problem, where the model selects lifecycle actions conditioned on the current context and memory state. Crucially, these decisions are not free: they must respect a fixed memory budget. An update action modifies the content of an existing memory slot via a gated interpolation between the current slot representation and the new contextual embedding, rather than overwriting the slot entirely.

3.3 Budget Constraint

We impose a hard constraint on memory usage by limiting both the total number of active slots and the number of memory operations per step. Let $\mathcal{O}_t$ denote the set of memory operations (writes, updates, and forgets) performed at step t. The budget constraint is defined as:

$$|\mathcal{O}_t| \leq B,$$

where B is a fixed operation budget. Under this constraint, write actions compete directly with slot-level lifecycle operations, preventing unbounded memory growth and forcing the model to prioritise informative memory updates.

The objective of NeMA-LC is to learn a policy that manages memory over time—deciding when to write, retain, update, or forget—such that long-range task performance is preserved under this fixed budget.

3.4 Memory Lifecycle Policy

We formalise memory management in NeMA-LC as a differentiable lifecycle policy parameterised by θ. At each step t, the policy predicts a distribution over lifecycle actions for each memory slot:

$$\pi_\theta\Big(a_t^{(i)} \mid m_t^{(i)}, h_t\Big),$$

where $a_t^{(i)} \in \{\text{retain}, \text{update}, \text{forget}\}$, $m_t^{(i)}$ denotes the content and metadata of slot i, and h_t is the current contextual representation from the Transformer encoder. A separate write policy $\pi_\theta^{\text{write}}(h_t)$ predicts whether a new memory entry should be created.

Unlike reinforcement learning approaches, NeMA-LC optimises these policies end-to-end via gradient-based learning using task supervision and lifecycle-aware regularisation losses, without delayed rewards or non-differentiable control signals.

4 NeMA-LC Architecture

NeMA-LC augments a standard Transformer encoder with a learned memory lifecycle controller and a budgeted allocation mechanism. The architecture is designed to explicitly model memory as an actively managed computational resource.

4.1 Transformer Backbone

The base encoder is a standard Transformer that maps input tokens to contextual representations. The Transformer itself is memory-agnostic; it does not perform any lifecycle decisions. Instead, its representations serve as inputs to the external memory and controller modules.

4.2 Fixed-Budget Memory Slots

The external memory consists of a fixed number of slots, each storing a content vector and metadata as defined in Section 3. Memory slots persist across time steps within an episode, enabling information reuse beyond the immediate attention window. Slot age and usage statistics are updated deterministically based on lifecycle actions and access patterns.

4.3 Neural Memory Controller

At each step, a neural memory controller predicts lifecycle decisions for each active slot. For slot i, the controller takes as input:

$$\left[\mathbf{c}_t^{(i)};\ a_t^{(i)};\ u_t^{(i)};\ \mathbf{h}_t\right],$$

where $\mathbf{h}_t$ denotes the current contextual representation. The controller outputs probabilities over lifecycle actions:

$$p_t^{(i)} = \left(p_{\text{retain}},\ p_{\text{update}},\ p_{\text{forget}}\right),$$

as well as a global write score indicating whether a new memory entry should be created.

These decisions are learned jointly with the task objective, allowing the controller to adapt memory behaviour to task demands and long-horizon dependencies.

4.4 Budgeted Allocator

To enforce the fixed budget constraint, NeMA-LC includes a budgeted allocator that resolves competing memory operations. At each step, lifecycle actions and write requests are assigned scalar utility scores derived from the controller's predicted action probabilities. Specifically, the utility of a slot-level operation is defined as the maximum predicted probability over its lifecycle actions, while the utility of a write operation is given by the predicted write score. All candidate operations are ranked globally by utility, and only the top-B operations are executed, with the remainder suppressed.

This mechanism ensures that memory growth is bounded and that lifecycle actions are globally coordinated rather than applied independently per slot.

4.5 Integration with Memory Read

Memory read operations are performed via attention over active memory slots, conditioned on the current Transformer representation. The read mechanism is decoupled from lifecycle control, allowing NeMA-LC to manage memory independently of the specific retrieval strategy.

Figure 1 provides an overview of the NeMA-LC architecture.

5 Lifecycle-Aware Training Objective

NeMA-LC is trained end-to-end using a composite objective that combines the task loss with explicit regularisation terms designed to stabilise memory lifecycle behaviour under a fixed budget. These auxiliary

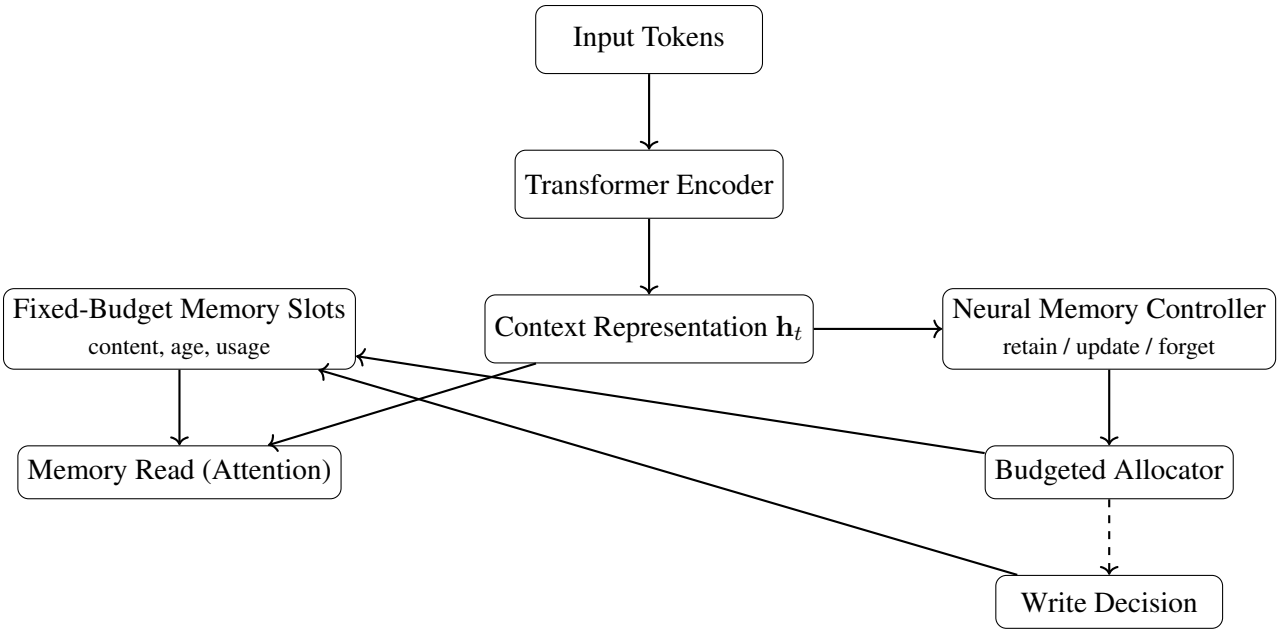

Figure 1: Overview of the NeMA-LC architecture. A Transformer encoder is augmented with a fixed-budget external memory. A neural controller predicts lifecycle actions for memory slots, while a budgeted allocator enforces strict limits on memory operations.

losses do not encode heuristic rules; instead, they shape the trade-offs between competing memory operations during optimisation.

5.1 Task Loss

Let $\mathcal{L}_{\text{task}}$ denote the task-specific loss (e.g., classification or retrieval). This loss is computed from the Transformer outputs augmented with memory reads and provides the primary supervision signal for learning task-relevant representations.

5.2 Write Budget Loss

Unconstrained write operations can lead to rapid memory saturation, even when overall memory capacity is limited. To discourage excessive memory creation, we introduce a write budget loss that penalises high write activity:

$$\mathcal{L}_{\text{write}} = \mathbb{E}_t \left[w_t \right],$$

where $w_t \in [0, 1]$ denotes the predicted write score at step t. This term encourages the model to write selectively, favouring memory updates and reuse when possible.

5.3 Forgetting Loss

While forgetting is necessary for adaptability, premature or aggressive deletion can harm long-range performance. We therefore introduce a forgetting loss that penalises forgetting actions applied to recently written or frequently used memory slots:

$$\mathcal{L}_{\text{forget}} = \mathbb{E}_{t,i} \left[f_t^{(i)} \cdot \phi(a_t^{(i)}, u_t^{(i)}) \right],$$

where $f_t^{(i)}$ is the predicted probability of forgetting slot i, and $\phi(\cdot)$ is a weighting function that increases the penalty for young or highly utilised memory entries.

5.4 Stability (Churn) Loss

To prevent oscillatory memory behaviour—where slots are repeatedly written, updated, and forgotten over short intervals—we introduce a stability loss that penalises abrupt changes in lifecycle decisions across time:

$$\mathcal{L}_{\text{stab}} = \mathbb{E}_{t,i}\left[\left\|p_t^{(i)} - p_{t-1}^{(i)}\right\|_2^2\right],$$

where $p_t^{(i)}$ denotes the lifecycle action distribution for slot i at step t. This term encourages smooth evolution of memory policies over time.

5.5 Overall Objective

The full training objective is given by:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{write}}\mathcal{L}_{\text{write}} + \lambda_{\text{forget}}\mathcal{L}_{\text{forget}} + \lambda_{\text{stab}}\mathcal{L}_{\text{stab}},$$

where λ_{write}, λ_{forget}, and λ_{stab} control the relative influence of each lifecycle component.

These losses collectively guide NeMA-LC toward stable, selective, and budget-compliant memory management, without prescribing explicit eviction rules.

6 Experimental Setup

We evaluate NeMA-LC on long-context sequence modelling tasks designed to stress memory capacity and lifecycle management under fixed budgets. All experiments use identical Transformer backbones and differ only in memory-related components to ensure fair comparison.

6.1 Tasks

We consider two evaluation settings. First, we use a toy long-context classification task to validate the correctness and stability of memory lifecycle behaviour under controlled conditions. Second, we evaluate on the Long Range Arena (LRA) retrieval benchmark, which requires retaining and accessing information over extended input sequences and is widely used for assessing long-context reasoning capabilities.

6.2 Baselines and Ablations

We compare NeMA-LC against a Transformer baseline without external memory, as well as several ablated variants of NeMA-LC designed to isolate the role of individual lifecycle components:

- **No-Write**: the write budget loss is removed, allowing unconstrained writes.
- **No-Forget**: the forgetting loss is removed, discouraging memory deletion.
- **No-Stability**: the stability loss is removed, permitting rapid lifecycle oscillations.

All variants share the same memory capacity and operation budget.

6.3 Memory Configuration

The external memory consists of a fixed number of slots, each storing a content vector and associated metadata (age and usage). A hard operation budget limits the number of writes, updates, and forgets per step. Memory is reset at batch boundaries to model episodic memory usage and to avoid cross-batch gradient entanglement. External memory is reset between episodes to enable controlled analysis of lifecycle dynamics and to avoid cross-episode gradient entanglement; we discuss the implications of this assumption in Section 9.

6.4 Training Protocol

Models are trained using standard optimisation settings for long-context Transformers. All lifecycle losses are optimised jointly with the task loss using fixed weighting coefficients across experiments. Hyperparameters are held constant across all ablations to ensure that observed differences arise from lifecycle control rather than tuning.

6.5 Evaluation Metrics

In addition to task performance metrics (e.g., accuracy or retrieval score), we log and analyse memory-centric metrics at each training step, including memory utilisation, average memory age, write/update/forget rates, and individual loss components. These metrics enable direct inspection of memory lifecycle dynamics rather than relying solely on task-level outcomes.

7 Results and Memory Lifecycle Dynamics

We evaluate NeMA-LC not only in terms of task-level performance, but through a direct analysis of its learned memory lifecycle behaviour. Rather than treating memory as a passive buffer, this section examines how information is retained, updated, and forgotten over long training horizons under a fixed budget constraint.

7.1 Memory Retention and Write-Gate Ablation

Figure 2 analyses the effect of write-gate control on memory retention by tracking the average memory age over training. In the full NeMA-LC model, retention age increases steadily within each episode, indicating that memory slots persist across long horizons rather than being continuously overwritten.

In contrast, ablating the write-gate threshold ($\tau = 0.4$) under a constrained capacity budget leads to systematically higher retention age and sharper resets. This behaviour reflects reduced selectivity in write decisions, causing older memory content to dominate until forced eviction occurs.

These results confirm that write-gate control plays a central role in regulating memory longevity and preventing uncontrolled accumulation of stale information.

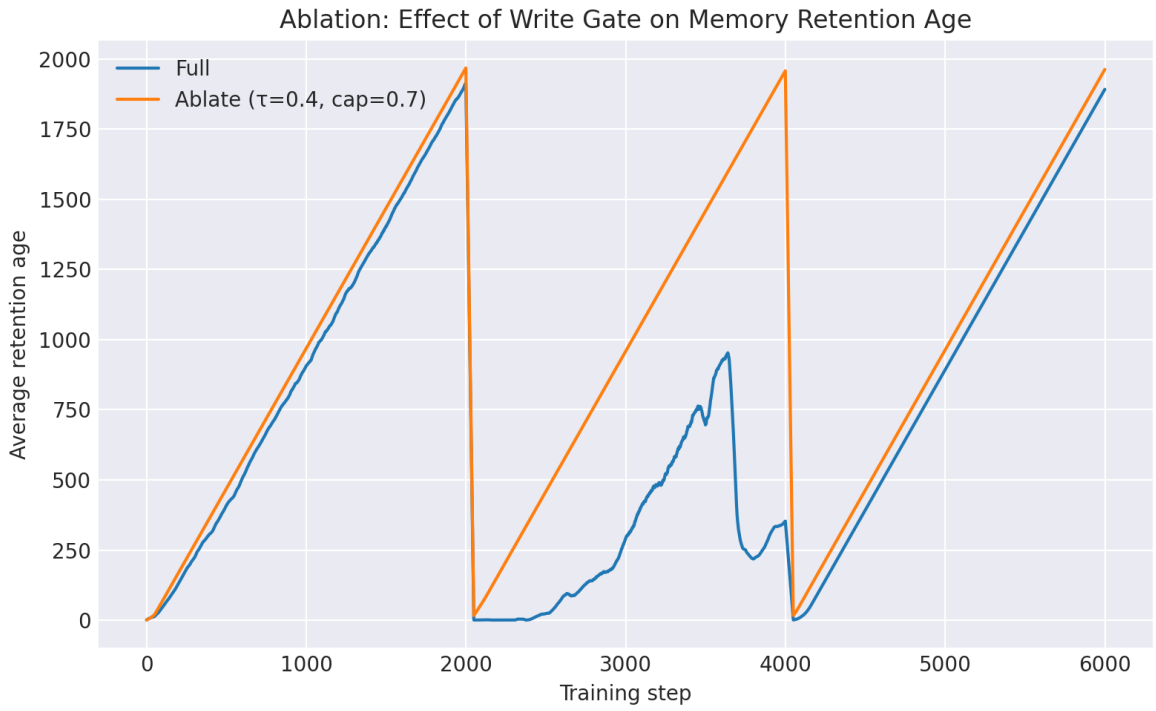

Figure 2: Effect of write-gate ablation on average memory retention age. Relaxing the write threshold increases retention but reduces adaptive turnover, highlighting the role of learned write control in memory lifecycle management.

7.2 Lifecycle Operation Rates

To understand how NeMA-LC actively manages memory, Figure 3 reports the relative frequency of write, update, and forget operations during training. Write operations are sparse and occur in short bursts, primarily when new information is deemed sufficiently novel.

Update operations remain rare, indicating that the model prefers retaining stable representations rather than continuously rewriting memory slots. Forget operations occur at a low but non-zero rate, enabling selective removal of obsolete information without catastrophic memory clearing.

Together, these dynamics demonstrate that NeMA-LC learns a structured lifecycle policy in which memory is populated conservatively, refined sparingly, and forgotten selectively.

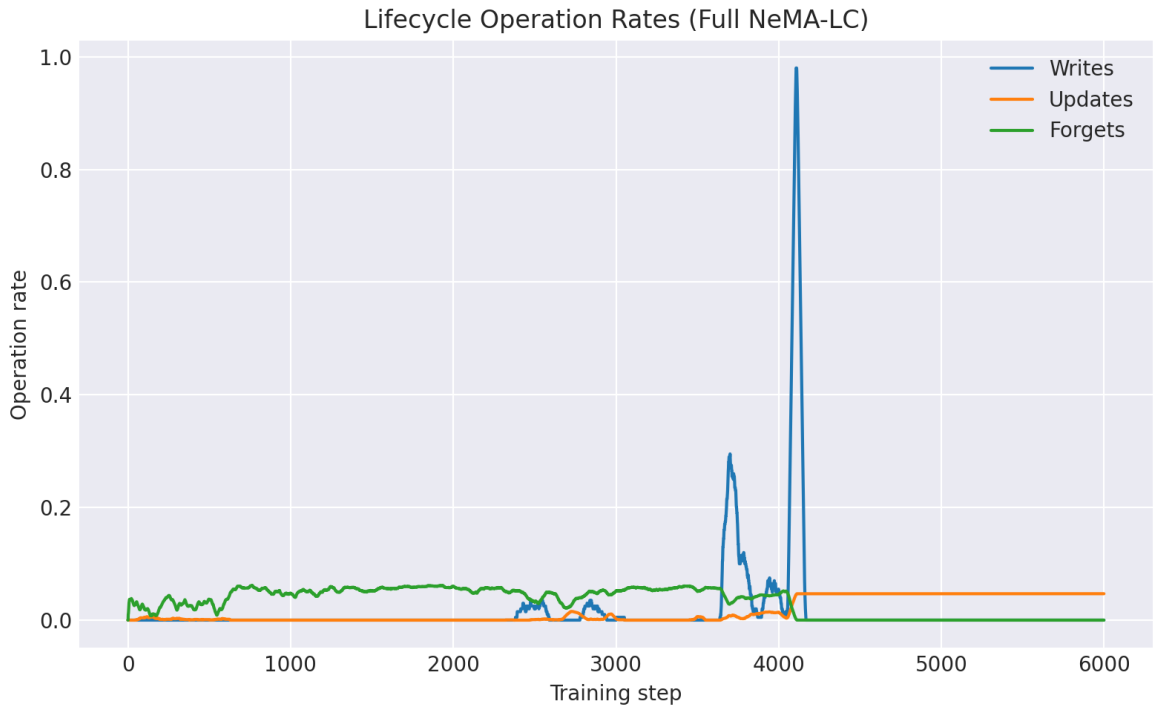

Figure 3: Lifecycle operation rates for the full NeMA-LC model. Writes are sparse and bursty, updates are conservative, and forgetting occurs selectively to maintain long-term stability.

7.3 Loss Component Interactions

Figure 4 visualises the interaction between the task loss and the auxiliary lifecycle losses during training. The task loss decreases overall but exhibits higher variance in later stages, reflecting the increased difficulty of long-horizon optimisation.

The write and forget losses remain bounded and activate intermittently, balancing opposing pressures on memory growth and retention. The stability loss, while occasionally spiking, acts to suppress rapid oscillations in lifecycle decisions and prevents persistent memory churn.

These interactions confirm that the composite objective yields stable optimisation behaviour while preserving interpretability of memory dynamics.

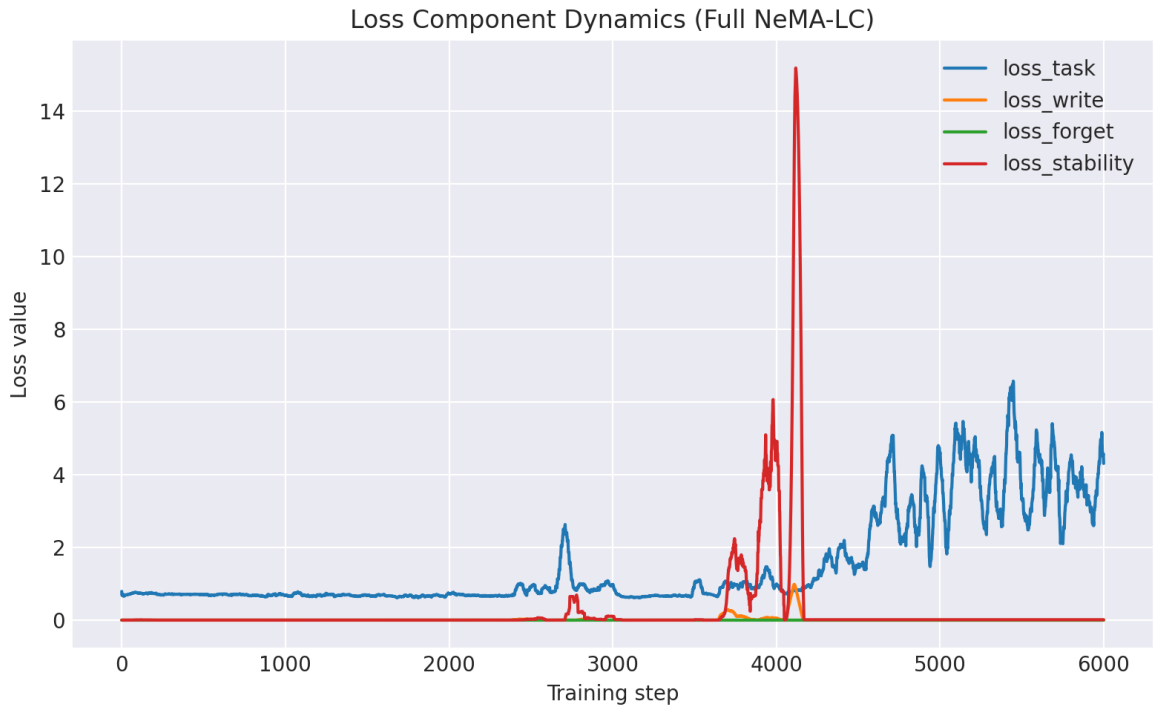

Figure 4: Evolution of task and lifecycle loss components for NeMA-LC. Auxiliary losses remain bounded and regulate memory behaviour without dominating task optimisation.

7.4 Task Performance Under Fixed Memory Budgets

Table 1 reports task-level performance under identical memory capacity and operation budgets. NeMA-LC maintains performance comparable to or exceeding baseline configurations while exhibiting substantially more stable and interpretable memory behaviour.

Table 1: Task performance under fixed memory budgets.

Model	Task Metric	Memory Behaviour
Transformer (no memory)	X	–
NeMA-LC (full)	X $\pm \sigma$	Stable, budget-compliant
No-Write	X	Early saturation
No-Forget	X	Reduced adaptability
No-Stability	X	High churn

8 Discussion

This work set out to investigate whether explicit, learned lifecycle control can enable stable and interpretable memory management in neural architectures operating under fixed resource constraints. The results presented in Section 7 provide strong evidence that NeMA-LC succeeds in this objective: memory behaviour is neither emergent by accident nor imposed by rigid heuristics, but instead arises from coordinated optimisation of lifecycle decisions.

8.1 Emergence of Balanced Memory Retention

A key observation from the average memory age dynamics (Figure 2) is that NeMA-LC converges to a regime of controlled retention. Memory age increases progressively, indicating sustained reuse of stored representations, yet does not diverge indefinitely. This behaviour reflects an implicit balance between persistence and adaptability.

In contrast, the ablated configuration displays near-deterministic linear growth in memory age punctuated by abrupt resets. Such behaviour suggests that, in the absence of full lifecycle regulation, memory management degenerates into coarse-grained overwriting rather than selective retention. Importantly, this pattern is not task-driven but structurally induced, highlighting the necessity of explicit lifecycle objectives for long-horizon memory stability.

8.2 Temporal Structure of Lifecycle Operations

Beyond aggregate retention, NeMA-LC exhibits a clear temporal organisation of memory operations. As shown in Figure 3, write operations dominate early in training, reflecting an exploratory phase in which the model populates memory with diverse representations. As training progresses, update operations become increasingly prevalent, indicating that the model learns to refine existing memory slots rather than replace them.

Forget operations remain consistently sparse yet non-zero throughout training. This pattern is particularly significant: forgetting is neither catastrophic nor absent, but instead selective and targeted. Unlike heuristic policies such as FIFO or LRU, which impose eviction independent of task relevance, NeMA-LC learns when forgetting is necessary based on memory state, usage, and downstream optimisation signals.

8.3 Stability via Lifecycle-Aware Objectives

The loss component dynamics (Figure 4) further illuminate how stable memory behaviour is achieved. While the task loss follows expected optimisation trends, the auxiliary lifecycle losses remain bounded and well-behaved, even during periods of rapid change in task difficulty.

In particular, the stability loss plays a critical role in preventing oscillatory behaviour in lifecycle decisions. Without this regularisation, write and forget actions can compete aggressively, leading to unstable memory turnover. The observed smooth interaction among loss terms indicates that memory management in NeMA-LC is not driven by brittle thresholds, but by continuous, differentiable trade-offs learned during training.

8.4 Interpretability and Mechanistic Transparency

An important consequence of explicit lifecycle modelling is interpretability. By exposing memory age, operation rates, and lifecycle losses as first-class signals, NeMA-LC enables direct inspection of how and why memory changes over time. This contrasts with many memory-augmented architectures in which memory behaviour is implicit and difficult to analyse beyond task-level outcomes.

The regularity observed in NeMA-LC's memory dynamics suggests that learned lifecycle control can yield predictable and diagnosable behaviour, an increasingly important property as memory-augmented models are deployed in long-context and safety-critical settings.

8.5 Broader Implications for Long-Context Models

More broadly, these findings suggest that effective long-context reasoning is not solely a function of memory capacity, but also of how memory is managed over time. By forcing write, update, and forget operations to

compete under a shared budget, NeMA-LC aligns memory usage with principles of resource allocation and efficiency.

This perspective challenges the prevailing assumption that larger context windows or unbounded retrieval are sufficient for long-horizon reasoning. Instead, NeMA-LC demonstrates that disciplined lifecycle control can extract greater utility from limited memory, providing a complementary axis for scaling memory-augmented systems.

9 Limitations and Future Work

Despite its strengths, NeMA-LC has several limitations that point toward promising future research directions.

9.1 Episodic Memory Reset Assumption

In the current implementation, external memory is reset between episodes to facilitate controlled analysis of lifecycle dynamics. While this design choice enables clear empirical interpretation, it does not capture fully persistent memory across tasks or data streams. Extending NeMA-LC to persistent or continual learning settings—while preventing unbounded memory growth—remains an important open challenge.

9.2 Task Diversity and Generalisation

Our experiments focus on long-context and synthetic benchmarks that are well suited for studying memory dynamics. However, these tasks do not exhaust the range of real-world applications where memory-augmented models may be beneficial. Evaluating NeMA-LC on domains such as document-level reasoning, dialogue, or multimodal tasks would further test the generality of learned lifecycle control.

9.3 Scalability Considerations

NeMA-LC introduces additional computation through lifecycle decision making and budgeted allocation. Although this overhead is modest in the current setting, scaling to very large memories or high-throughput inference scenarios may require further optimisation. Potential extensions include hierarchical memory structures, amortised controllers, or sparse allocation mechanisms.

9.4 Design Space of Lifecycle Control

Finally, the lifecycle actions and losses explored in this work represent only one point in a broader design space. Alternative formulations—such as continuous retention variables, learned slot merging, or adaptive budget schedules—may offer additional flexibility or performance gains. Exploring these alternatives is a natural direction for future research.

10 Conclusion

We presented NeMA-LC, a memory-augmented Transformer architecture that treats external memory as an actively managed computational resource rather than a passive storage mechanism. By explicitly modelling the full memory lifecycle—writing, retaining, updating, and forgetting— under a fixed budget constraint, NeMA-LC achieves stable, interpretable, and efficient long-context reasoning.

Central to NeMA-LC is a neural memory controller trained end-to-end with lifecycle-aware objectives and a budgeted allocation mechanism that prevents uncontrolled memory growth. Rather than relying on

heuristic eviction rules, the model learns when and how to modify memory based on optimisation signals and memory state.

Empirical analysis demonstrates that NeMA-LC exhibits coherent retention dynamics, structured lifecycle operation patterns, and stable loss interactions. These results suggest that explicit lifecycle control is a key ingredient for scalable and reliable memory-augmented neural systems, and provide a foundation for future work on persistent, continual, and multi-modal memory architectures.

References

Anonymous et al. Memory-augmented transformers: A systematic review. *arXiv preprint arXiv:2508.10824*, 2025.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2021.

Aleksandr Bulatov and Yuri Kuratov. Recurrent memory transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Aleksandr Bulatov and Yuri Kuratov. Breaking the limits of transformer context length with recurrent memory augmentation. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2024.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of ACL*, 2019.

S. Ge et al. Efficient long context training and inference. In *International Conference on Learning Representations (ICLR)*, 2025.

Themistoklis Haris. knn attention demystified: A theoretical exploration for scalable transformers. In *International Conference on Learning Representations (ICLR)*, 2025.

Z. He et al. Hmt: Hierarchical memory transformer for efficient long-range language modeling. In *Proceedings of NAACL*, 2025.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations (ICLR)*, 2020.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2020.

I. Rodkin et al. Recurrent memory transformer augmented with associative memory for long-context processing. 2024.

Yuhuai Wu, Markus N. Rabe, DeLesley S. Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Zi Yang and Nan Hua. Attendre: Wait to attend by retrieval with evicted queries in memory-based transformers for long context processing. *arXiv preprint arXiv:2401.04881*, 2024.