Memory-Augmented Transformers via Learnable Cross-Attention Memory Banks

Tasmay Pankaj Tibrewal*
tasmay.tibrewal@fractal.ai
Fractal AI Research

Pritish Saha*
pritish.saha@kgpian.iitkgp.ac.in
IIT Kharagpur

Ankit Meda*
ankitm18@kgpian.iitkgp.ac.in
IIT Kharagpur

January 2026

Abstract

Transformers, despite their remarkable success across natural language processing and beyond, lack an explicit architectural mechanism for persistent memory storage and retrieval. Current approaches to memory augmentation predominantly rely on engineering solutions such as KV-caching or retrieval-augmented generation, which operate at inference time rather than at the architectural level. We propose a novel architectural modification: the integration of learnable memory banks accessed via cross-attention mechanisms within transformer blocks. This memory bank consists of latent tokens, randomly initialized and learned during training, which the model can reference as contextual information. We present two complementary research directions: (1) training memory-augmented transformers from scratch, and (2) utilizing memory layers as parameter-efficient adapters for existing pretrained models. We further introduce chapter-based routing inspired by Mixture-of-Experts (MoE) architectures to enable efficient scaling of memory bank size. Additionally, we outline a future direction for dynamic memory updates during inference to enable persistent context retention across conversations. This proposal targets submission to the ICLR 2026 Workshop on New Frontiers in Associative Memories.

1 Introduction

Transformer architectures [Vaswani et al., 2017] have become the dominant paradigm for sequence modeling, achieving state-of-the-art results across natural language processing, computer vision, and multimodal applications. However, a fundamental limitation persists: transformers do not possess an explicit, architecturally integrated memory layer. While the self-attention mechanism allows tokens to attend to all other tokens within the context window, there is no dedicated component for storing and retrieving knowledge that persists beyond the immediate input sequence. Notably, recent theoretical work has established connections between attention mechanisms and associative memory systems, showing that the softmax attention operation corresponds to the update rule of Modern Hopfield Networks [Ramsauer et al., 2020]. This connection suggests that explicit memory mechanisms may be a natural extension of the transformer architecture.

Current solutions to this limitation fall into two broad categories. The first involves engineering approaches such as KV-caching, where key-value pairs from previous tokens are stored and

*Equal contribution.

reused during inference to avoid redundant computation. The second encompasses retrieval-augmented generation (RAG) systems [Lewis et al., 2020], where external documents are retrieved and concatenated to the input context. While these methods have proven effective in practice, they represent applied solutions rather than fundamental architectural innovations. The memory in these systems exists outside the model's learned representations.

Several prior works have explored memory augmentation at the architectural level. Memformer [Wu et al., 2020] introduced cross-attention between layer activations and external memory banks. The Recurrent Memory Transformer (RMT) [Bulatov et al., 2022] augments transformers with global memory tokens passed between segments to enable recurrence. The Memorizing Transformer [Wu et al., 2022] uses approximate kNN lookup into a non-differentiable memory of past key-value pairs. Sandler et al. [Sandler et al., 2022] demonstrated learnable memory tokens for fine-tuning image transformers. Product Key Memory layers [Lample et al., 2019] showed how to efficiently access large memory banks using product key addressing. More recently, comprehensive surveys [Omidi et al., 2025] have systematically categorized these approaches according to functional objectives, memory representations, and integration mechanisms.

In this proposal, we present an architectural modification that introduces a learnable memory bank integrated via cross-attention. Unlike inference-time memory solutions, our memory bank is learned during training, allowing the model to develop compressed, efficient representations of knowledge. We propose two research tracks: (1) training memory-augmented models from scratch, and (2) using memory layers as parameter-efficient adapters. We further introduce MoE-inspired chapter-based routing to scale memory efficiently, and outline future work on dynamic memory updates during inference.

2 Proposed Architecture

2.1 Core Memory Bank Design

We propose augmenting the standard transformer block with a cross-attention layer that enables prompt tokens to attend to a set of learnable memory tokens. Let $\mathbf{M} \in \mathbb{R}^{N_m \times d}$ denote the memory bank, where N_m is the number of memory tokens and d is the embedding dimension. This memory bank is randomly initialized and updated through gradient descent during training.

Given a sequence of prompt tokens with hidden representations $\mathbf{H} \in \mathbb{R}^{L \times d}$ (where L is the sequence length), the cross-attention operation proceeds as follows:

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_Q \tag{1}$$

$$\mathbf{K} = \mathbf{M}\mathbf{W}_K \tag{2}$$

$$\mathbf{V} = \mathbf{M}\mathbf{W}_V \tag{3}$$

$$\text{MemAttn}(\mathbf{H}, \mathbf{M}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{4}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learnable projection matrices and d_k is the head dimension.

2.2 Transformer Block Variants

We consider two variants for integrating the memory cross-attention layer within a transformer block:

Variant A (Post-Self-Attention):

1. Self-attention layer

2. Memory cross-attention layer
3. MLP layer

Variant B (Post-MLP):

1. Self-attention layer
2. MLP layer
3. Memory cross-attention layer
4. Additional MLP layer

Variant B provides an additional non-linear transformation after memory retrieval, potentially enabling more complex integration of retrieved information.

2.3 Memory Layer Placement Strategies

Rather than placing memory cross-attention in every transformer block, selective placement may prove more efficient:

- **First-k layers:** Memory reference in the first k layers, where syntactic and lower-level information is typically processed.
- **Last-k layers:** Memory reference in the final k layers, where higher-level semantic information is integrated.
- **Every-nth layer:** Periodic memory reference (e.g., every 3rd layer).
- **All layers:** Full memory integration throughout the network.

The optimal placement strategy likely depends on the type of information stored in the memory bank and the downstream task requirements.

2.4 Shared vs. Per-Layer Memory Banks

Two memory bank configurations are possible:

Per-Layer Memory Banks: Each memory-enabled layer maintains its own memory bank $\mathbf{M}^{(l)} \in \mathbb{R}^{N_m^{(l)} \times d}$. This allows different layers to store information at different levels of abstraction, analogous to how different transformer layers capture different linguistic phenomena.

Shared Memory Bank: A single memory bank $\mathbf{M} \in \mathbb{R}^{N_m \times d}$ is shared across all memory-enabled layers. This configuration provides each layer access to a larger pool of information but raises a concern: different layers operate in slightly different representational vector-spaces (linear subspaces).

We address this vector-space mismatch concern as follows. The projection matrices $\mathbf{W}_K^{(l)}$ and $\mathbf{W}_V^{(l)}$ at each layer l can learn not only the key/value transformations but also the vector-space transformation between the shared memory space and the layer-specific representation space. Since these projection matrices have dimension $d \times d_k$, and a linear transformation of this size is sufficient to map between any two d-dimensional vector-spaces, the layer-specific projections can absorb the vector-space alignment task without requiring additional parameters.

Formally, we can conceptualize this as:

$$\mathbf{W}_K^{(l)} = \mathbf{W}_K^{\text{core}} \cdot \mathbf{T}_K^{(l)} \tag{5}$$

$$\mathbf{W}_V^{(l)} = \mathbf{W}_V^{\text{core}} \cdot \mathbf{T}_V^{(l)} \tag{6}$$

where $\mathbf{W}_K^{\text{core}}$ performs the standard key projection and $\mathbf{T}_K^{(l)}$ handles vector-space transformation. However, since both are learnable and their product is also a $d \times d_k$ matrix, a single learned $\mathbf{W}_K^{(l)}$ can capture both functionalities.

The shared memory configuration has the advantage of providing each layer access to a larger information pool. A potential downside is that different layers may require different types of information (e.g., syntactic information in early layers, factual knowledge in middle layers), and shared memory may force layers to attend to irrelevant tokens. We propose to investigate this trade-off empirically.

3 Scaling Memory via Chapter-Based Routing

3.1 Motivation

A key challenge in memory-augmented transformers is scaling the memory bank size. With N_m memory tokens and sequence length L, the cross-attention computation scales as $O(L \cdot N_m)$. For large memory banks (e.g., $N_m = 100,000$ tokens), this becomes computationally prohibitive.

We address this challenge by introducing chapter-based routing, inspired by Mixture-of-Experts (MoE) architectures [Shazeer et al., 2017, Fedus et al., 2022]. Rather than attending to all memory tokens, we partition the memory bank into "chapters" and use a router to select the most relevant chapters for each input.

3.2 Chapter Organization

Let the memory bank be partitioned into C chapters, each containing $N_c = N_m/C$ tokens:

$$\mathbf{M} = [\mathbf{M}_1; \mathbf{M}_2; \ldots; \mathbf{M}_C] \tag{7}$$

where $\mathbf{M}_c \in \mathbb{R}^{N_c \times d}$ denotes the c-th chapter.

3.3 Router Architecture

The router computes chapter importance scores based on the input sequence. Let $\bar{\mathbf{h}} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{h}_i$ be the mean-pooled representation of the input sequence. The router computes:

$$\mathbf{s} = \text{softmax}(\mathbf{W}_r \bar{\mathbf{h}} + \mathbf{b}_r) \tag{8}$$

where $\mathbf{W}_r \in \mathbb{R}^{C \times d}$ and $\mathbf{b}_r \in \mathbb{R}^C$ are learnable parameters, and $\mathbf{s} \in \mathbb{R}^C$ contains the importance scores for each chapter.

We select the top-k chapters with highest scores and perform cross-attention only over the tokens in these selected chapters. If $k = 20$ chapters are selected from $C = 100$ chapters (each containing $N_c = 1,000$ tokens), we attend to only $20,000$ tokens instead of $100,000$, achieving a $5\times$ reduction in attention computation.

3.4 Token-Level Routing Challenges

Ideally, routing would be performed at the token level, allowing different query tokens to attend to different memory chapters. This mirrors the token-level expert assignment in standard MoE. However, token-level routing presents significant memory challenges.

Consider the tensor dimensions for token-level memory cross-attention:

$$\mathbf{Q} \in \mathbb{R}^{B \times H \times L \times D} \tag{9}$$

$$\mathbf{K} \in \mathbb{R}^{B \times L \times H \times N_{\text{routed}} \times D} \tag{10}$$

where B is batch size, H is number of heads, L is sequence length, D is head dimension ($D = d_k$), and N_{routed} is the number of routed memory tokens per query.

For practical values ($B = 250$, $L = 10,000$, $H = 32$, $D = 128$, $N_{\text{routed}} = 16,000$), the key tensor would require approximately 300 TB of memory, which is clearly infeasible.

We identify a potential solution through custom CUDA kernels. Rather than materializing the full key tensor, a custom matrix multiplication kernel could:

1. Store only the unique chapters in memory: $\mathbf{K}_{\text{unique}} \in \mathbb{R}^{C_{\text{unique}} \times H \times N_c \times D}$
2. Maintain a routing index that maps each query token to its assigned chapters
3. Perform the attention computation by referencing the same chapter data for multiple queries

This would reduce memory from $O(B \cdot L \cdot N_{\text{routed}})$ to $O(C_{\text{unique}} \cdot N_c)$, a reduction of approximately $40,000\times$ for typical hyperparameters. However, implementing this custom kernel requires significant CUDA expertise and is beyond the scope of our initial experiments. We leave token-level routing as future work and proceed with sequence-level routing for our workshop submission.

4 Memory Adapters for Efficient Fine-Tuning

4.1 Motivation

Given the timeline constraints for the ICLR workshop (deadline: February 14, 2026), pretraining a memory-augmented transformer from scratch and conducting comprehensive experiments may not be feasible within approximately three weeks. We therefore propose an alternative approach: using memory layers as parameter-efficient adapters for existing pretrained models.

This approach offers several advantages:

- Leverages the knowledge already encoded in pretrained weights
- Requires training only the memory bank and associated projection matrices
- Enables direct comparison with established PEFT methods (LoRA [Hu et al., 2021], adapters [Houlsby et al., 2019])
- Reduces computational requirements for experiments

4.2 Memory Adapter Architecture

For a pretrained transformer, we insert memory cross-attention layers at selected positions while keeping the original weights frozen. The trainable parameters consist of:

- Memory bank $\mathbf{M} \in \mathbb{R}^{N_m \times d}$
- Projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ per memory layer
- (Optional) Router parameters for chapter-based routing

4.3 Memory Compression Techniques

To reduce the parameter count and memory footprint of memory adapters, we propose several compression strategies:

Quantized Memory Banks: Store memory tokens in lower precision (e.g., 4-bit or 8-bit) while maintaining full precision for computations.

Low-Rank Memory Factorization: Inspired by LoRA [Hu et al., 2021], we can decompose the memory bank into lower-rank factors:

$$\mathbf{M} = \mathbf{A}\mathbf{B}^\top \tag{11}$$

where $\mathbf{A} \in \mathbb{R}^{N_m \times r}$ and $\mathbf{B} \in \mathbb{R}^{d \times r}$ with $r \ll \min(N_m, d)$.

This reduces the parameter count from $N_m \cdot d$ to $(N_m + d) \cdot r$. However, this comes at the cost of reduced expressiveness, as each memory token can now store less information.

Low-Rank Projections: The projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ can also be factorized:

$$\mathbf{W}_Q = \mathbf{W}_Q^{\text{down}} \mathbf{W}_Q^{\text{up}} \tag{12}$$

where $\mathbf{W}_Q^{\text{down}} \in \mathbb{R}^{d \times r}$ and $\mathbf{W}_Q^{\text{up}} \in \mathbb{R}^{r \times d_k}$.

4.4 Proposed Experiments

We propose the following experimental comparisons using a base model such as Qwen-2.5-1.5B [Qwen Team, 2024] or Qwen-2.5-Math-1.5B:

1. **Full Fine-Tuning:** Update all model parameters (baseline)
2. **LoRA:** Low-rank adaptation of attention weights
3. **Memory Adapters (Ours):** Memory bank with cross-attention
4. **Memory Adapters + LoRA:** Combination of both approaches

Datasets: We consider fine-tuning on:

- DeepSeek-R1 reasoning traces from Open-R1 math dataset (to evaluate whether memory can capture reasoning patterns)
- Domain-specific datasets (medical, legal) where factual knowledge retention is critical

Evaluation: Performance will be measured on held-out test sets, with particular attention to:

- Task accuracy
- Parameter efficiency (trainable parameters vs. performance)
- Inference speed overhead from memory attention

4.5 Preventing Knowledge Loss During Fine-Tuning

A known issue with fine-tuning is catastrophic forgetting, where the model loses information acquired during pretraining. Memory adapters may help mitigate this issue, as the pretrained weights remain frozen. Additionally, we propose:

Memory Freezing: After pretraining (for from-scratch models) or an initial fine-tuning phase, freeze the memory bank and associated $\mathbf{W}_K, \mathbf{W}_V$ matrices while continuing to train other components.

Differential Learning Rates: Apply lower learning rates to memory bank parameters compared to other trainable parameters, reducing the rate at which memorized information is overwritten.

5 From-Scratch Training

While memory adapters are our primary focus for the workshop submission, we also outline plans for training memory-augmented transformers from scratch, if possible in time.

5.1 Proposed Setup

- **Model Size:** Approximately 100M parameters
- **Pretraining Data:** 5-10 billion tokens
- **Instruction Fine-Tuning:** 10-100M tokens
- **Memory Configuration:** 10,000-1M memory tokens (shared or per-layer variants)

5.2 Expected Outcomes

We hypothesize that the learned memory bank will contain more compressed and efficient representations compared to:

- Textual memory (as in retrieval-augmented systems)
- Non-differentiable memory stores (as in Memorizing Transformers [Wu et al., 2022])
- Inference-time memory (as in TransformerFAM [Hwang et al., 2024])

The model should be compared against a baseline transformer of equivalent parameter count (excluding memory bank) on standard language modeling benchmarks.

6 Dynamic Memory Update During Inference (Future Work)

6.1 Motivation

The memory mechanisms described thus far are static: the memory bank is fixed after training and does not update during inference. This limits the model's ability to incorporate new information encountered at inference time.

We propose extending the architecture with a dynamic "context bank" that is updated during inference, enabling:

- Long-context retention beyond the attention window
- Cross-conversation memory (remembering information across sessions)
- Personal agents that accumulate knowledge about users over time

6.2 Context Bank Architecture

We maintain a separate context bank $\mathbf{C} \in \mathbb{R}^{N_{ct} \times d}$ that stores compressed representations of inference-time inputs. The context bank is distinct from the memory bank to prevent training-time knowledge from being overwritten.

Compression: Input sequences are compressed before storage using a VAE-style encoder. Given input tokens with embeddings from the first layer $\mathbf{E}^{(1)} \in \mathbb{R}^{L \times d}$ and last layer $\mathbf{E}^{(L_{\max})} \in \mathbb{R}^{L \times d}$, we train an encoder:

$$\mathbf{z} = \text{Encoder}([\mathbf{E}^{(1)}; \mathbf{E}^{(L_{\max})}]) \tag{13}$$

where $\mathbf{z} \in \mathbb{R}^{L_c \times d}$ with $L_c \ll L$ (e.g., compressing 10,000 tokens to 100 tokens).

The encoder is trained to reconstruct the concatenated embeddings, ensuring that both raw semantic information (from early layers) and processed information (from late layers) are preserved.

Storage: Compressed representations $\mathbf{z}$ are appended to the context bank until a maximum capacity $N_{ct}^{\max}$ is reached.

Retrieval: Since attending to a large context bank (e.g., 1M tokens) is computationally prohibitive, we use clustering-based retrieval (clusters are analogous to chapters here, but since we are adding tokens in test-time, we cannot have a trained router for retrieval and thus we cluster similar tokens together for retrieval), similar to approaches in efficient attention literature [RetrievalAttention, 2024]:

1. Cluster context bank tokens using k-means or hierarchical clustering
2. For a new query, compute dot products with cluster centroids
3. Select top-k clusters and attend only to tokens within those clusters

This approach requires no training at inference time, as cluster selection is based on embedding similarity rather than learned routing.

6.3 Memory Consolidation

When the context bank reaches capacity and new information must be stored, we propose a consolidation mechanism inspired by how humans forget information:

Vector Merging: Identify pairs of vectors with highest cosine similarity and merge them via addition or averaging:

$$\mathbf{c}_{\text{merged}} = \frac{\mathbf{c}_i + \mathbf{c}_j}{2} \tag{14}$$

This is inspired by RAG systems where chunk embeddings are averaged without catastrophic information loss for semantic matching.

Importance-Weighted Merging: Incorporate additional factors when selecting which vectors to merge:

- **Reference frequency:** How often a vector has been attended to
- **Attention importance:** Average softmax weight when attended
- **Recency:** Newer information is retained longer (benefit of doubt for importance)

A scoring function could weight these factors:

$$\text{importance}(\mathbf{c}) = \alpha \cdot \text{freq}(\mathbf{c}) + \beta \cdot \text{avg\_attn}(\mathbf{c}) + \gamma \cdot \text{recency}(\mathbf{c}) \tag{15}$$

Vectors with low importance scores are prioritized for merging.

Note: All compression techniques (low-rank, quantization) described for the memory bank can also be applied to the context bank.

7 Relation to Prior Work

Our proposal builds upon and differs from several lines of prior research. We provide a comprehensive comparison to position our contributions.

Associative Memory and Hopfield Networks: Classical Hopfield networks [Hopfield, 1982] provided the foundational framework for associative memory in neural networks. Ramsauer et al. [Ramsauer et al., 2020] established that the attention mechanism in transformers is mathematically equivalent to the update rule of Modern Hopfield Networks, providing a theoretical grounding for viewing transformers as associative memory systems. Our work extends this view by adding an explicit, learnable memory bank that expands the set of patterns available for retrieval.

Cross-Attention Memory with Dynamic Updates: Several works have explored cross-attention to memory banks. Memformer [Wu et al., 2020] introduced cross-attention between layer activations and external memory banks—using the same Q/K/V formulation as our approach (input provides Q, memory provides K,V)—but with gated updates that modify memory during the forward pass. Most recently, LM2 (Large Memory Models) [Kang et al., 2025] proposed an auxiliary memory module with cross-attention and LSTM-style gating (input, output, forget gates) for dynamic memory updates. Our key distinction is that our memory bank is **static after training**: it stores persistent learned knowledge rather than working memory that evolves during inference. This design choice targets knowledge retention rather than context tracking.

Conditional Memory and Hash-Based Lookup: DeepSeek's Engram [DeepSeek, 2026] introduces conditional memory as a complementary sparsity axis to MoE, using hash-based O(1) lookup for static token patterns (common phrases, entities). While Engram and our approach share the goal of separating memory from computation, the mechanisms differ fundamentally: Engram uses deterministic hash addressing for specific n-grams, while our attention-based approach enables soft retrieval with learned content-based addressing. Our chapter routing provides semantic organization, whereas Engram's hashing is pattern-based.

Product Key Memory and Memory Layers: Lample et al. [Lample et al., 2019] introduced Product Key Memory layers that enable efficient access to very large memory banks using product key addressing. More recently, Memory Layers at Scale [Berges et al., 2025] demonstrated that trainable key-value memory layers can be scaled effectively and show strong factual gains. These techniques use sparse top-k retrieval rather than attention and could be integrated with our chapter-based routing for further scalability.

Memory Tokens and Global Memory: Memory Transformer [Burtsev et al., 2020] and GMAT [Gupta and Berant, 2020] add special memory tokens that participate in self-attention (not cross-attention). ETC [Ainslie et al., 2020] uses global tokens for encoding long and structured inputs. These systems demonstrate that memory-like token mechanisms help with long-context reasoning, but memory tokens are typically dynamic per sequence and use self-attention rather than cross-attention to a separate persistent store.

Recurrent Memory Approaches: The Recurrent Memory Transformer (RMT) [Bulatov et al., 2022, 2023] augments transformers with memory tokens passed between segments via self-attention. The Associative Recurrent Memory Transformer (ARMT) [Rodkin et al., 2024] extends RMT with linear attention-style associative memory, achieving impressive results on 50M+ token sequences. Block-Recurrent Transformers [Hutchins et al., 2022] use block-level recurrence. These approaches focus on segment-level recurrence for long sequences; our memory bank stores learned knowledge accessed via cross-attention without recurrence.

Non-Parametric Memory: Memorizing Transformer [Wu et al., 2022] stores actual KV pairs from past forward passes and retrieves via kNN lookup. kNN-LM [Khandelwal et al., 2019] retrieves from a datastore of context-target pairs. These non-parametric approaches cache actual activations rather than learning compressed representations. Our memory is learned end-to-end, potentially storing more efficient representations.

Entity and Knowledge-Specific Memory: Entities as Experts [Fevry et al., 2020] stores entity embeddings in dedicated memory slots with sparse access. Mention Memory [de Armas et al., 2021] stores 150M entity mention embeddings accessed via attention. QAMAT [Chen et al., 2023] uses dedicated memory for question-answering. These demonstrate the value of explicit memory structures but are task/entity-specific, whereas our approach targets general learned knowledge.

Long-Context Transformers: Transformer-XL [Dai et al., 2019] introduced segment-level recurrence. Compressive Transformers [Rae et al., 2020] compress past activations. Infini-attention [Munkhdalai et al., 2024] proposed compressive memory within attention. Augmenting Language Models with Long-Term Memory [Wang et al., 2023] adds persistent memory for extended conversations. These focus on extending effective context length, while we focus on persistent learned knowledge.

Test-Time Memory and Adaptation: Titans [Behrouz et al., 2025] learns what to memorize at test time with dynamic updates. Larimar [Liu et al., 2024] provides episodic memory control for LLMs. NAMMs [Cetin et al., 2025] use evolutionary optimization for memory decisions. These approaches update memory during inference; our memory is fixed after training, providing stable knowledge without runtime overhead.

Cross-Attention Architectures: Perceiver [Jaegle et al., 2021a] and Perceiver IO [Jaegle et al., 2021b] use cross-attention to a latent array as a computational bottleneck. Flamingo [Alayrac et al., 2022] uses cross-attention to inject visual information into frozen language models. Our memory cross-attention is conceptually similar but targets persistent knowledge storage rather than multimodal fusion or computational bottlenecks.

Learnable Memory Tokens: Sandler et al. [Sandler et al., 2022] demonstrated learnable memory tokens for vision transformers, concatenating them to input for self-attention. Heterogeneous Memory Augmented Neural Networks [Park et al., 2023] use learnable memory for OOD generalization. Our work extends the learnable memory concept to language models with cross-attention (not self-attention), chapter-based routing, and memory adapter formulations.

Explicit Memory Systems: Memory3 [Yang et al., 2024] models explicit memory as retrievable model parameters. Token Turing Machines [Hawthorne et al., 2024] use memory tapes with read/write operations. These represent alternative memory paradigms; our approach uses standard attention mechanics for compatibility with existing transformer infrastructure.

Context Compression: Activation Beacon [Activation Beacon, 2024] and IC-Former [Wang et al., 2024] compress long contexts into shorter representations. Our context bank proposal incorporates similar compression ideas but maintains a persistent store across inference sessions with importance-weighted consolidation.

Retrieval-Augmented Methods: RAG [Lewis et al., 2020] and RETRO [**?**] augment models with retrieved text from external corpora. These store memory as text requiring re-encoding; our approach stores learned latent representations that may be more compact and task-appropriate.

Routing and Sparse Attention: Routing Transformer [Roy et al., 2021] uses clustering for content-based sparse attention. MoE architectures [Shazeer et al., 2017, Fedus et al., 2022] use token-wise routing to experts. Our chapter routing applies MoE-style routing to memory chapters—a novel application that enables scaling memory banks while maintaining attention-based access.

Parameter-Efficient Fine-Tuning: LoRA [Hu et al., 2021] and QLoRA [Dettmers et al., 2023] enable efficient adaptation through low-rank updates. Prefix-tuning [Li and Liang, 2021] and P-Tuning v2 [Liu et al., 2022] add learned continuous prompts attended via self-attention, which is conceptually close to learnable tokens. Our memory adapters differ in using cross-attention to a separate memory store with explicit capacity scaling via chapters.

7.1 Summary of Distinctions

To our knowledge, no prior work combines all of: (1) learnable latent memory bank, (2) cross-attention access (input Q, memory K,V), (3) static/persistent memory after training, (4) MoE-style chapter routing for scaling, and (5) memory adapter formulation for PEFT. The closest works are Memformer and LM2 (cross-attention but dynamic updates), Sandler et al. (learnable but self-attention), PKM/Engram (persistent but different access mechanisms), and prefix-tuning (learnable tokens but task-specific prompting).

8 Novelty Summary

The novel contributions of this proposal include:

1. **Learnable cross-attention memory banks** for language transformers, with analysis of shared vs. per-layer configurations and the vector-space alignment property of projection matrices.

2. **Chapter-based routing** for scaling memory bank size, including analysis of token-level routing challenges and a proposed custom CUDA kernel solution.

3. **Memory adapters** as a parameter-efficient fine-tuning method, with low-rank and quantization variants.

4. **Comprehensive experimental framework** comparing memory adapters against LoRA and full fine-tuning.

5. **Dynamic context bank** concept with VAE compression, clustering-based retrieval, and importance-weighted memory consolidation (future work).

9 Conclusion

We have presented a comprehensive proposal for memory-augmented transformers via learnable cross-attention memory banks. The architecture provides an explicit mechanism for knowledge storage and retrieval at the architectural level, addressing a fundamental limitation of standard transformers. Chapter-based routing enables efficient scaling, while memory adapters provide a practical path to experimentation within workshop timelines. The dynamic context bank extends these ideas to enable persistent memory across inference sessions. We believe this direction holds significant promise for improving the knowledge retention and contextual reasoning capabilities of transformer-based language models.

References

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554-2558.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G.K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2020). Hopfield networks is all you need. In *International Conference on Learning Representations*. `https://arxiv.org/abs/2008.02217`

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. `https://arxiv.org/abs/1706.03762`

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33. `https://arxiv.org/abs/2005.11401`

Wu, Q., Lan, Z., Gu, J., and Yu, Z. (2020). Memformer: The memory-augmented transformer. *arXiv preprint arXiv:2010.06891*. `https://arxiv.org/abs/2010.06891`

Bulatov, A., Kuratov, Y., and Burtsev, M.S. (2022). Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35. `https://arxiv.org/abs/2207.06881`

Bulatov, A., Kuratov, Y., and Burtsev, M.S. (2023). Scaling transformer to 1M tokens and beyond with RMT. *arXiv preprint arXiv:2304.11062*. `https://arxiv.org/abs/2304.11062`

Wu, Y., Rabe, M.N., Hutchins, D., and Szegedy, C. (2022). Memorizing transformers. In *International Conference on Learning Representations*. `https://arxiv.org/abs/2203.08913`

Sandler, M., Zhmoginov, A., Vladymyrov, M., and Jackson, A. (2022). Fine-tuning image transformers using learnable memory. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. `https://arxiv.org/abs/2203.15243`

Lample, G., Sablayrolles, A., Ranzato, M., Denoyer, L., and Jégou, H. (2019). Large memory layers with product keys. In *Advances in Neural Information Processing Systems*, volume 32. `https://arxiv.org/abs/1907.05242`

Omidi, P., Huang, X., Laborieux, A., Nikpour, B., Shi, T., and Eshaghi, A. (2025). Memory-augmented transformers: A systematic review from neuroscience principles to technical solutions. *arXiv preprint arXiv:2508.10824*. `https://arxiv.org/abs/2508.10824`

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*. https://arxiv.org/abs/1701.06538

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1-39. https://arxiv.org/abs/2101.03961

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. https://arxiv.org/abs/2106.09685

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36. https://arxiv.org/abs/2305.14314

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. https://arxiv.org/abs/1902.00751

Qwen Team (2024). Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*. https://arxiv.org/abs/2412.15115

Zhang, P., Qian, H., Ye, Q., and Dou, Z. (2024). Long context compression with activation beacon. *arXiv preprint arXiv:2401.03462*. https://arxiv.org/abs/2401.03462

Wang, X., Chen, Z., Xu, T., Xie, Z., He, Y., and Chen, E. (2024). In-context former: Lightning-fast compressing context for large language model. In *Findings of EMNLP 2024*. https://arxiv.org/abs/2406.13618

Cetin, E., Palmieri, A., Poli, M., Coda, D., Fang, B., and Tang, L. (2025). An evolved universal transformer memory. Sakana AI Blog. https://sakana.ai/namm/ (Accessed: January 2026).

Hwang, D., Wang, W., Huo, Z., Sim, K.C., and Mengibar, P. (2024). TransformerFAM: Feedback attention is working memory. *arXiv preprint arXiv:2404.09173*. https://arxiv.org/abs/2404.09173

Liu, D., et al. (2024). RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. *arXiv preprint arXiv:2409.10516*. https://arxiv.org/abs/2409.10516

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. (2022). Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35. https://arxiv.org/abs/2202.09368

Ge, T., et al. (2023). In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*. https://arxiv.org/abs/2307.06945

He, Z., Cao, Y., Qin, Z., Prakriya, N., Sun, Y., and Cong, J. (2024). HMT: Hierarchical memory transformer for efficient long context language processing. *arXiv preprint arXiv:2405.06067*. https://arxiv.org/abs/2405.06067

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. https://arxiv.org/abs/1901.02860

Rae, J.W., Potapenko, A., Jayakumar, S.M., Hillier, C., and Lillicrap, T.P. (2020). Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*. https://arxiv.org/abs/1911.05507

Munkhdalai, T., Faruqui, M., and Gopal, S. (2024). Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*. https://arxiv.org/abs/2404.07143

Behrouz, A., Zhong, P., and Mirrokni, V. (2025). Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*. https://arxiv.org/abs/2501.00663

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2019). Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*. https://arxiv.org/abs/1911.00172

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paez, P., Sheridan, G., Landon, J., and Sifre, L. (2022). Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*. https://arxiv.org/abs/2112.04426

Berges, V.-P., Oguz, B., Haziza, D., Yih, W.-t., Zettlemoyer, L., and Ghosh, G. (2025). Memory layers at scale. In *International Conference on Machine Learning*. https://arxiv.org/abs/2412.09764

Burtsev, M.S., Kuratov, Y., Peganov, A., and Sapunov, G.V. (2020). Memory transformer. *arXiv preprint arXiv:2006.11527*. https://arxiv.org/abs/2006.11527

Gupta, A. and Berant, J. (2020). GMAT: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274*. https://arxiv.org/abs/2006.03274

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. https://arxiv.org/abs/2004.08483

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. (2021). Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53-68. https://arxiv.org/abs/2003.05997

Li, X.L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. https://arxiv.org/abs/2101.00190

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., and Tang, J. (2022). P-Tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. https://arxiv.org/abs/2110.07602

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., and Carreira, J. (2021). Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*. https://arxiv.org/abs/2103.03206

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ber, D., Lakshminarayanan, B., Zisserman, A., Vinyals, O., and Carreira, J. (2021). Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*. https://arxiv.org/abs/2107.14795

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Bińkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35. https://arxiv.org/abs/2204.14198

Kang, J., Wu, W., Christianos, F., Chan, A.J., Greenlee, F., Thomas, G., Purtorab, M., and Toulis, A. (2025). LM2: Large memory models. *arXiv preprint arXiv:2502.06049*. https://arxiv.org/abs/2502.06049

DeepSeek-AI (2026). Engram: Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*. https://arxiv.org/abs/2601.07372

Rodkin, I., Kuratov, Y., Bulatov, A., and Burtsev, M. (2024). Associative recurrent memory transformer. In *ICML 2024 Workshop on Next Generation of Sequence Modeling Architectures*. https://arxiv.org/abs/2407.04841

Hutchins, D., Schlag, I., Wu, Y., Dyer, E., and Neyshabur, B. (2022). Block-recurrent transformers. In *Advances in Neural Information Processing Systems*, volume 35. https://arxiv.org/abs/2203.07852

Fevry, T., Soares, L.B., FitzGerald, N., Choi, E., and Kwiatkowski, T. (2020). Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. https://arxiv.org/abs/2004.07202

de Armas, R.M., Martins, B., and Calado, P. (2021). Mention memory: Incorporating textual knowledge into transformers through entity mention attention. *arXiv preprint arXiv:2110.06176*. https://arxiv.org/abs/2110.06176

Chen, W., Pat, A., and Roth, D. (2023). Augmenting pre-trained language models with QA-memory for open-domain question answering. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. https://aclanthology.org/2023.eacl-main.117/

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., and Wei, F. (2023). Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36. https://arxiv.org/abs/2306.07174

Liu, P., Zhang, Y., and Weld, D.S. (2024). Larimar: Large language models with episodic memory control. *arXiv preprint arXiv:2403.11901*. https://arxiv.org/abs/2403.11901

Park, S., Kim, J., and Lee, J. (2023). Heterogeneous memory augmented neural networks. *arXiv preprint arXiv:2310.10909*. https://arxiv.org/abs/2310.10909

Yang, H., et al. (2024). Memory3: Language modeling with explicit memory. *arXiv preprint*. https://arxiv.org/abs/2407.01178

Hawthorne, C., et al. (2024). Token turing machines are efficient vision models. *arXiv preprint arXiv:2409.07613*. https://arxiv.org/abs/2409.07613