

Solving the Clive Wearing Problem: One-Shot Episodic Memory for Frozen Transformers

Tommi Niemi
Rotko Networks
tommi@rotko.net

April 2026

Abstract

We enable frozen transformers to form new memories without gradient descent. The model’s own hidden states are stored as episodic memories; on recall, they bias token generation through direct logit injection. A frozen Qwen 2.5 0.5B taught three novel facts recalls all three at 100% accuracy. No weights are modified. No gradients are computed. Memories persist to disk across sessions. Code and reproduction: <https://git.rotko.net/tommi/epimem>.

1 The Clive Wearing Problem

Clive Wearing lost his hippocampus to encephalitis in 1985. He retained every skill—piano, language, conducting—but could not form a single new memory. Every 7 seconds, he believed he had just woken up for the first time. His diary: “8:31 AM Now I am awake. 8:34 AM Now I am properly awake.” Each entry crossed out moments later.

Current LLMs are Clive Wearing. They possess sophisticated capabilities—reasoning, language, world knowledge—but cannot form new memories. Every conversation starts from zero. The context window is their 7-second span. When it clears, everything is gone.

Fine-tuning modifies weights and causes catastrophic forgetting. RAG re-encodes text into the context window every time—no actual learning occurs. LoRA still requires gradients. In-context learning vanishes when the conversation ends.

We give the frozen model a hippocampus: an external episodic memory that stores hidden-state patterns and replays them to bias future processing. The backbone never changes. It just receives hippocampal input that steers its output toward learned associations.

2 Method

2.1 Architecture

Two components:

Frozen backbone (Qwen 2.5 0.5B, 896-dimensional hidden states): The pretrained transformer. Processes input tokens, produces hidden state vectors. Weights are never modified at any point.

Episodic memory bank: A key-value store where the *key* is the backbone’s hidden state vector at the final token position—the model’s internal representation of the prompt in its own learned space—and the *value* is per-position logit biases for the correct continuation tokens—which token to boost at each generation step.

2.2 Teaching (One Forward Pass)

Given a prompt P and desired answer A :

1. **Extract key:** Run backbone on P . Extract hidden state $\mathbf{h} = \text{backbone}(P)$ at the final token. This 896-dimensional vector encodes the backbone’s understanding of the prompt.
2. **Compute logit biases:** Run backbone on the concatenation $P \parallel A$. At each answer token position i , compute the gap between the correct token’s logit and the maximum logit. The bias overcomes this gap plus a margin:

$$b_i = \max(\max_j \ell_j - \ell_{t_i}, 5.0) + 5.0$$

where ℓ_j are logits at position i and t_i is the correct token. This produces one (t_i, b_i) pair per answer token.

3. **Store:** Save (key = $\mathbf{h}$, value = $\{(t_i, b_i)\}$) to the memory bank.

One forward pass. No iteration. No loss function. No gradients.

2.3 Recall (Similarity Search + Injection)

Given a new query Q :

1. **Extract query key:** $\mathbf{h}_q = \text{backbone}(Q)$ at the final token.
2. **Search:** For each stored episode, compute cosine similarity $\cos(\mathbf{h}_q, \mathbf{h}_{\text{stored}})$. Return the best match above threshold.
3. **Generate with injection:** At generation step i , if the matched episode has a logit bias (t_i, b_i) for step i , add b_i to the backbone’s logit for token t_i before sampling. After all biases are applied, the backbone continues generating freely.

The backbone generates fluent text beyond the taught answer—the logit biases seed the first tokens, and the language model’s coherence completes the sentence naturally.

2.4 Persistence

The memory bank serializes to JSON: each episode stores the 896-dimensional key vector and the list of (t_i, b_i) pairs. Load the file, and all memories are available. No retraining. No warm-up. Instant recall.

2.5 Why Hidden States, Not Text

RAG stores text and re-encodes it. This has three costs: (1) context window consumption—retrieved passages compete with the actual input for attention; (2) re-encoding latency—the backbone must process retrieved text tokens; (3) representation mismatch—the retrieval embedding space (typically a separate encoder) doesn’t match the generative model’s internal space.

Storing hidden states eliminates all three. The memory is already in the backbone’s native representation. The key and query are produced by the same function—cosine similarity is exact (1.000 for identical prompts). Injection is a single scalar addition to one logit per generation step.

3 Experiments

3.1 Setup

- **Backbone:** Qwen 2.5 0.5B (896-dim hidden states)
- **Inference:** PyTorch via HuggingFace `transformers` (also works with ONNX Runtime)
- **Hardware:** Any machine with Python 3 and ~ 2 GB RAM. No GPU required.
- **Gradient computation:** None. At no point—not during teaching, recall, or persistence.

3.2 One-Shot Fact Learning

We teach three facts about “Zyphraxia”—a word absent from Qwen’s training data:

Prompt	Taught	Recalled	Sim.
“The capital of Zyphraxia is”	Novaheim	Novaheim, a city of 100	1.000
“The ruler of Zyphraxia is”	Queen Stellara	Queen Stellara. She is...	1.000
“The currency of Zyphraxia is”	Glimmers	Glimmers. The currency...	1.000

Table 1: One-shot fact recall. All three novel facts recalled correctly with cosine similarity 1.000. The backbone generates fluent continuations beyond the taught answer.

3.3 Persistence

The memory bank is saved to JSON (77 KB for 3 episodes with 896-dim keys). After reloading from disk, all three facts are recalled identically: 3/3 pre-save, 3/3 post-reload.

3.4 Reproduction

```
git clone https://git.rotko.net/tommi/epimem
cd epimem
pip install transformers torch numpy
python python/epimem.py
```

Downloads Qwen 2.5 0.5B from HuggingFace (~ 1 GB, cached after first run). Teaches 3 facts, recalls 6/6 (3 pre-save + 3 post-reload). Runs in ~ 30 seconds after model is cached.

4 Related Work

4.1 Training-Free Episodic Memory

CAMELoT (?) is the closest prior work: a training-free consolidated associative memory for frozen LLMs. It stores key-value pairs from transformer attention layers, retrieves by cosine similarity, and injects as attention prefixes. Our approach differs in what is stored (logit biases vs. KV pairs) and where injection occurs (output logits vs. attention mechanism).

EM-LLM (?) stores KV pairs from attention heads as episodic events, retrieves by k -NN with temporal contiguity, and prepends retrieved pairs into the context window. The backbone is frozen

and no training is required. The key difference: EM-LLM injects at the attention level (KV cache extension), we inject at the output level (logit biases).

Larimar (?) adds episodic memory to frozen LLMs via a memory matrix with pseudo-inverse retrieval. Unlike our approach, Larimar requires training the memory encoder/decoder with a variational objective.

4.2 Retrieval-Augmented Generation

RAG (?) retrieves text passages and inserts them into the context window. The model re-encodes retrieved text each time. We store hidden states and inject logit biases—no re-encoding, no context consumption, no attention cost.

4.3 Knowledge Editing

ROME (?) and MEMIT (?) edit factual associations by modifying specific weight matrices via rank-one updates. Our method makes zero modifications to any weight.

4.4 What Distinguishes This Work

All prior training-free episodic memory systems inject at the attention level—modifying KV caches, prepending context, or adding cross-attention. We inject at the logit level: the retrieved memory directly steers which tokens are generated, without touching the model’s internal representations. This is simpler (one scalar addition per token per step), cheaper (no attention recomputation), and more interpretable (the bias values directly indicate how strongly each token is boosted).

5 Limitations

Backbone lock-in. Memories are tied to the specific backbone. Changing the model invalidates all stored keys. Migration requires re-encoding through the new backbone.

Key collision. Semantically different prompts with similar hidden states may trigger incorrect recall. A similarity threshold mitigates this but doesn’t eliminate it.

Linear scan. Retrieval is $O(n)$ over stored episodes. For banks exceeding $\sim 100K$ episodes, approximate nearest neighbor indexing would be needed.

Per-position biases. The current implementation stores biases per generation step. This is simple but doesn’t generalize to variable-length reformulations of the same answer.

6 Conclusion

Frozen transformers cannot form new memories. We give them a hippocampus.

The method is minimal: store the backbone’s own hidden state as a key, store logit biases as a value, retrieve by cosine similarity, inject during generation. No gradients. No weight changes. No training loop. One forward pass to teach. One lookup to recall. Memories persist to disk.

The 200-line Python implementation reproduces the full result. The Clive Wearing Problem—intelligent systems that cannot form new memories—has a working solution.

References

- Das, P., Natarajan, S., Singh, S., et al. Larimar: Large Language Models with Episodic Memory Control. *ICML*, 2024. arXiv:2403.11901.
- Fountas, Z., Bisk, Y., et al. Human-inspired Episodic Memory for Infinite Context LLMs. arXiv:2407.09450, 2024.
- Graves, A., Wayne, G., and Danihelka, I. Neural Turing Machines. arXiv:1410.5401, 2014.
- Graves, A., Wayne, G., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.
- Jang, J., et al. CAMELoT: Towards Large Language Models with Training-Free Consolidated Associative Memory. arXiv:2402.13449, 2024.
- Lewis, P., Perez, E., et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*, 2020.
- Meng, K., Bau, D., Mitchell, A., and Finn, C. Locating and Editing Factual Associations in GPT. *NeurIPS*, 2022.
- Meng, K., Sharma, A., Andonian, A., et al. Mass-Editing Memory in a Transformer. *ICLR*, 2023.