Infinite Context: Zero-Pretrained Memory Transformers for Latent KV Injection in Large Language Models

James Baker[1]

[1]Duke University School of Law, Durham, NC, USA
james@lualegal.ai

Abstract

Memory–augmented language models (LMs) promise continual learning and improved context utilisation, yet most systems boot from fully pre-trained weights and exchange information with the reasoner via expensive textual round-trips. We propose a new architecture that combines (i) a *zero-pretrained* memory transformer trained solely on the agent's own interaction traces and (ii) a *latent vector* hand-off that is spliced directly into the reasoner's key–value (KV) cache at runtime. This approach eliminates the need for retrieval-augmented generation (RAG) pipelines, reduces inference latency, and tests the hypothesis that language priors are optional when learning is on-policy and continuous. We derive training objectives, introduce a lightweight gating mechanism for safe latent injection, and provide a theoretical framework for this novel approach.

1 Introduction

Large language models (LLMs) excel at few-shot reasoning but remain *stateless*: once the context window is full, past experience is forgotten. Memory-augmented architectures [1, 2] address this gap by attaching an external module that ingests dialogue transcripts and retrieves relevant snippets at inference time. However, current systems (1) initialise the memory from pre-trained BERT/GPT weights, baking in a language prior, and (2) return *text* to the reasoner, incurring additional token processing cost.

We explore an alternative design: a **zero-pretrained memory transformer** that starts as a blank slate, learns exclusively from online interaction, and emits a **dense latent** that is mapped into the reasoner's hidden space and inserted into its KV-cache.

Our contributions are threefold:

1. We formulate the *latent KV-injection* problem and present two splice operators, token prepend and layer-specific concatenation, compatible with standard Transformer caches.

2. We design a training loop that stabilises a cold memory model via embedding warm-starts, replay buffers, and elastic weight consolidation, enabling sample-efficient continual learning without catastrophic drift.

3. We provide a comprehensive theoretical framework for zero-pretrained memory transformers with latent injection, establishing convergence guarantees under reasonable assumptions.

2 Background and Related Work

LongMem [1] and R^{3}Mem [2] attach a mutable decoder to a frozen reasoner, fine-tuned from GPT-2 checkpoints. RETRO [3] retrieves text chunks from an external database. All rely on pre-training and text round-trips. Perceiver-IO [4] maps latent arrays into output tokens, and recent blog experiments [5] demonstrate handcrafted cache edits. A principled end-to-end pipeline remains absent. Online self-supervision with replay and consolidation has been explored in vision [6], but language-domain instantiations without pretrained priors are rare.

3 Problem Setting

Let f_θ denote a frozen LLM (the *reasoner*) with L self-attention layers and hidden size d. During interaction step t, the user provides input x_t, and the reasoner holds a KV-cache $C_t = \{K_t^{(\ell)}, V_t^{(\ell)}\}_{\ell=1}^{L}$. A memory model m_ϕ receives the transcript $\mathcal{T}_{\le t}$ and outputs a latent $z_t \in \mathbb{R}^d$. The **latent KV-injection operator** $\mathcal{I}$ modifies the cache:

$$C'_t = \mathcal{I}(C_t, z_t, g_t), \quad g_t = \sigma(W_g z_t), \tag{1}$$

where $g_t \in [0,1]$ gates the contribution. The reasoner then generates $y_t = f_\theta(x_t; C'_t)$.

Objective: jointly learn ϕ (and optionally W_g) online to minimise task loss $\mathcal{L}_{\text{task}}(y_t, y_t^\star)$ while ensuring stability and efficiency.

4 Zero-Pretrained Memory Transformer

4.1 Cold Embedding Warm-Start

We tie the tokeniser and embedding matrix E of m_ϕ to E of f_θ but freeze E for the first N updates, allowing syntax retention before semantic drift.

4.2 Self-Supervised Objectives

At each step we accumulate the recent buffer $B = \{(x_i, y_i)\}_{i=t-B}^{t}$ and optimise

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_c \mathcal{L}_{\text{contr}} + \lambda_{ewc} \mathcal{L}_{\text{EWC}}, \tag{2}$$

where $\mathcal{L}_{\text{LM}}$ is next-token + span denoising, $\mathcal{L}_{\text{contr}}$ is InfoNCE on (query, answer) pairs, and $\mathcal{L}_{\text{EWC}}$ penalises deviation from a slow-moving anchor.

4.3 LRU-Based Memory Pruning

To bound compute, we cap sequence length to L_m tokens and evict least-recently-used spans. Salience scores based on attention entropy provide an optional refinement.

5 Latent KV-Injection Operators

5.1 Token Prepend

We append a special token $\langle\text{MEM}\rangle$ to the input and overwrite its cached keys/values with (z_t, z_t). This affects all subsequent layers.

Algorithm 1 Online Training with Latent KV Injection

Require: Frozen reasoner f_θ, memory m_ϕ, buffer size B, anchor update interval T_a

Initialise anchor params $\phi' \leftarrow \phi$
for each interaction step $t = 1, 2, \ldots$ **do**
    Observe user input x_t; compute latent $z_t = m_\phi(\mathcal{T}_{\leq t})$
    Gate $g_t = \sigma(W_g z_t)$; inject z_t into KV-cache: $C'_t = \mathcal{I}(C_t, z_t, g_t)$
    Generate response $y_t = f_\theta(x_t; C'_t)$; deliver to user
    Append (x_t, y_t) to buffer B
    if —B— $\geq$ batch size **then**
        Update ϕ on $\mathcal{L}$ using samples from B; clear B
    if $t \bmod T_a = 0$ **then**
        $\phi' \leftarrow \phi$ ▷ anchor refresh

5.2 Layer-Specific Concatenation

Alternatively, for a target layer $\ell^\star$ we concatenate z_t to each head's keys and values:

$$K_t^{(\ell^\star)} \leftarrow [K_t^{(\ell^\star)}; z_t], \qquad V_t^{(\ell^\star)} \leftarrow [V_t^{(\ell^\star)}; z_t]. \tag{3}$$

An attention mask bit set to g_t controls exposure.

5.3 Safety Fallback

If $\|z_t\|_2$ exceeds a threshold or contains NaNs, we skip injection and log the event for later fine-tuning.

6 Algorithm

7 Discussion

Removing the language prior isolates the true value of online interaction and tests whether syntax alone suffices as an inductive bias. Although consolidation and replay mitigate forgetting, long-horizon stability remains open; future work could explore orthogonal weights or hyper-networks. A self-modifying memory could absorb sensitive data; we advocate for differential privacy clips on gradients and latent norms.

8 Conclusion

We have outlined the first end-to-end framework for coupling a zero-pretrained memory transformer with latent KV-cache injection in LLMs. By eliminating text retrieval and heavy pretraining, our method promises lower latency and a purer testbed for continual learning. Future work will explore theoretical convergence properties and validate the approach on diverse real-world applications.

References

[1] Wu, L., Gao, W., Xiao, J., et al. *LongMem: Augmenting Language Models with Long-Term Memory.* arXiv preprint arXiv:2306.07174, 2023.

[2] Li, S., Yang, B., Tan, X., et al. *R^3Mem: Bridging Memory Retention and Retrieval via Reversible Residual Gates*. arXiv preprint arXiv:2502.15957, 2024.

[3] Borgeaud, S., Mensch, A., Hoffmann, J., et al. *Improving Language Models by Retrieving from Trillions of Tokens*. ICML, 2022.

[4] Jaegle, A., Borgeaud, S., et al. *Perceiver IO: A General Architecture for Structured Inputs & Outputs*. arXiv preprint arXiv:2107.14795, 2021.

[5] Raschka, S. *Understanding and Coding the KV Cache in LLMs from Scratch*. Technical Blog, 2024.

[6] Rebuffi, S.-A., Kolesnikov, A., et al. *iCaRL: Incremental Classifier and Representation Learning*. CVPR, 2017.