Enabling Explicit Memory in Pretrained LLMs for Enhanced Reasoning and Efficiency

Junzhi Chen [1] Dipesh Tharu Mahato [1] Xinran Lin [1] Akshay Nuthanapati [1]

[1]New York University

Background & Motivation

Problem: Read/Write Cost Trade-off

- **Model Parameters**: Expensive to update
- **External Documents (RAG)**: High decoding cost and context fragmentation

Solution: Explicit Memory
Capable of capturing specific factual knowledge as key-value attention pairs, enabling more efficient inference without retraining the full model.

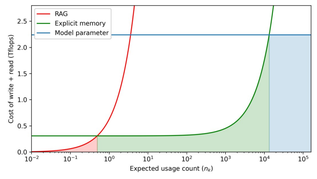


Figure 1. The total cost (TFlops) of writing and reading a piece of knowledge by 2.4B model.

Approaches

Memory Design

- **Explicit Memory**: Key-value pairs for factual knowledge of first 8/16 transformer layers
- **Dual Operation Modes**:
  - **Warm Start**: Preload and cache explicit memories before inference
  - **Cold Start**: Convert to memory on first retrieval (saves storage)

Inference Process

- **Memory Retrieval**: Embed reference/query using BGE-M3 and search via FAISS IVFPQ
- **Memory Reading**: In M3 model, explicit memory is injected as sparse attention key-value pairs into select attention heads.
  The standard attention output at layer l, head h, for token i is:

$$Y_i^{l,h} = \text{softmax}\left(\frac{Q_i^{l,h}(K_{\leq i}^{l,h})^T}{\sqrt{d_h}}\right)V_{\leq i}^{l,h}W_O^{l,h} \tag{1}$$

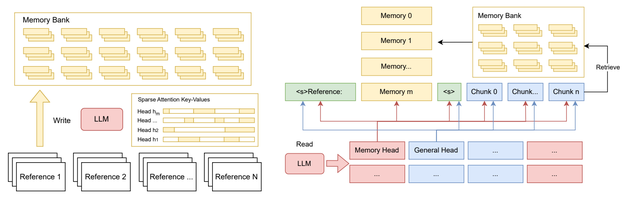
where $Q_i^{l,h} = X_i^{l,h}W_Q^{l,h}$ and $K_{\leq i}^{l,h}, V_{\leq i}^{l,h}$ are key-values from context tokens.
To integrate explicit memory, we modify the output to:

$$Y_i^{l,h} = \alpha_i^{l,h} \cdot [V_0^{l,h}, V_1^{l,h}, \ldots, V_4^{l,h}, V_{\leq i}^{l,h}]W_O^{l,h} \tag{2}$$

with attention weights:

$$\alpha_i^{l,h} = \text{softmax}\left(\frac{Q_i^{l,h} \cdot [K_0^{l,h}, \ldots, K_4^{l,h}, K_{\leq i}^{l,h}]^T}{\sqrt{d_h}}\right) \tag{3}$$

Here, $K_j^{l,h}, V_j^{l,h}$ represent key-value pairs from 5 retrieved memory chunks. This mechanism allows the model to attend over both explicit memory and prompt context, without modifying the pretrained architecture.



(a) Memory writing process (b) Memory reading process

Figure 2. Memory read/write mechanisms

Memory Optimization

Sparsification Strategy

- Layer Selection: **First 8/16 layers** (Use only half of the attention layers as memory layers)
- Token Selection: **8/128 tokens per head** (Select top-8 tokens per head by attention weights)
- Head Selection: All key-value heads

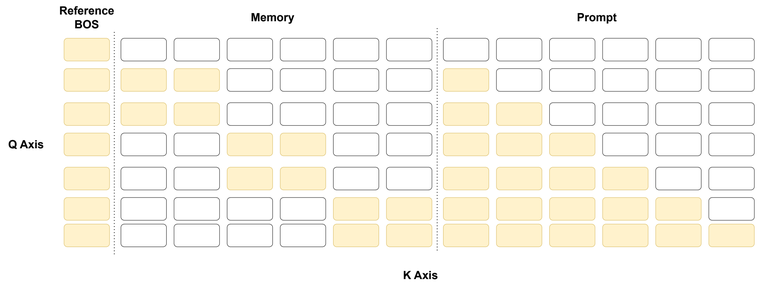


Figure 3. Attention Mask in M3 model preprocess. Assume retrieve 2 memory chunks for every 2 tokens in prompt, and each memory is only in 1 token length. The reference bos can be seen by every token, each memory can only be seen by its corresponding token, and casual mask is used as usual in the prompt part.

Data & Knowledge Base

We curated a high-quality 120k+ sample dataset spanning three domains with a focus on reasoning-enhancing abilities. The data spans three core domains:

- **Math**: MetaMathQA, OpenR1-Math-220k
- **Code**: CodeAlpaca-18k, CodeIO-PyEdu
- **Reasoning**: Capybara

Curation Pipeline

- **MinHash Deduplication:** Removed samples with >80% similarity
- **Rule-Based Filtering:** Filtered short/low-entropy/noisy text (e.g. <50 words, entropy <2 bits)
- **TinyBERT Filtering:** Fine-tuned a TinyBERT binary classifier to filter semantically weak data

We use all of the training data as the reference dataset. The reference dataset is segmented into chunks of 128 token lengths and embedded by BGE-M3. The reference embedding is then added to a FAISS index for retrieval.

Experiments

We choose Llama-3.2-1b instruct as the base model for memory training with following details:

- Retrieve 5 memory chunks every 64 tokens (each chunk: 128 tokens → sparsified to 8).
- Knowledge base: full training data + GSM8K, MATH, MBPP.
- Retrieval via IVF + Product Quantization: `n_list = 512`, `m = 16`, `nbits = 8`, `nprobe = 32`.
- Training: LR = `1e-5`, batch size = 4, grad acc = 4, epochs = 2, warm-up = 10%, cosine LR scheduler.

After training, we test the model performance on the following benchmarks with evaluation settings:

- **HumanEval:** 0-shot, pass@1, max length 1024
- **GSM8K:** 8-shot, CoT prompting (maj@1), max length 1024
- **MBPP:** 3-shot, generative, pass@1, max length 256
- **MATH (Hard):** 0-shot, CoT, SymPy-based exact match, max length 5120

Results

Backbone	Method	Code			Maths		
		HumanEval	MBPP	Average	MATH	GSM8K	Average
LlaMA-3.2-1B-instruct	Base	**0.3415**	**0.354**	**0.3478**	**0.0899**	0.4587	0.2738
LlaMA-3.2-1B-instruct	RAG	0.3354	0.088	0.2117	0.0823	0.4602	0.2713
LlaMA-3.2-1B-instruct	With Memory	0.2012	0.0	0.1006	**0.0899**	0.4625	0.2762
M3-1B-full	Train With Memory	0.2500	0.056	0.153	0.0718	0.4670	0.2694
M3-1B-full	Train Without Memory	0.04878	0.1020	0.0754	0.0498	0.4117	0.2308
M3-1B-math	Train With Memory	-	-	-	0.0521	**0.5057**	**0.2789**
M3-1B-code	Train With Memory	0.2134	0.2780	0.2457	-	-	-
LlaMA-3.2-3B-instruct	Base	0.5061	0.398	0.4521	0.3421	0.7400	0.5411
LlaMA-3.2-3B-instruct	RAG	-	0.150	-	-	-	-
LlaMA-3.1-8B-instruct	Base	0.5549	0.518	0.5365	0.3721	0.7302	0.5512
LlaMA-3.1-8B-instruct	RAG	-	0.162	-	-	-	-

Table 1. Experiment results. The most effective approach in 1B size is highlighted in bold, while the second-best is underlined.

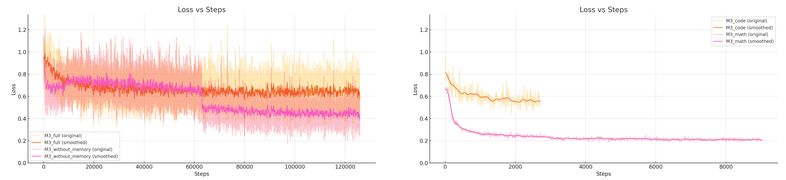


Figure 4. Loss versus Steps curves for LLaMA-3.2-1B-Instruct training. The left panel shows full training on the M3 training dataset, comparing models trained with memory and without memory. The right panel presents training curves for models trained separately on the code and math datasets.

Limitations & Future Work

Insufficient Knowledge Base:
Hongkang Yang et al. (2024) used over 500TB of data including factual, coding, and math. Our setup excluded factual memory due to storage limits, possibly weakening performance.
Future Work: Incorporate factual memory when storage permits.

Simplistic Supervised Fine-Tuning:
We only applied SFT to a pre-trained model, while the reference work included full-stack training (pretraining + SFT + post-training).
Future Work: Explore LoRA fine-tuning, tune memory heads, and select long-context attention heads.

Small Model Capacity:
We used a 1B-parameter model, which may lack capacity to utilize explicit memory effectively.
Future Work: Apply explicit memory techniques to larger models.

Evaluation Inconsistencies:
Used `bigcode-evaluation-harness` and `lm-evaluation-harness`, but MBPP and MATH had to be manually configured. Slight mismatches may have affected accuracy.
Future Work: Carefully verify and align evaluation code with official configurations.

References

[1] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[2] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[3] Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory3: Language modeling with explicit memory. *arXiv preprint arXiv:2407.01178*, 2024.