TOWARDS COMPRESSIVE AND SCALABLE RECURRENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent memory unlocks an unbounded context horizon for language models within a finite memory footprint. We introduce ***Elastic Memory***, a memory architecture that compresses historical context into polynomial states and reconstructs these features for attention. By leveraging HiPPO for parallelized online compression and polynomial sampling for flexible retrieval, our approach operates without additional trainable parameters. This design decouples memory capacity from the backbone model, maintaining a fixed-size recurrent state regardless of sequence length and model size. In from-scratch pretraining, Elastic Memory outperforms strong baselines on PPL and LongPPL across three long-document domains. Notably, it achieves lower perplexity than Memorizing Transformer with a 16× smaller memory footprint, and scales robustly with model size and memory size. During continued pretraining (CPT) on Llama-2 7B, it matches FreqKV's quality while delivering a 1.83× throughput speedup. For supervised fine-tuning (SFT), Elastic Memory achieves strong results on LongBench, including a 5.31-point gain over FreqKV on RepoBench-P. Elastic Memory provide a principled, scalable and parameter-free recurrent memory for LLMs. [1]

1 INTRODUCTION

Recurrent memory offers a powerful mechanism to extend a language model's context horizon beyond local constraints, but its practical viability hinges on the trade-off between representation quality, computational overhead, and storage footprint. Existing architectures navigate this space with significant compromises. External memories retain past representations and retrieve selected entries, inherently tying their access costs and storage requirements to the length of the stored history (Dai et al., 2019; Wu et al., 2022). Associative memories compress history into states bounded by the dimensions of attention heads and feature widths (Schlag et al., 2021; Munkhdalai et al., 2024). Meanwhile, learned summary memories provide configurable capacity, but expanding this capacity introduces additional trainable parameters and computational burdens (Bulatov et al., 2022; Chen et al., 2025). These limitations motivate a central question: *can a compact, fixed-size memory represent unbounded history effectively while scaling its capacity entirely independently of the backbone language model?*

To address this, we introduce **Elastic Memory**, an architecture that compresses historical context into the coefficients of an online polynomial approximation. Leveraging the HiPPO framework (Gu et al., 2020) for projection dynamics, we design a parallel block update mechanism paired with a novel *polynomial-sampling* reader for Transformer attention (Figure 1). In this design, each block reads reconstructed historical representations, attends to them alongside local context, and updates the recurrent state. Crucially, the state summarizes history at a configurable polynomial order; increasing memory capacity improves representation resolution without enlarging the backbone model. Because all compression and reconstruction operators are deterministic, Elastic Memory operates completely free of additional trainable parameters across pretraining, continued pretraining (CPT), and supervised fine-tuning (SFT). Furthermore, the polynomial-sampling reader permits post-training adjustments to historical reconstruction, enabling existing checkpoints to dynamically optimize their utilization of stored information.

[1] Code will be released at `anonymous.url.com`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

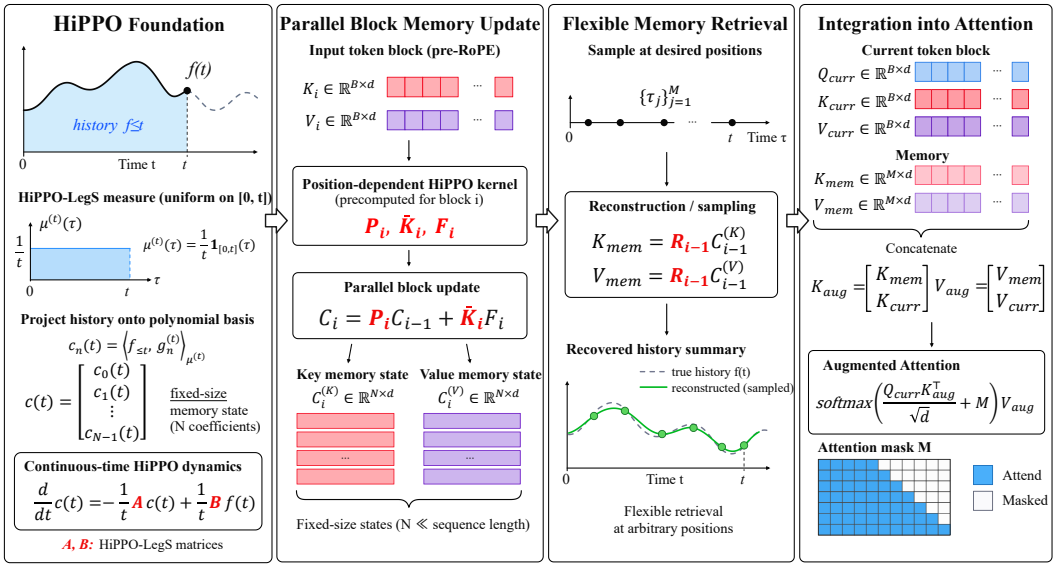


Figure 1: Overview of Elastic Memory. (a) HiPPO represents history as coefficients of an online polynomial approximation. (b) Precomputed operators update the key and value states in parallel within each block. (c) Polynomial sampling reconstructs a compact history at selected positions. (d) Reconstructed keys and values augment causal attention over the current block.

Our experiments validate the efficacy and efficiency of this compact memory paradigm across three distinct training stages. In from-scratch pretraining (PG-19, Proof-Pile, FineWeb-Edu), Elastic Memory outperforms strong baselines on PPL and LongPPL, yielding up to a 53.6% relative LongPPL reduction on FineWeb-Edu. On Proof-Pile, it achieves lower perplexity than Memorizing Transformer with a 16× smaller memory budget and surpasses Melodi at restricted budgets without introducing trainable parameters. This architectural advantage scales robustly from 100M to 400M parameters, achieving 1.52× the training throughput of Melodi at the largest scale. During CPT on Llama-2 7B, Elastic Memory matches the modeling quality of the highly competitive FreqKV baseline (Kai et al., 2026) while delivering a 1.83× throughput speedup and successfully generalizing from an 8K training window to 32K evaluation lengths. Finally, in SFT on Llama-3-8B-Instruct, the architecture achieves strong results on LongBench, highlighted by a 5.31-point gain over FreqKV on RepoBench-P. Collectively, these findings establish Elastic Memory as a principled, highly scalable, and parameter-free paradigm for long-context language modeling.

2 Elastic Memory

Elastic Memory equips Transformer attention with a recurrent representation of historical keys and values. At each block, it reconstructs history from the previous state, uses that history in causal attention, and updates the state with the current block. The HiPPO projection is inherited from Gu et al. (2020); the architecture combines its blockwise application to keys and values with polynomial reconstruction for attention. Figure 1 shows the data flow, and Algorithm 1 summarizes the forward pass.

2.1 Representing History with Polynomial States

We treat each feature channel of the key and value sequences as a signal evolving over token positions. Keys enter the memory before rotary position encoding (RoPE), preserving the underlying feature signal for compression. For a scalar channel f, HiPPO-LegS maintains the coefficients of the

degree-$N-1$ polynomial that best approximates its history under a uniform measure on $[0, t]$:

$$g^{(t)} = \arg\min_{g \in \mathcal{P}_{N-1}} \frac{1}{t} \int_0^t \big(f(x) - g(x)\big)^2 \, dx, \qquad g^{(t)}(x) = \sum_{n=0}^{N-1} c_n(t) g_n^{(t)}(x). \tag{1}$$

Here $g_n^{(t)}(x) = \sqrt{2n+1} P_n(2x/t - 1)$ is the scaled Legendre basis, and $c_n(t)$ is its projection coefficient. The optimization defines the representation; the HiPPO recurrence updates it without rereading the past. Appendix C.1 gives the projection and state-space derivation.

Let blocks have length L and use zero-based indices. Before block i, the processed history has length $t_i = iL$. For each attention head, we store $\mathbf{C}_i^{(K)} \in \mathbb{R}^{N \times d_k}$ and $\mathbf{C}_i^{(V)} \in \mathbb{R}^{N \times d_v}$, with one coefficient vector per feature channel. Both states represent the history up to t_i; the first block begins with an empty history. The polynomial order N sets the amount of information the state can represent. Increasing N gives the approximation more degrees of freedom, so a larger memory can describe the same history more faithfully. This is the capacity axis evaluated in Section 3.1.

2.2 Updating Memory in Parallel Blocks

Discretizing the HiPPO dynamics gives fixed, position-dependent operators $\bar{\mathbf{A}}_k \in \mathbb{R}^{N \times N}$ and $\bar{\mathbf{B}}_k \in \mathbb{R}^N$ for the update

$$c_{k+1} = \bar{\mathbf{A}}_k c_k + \bar{\mathbf{B}}_k f_k. \tag{2}$$

We use zero-order hold (ZOH), with the initialization and discretization specified in Appendix C.2. The operators depend on token position and polynomial order, and introduce no learned weights.

To process an entire block in parallel, we compose its token updates. For block features $\mathbf{F}_i \in \mathbb{R}^{L \times d}$, the resulting update is

$$\mathbf{C}_{i+1} = \mathbf{P}_i \mathbf{C}_i + \bar{\mathbf{K}}_i \mathbf{F}_i. \tag{3}$$

The transition $\mathbf{P}_i \in \mathbb{R}^{N \times N}$ evolves the previous state across the block, while the HiPPO kernel $\bar{\mathbf{K}}_i \in \mathbb{R}^{N \times L}$ writes all new features into that state. The same update applies separately to raw keys and values. Appendix C.3 derives these operators and their equivalence to the discrete recurrence.

For a configured context range, the position-dependent operators can be precomputed and cached. A forward pass then loads the operators for block i and performs the two matrix products in Eq. 3. Blocks retain their recurrent ordering, while all tokens within an update are processed together. This organization makes the polynomial state practical for blockwise Transformer training and avoids a serial update loop over tokens.

2.3 Reconstructing History for Attention

Polynomial sampling. The coefficient state can reconstruct historical features at any selected positions. For $i > 0$, choose M points $x_j \in [0, t_i)$ and evaluate the basis to form a reconstruction matrix:

$$(\mathbf{R}_i)_{jn} = \sqrt{2n+1}\, P_n\left(\frac{2x_j}{t_i} - 1\right), \qquad \mathbf{R}_i \in \mathbb{R}^{M \times N}. \tag{4}$$

The historical keys and values are recovered in parallel:

$$\mathbf{K}_{\text{mem}} = \mathbf{R}_i \mathbf{C}_i^{(K)}, \qquad \mathbf{V}_{\text{mem}} = \mathbf{R}_i \mathbf{C}_i^{(V)}. \tag{5}$$

These operations reconstruct feature vectors for ordinary attention. The sampling coordinates specify where the reader accesses the approximation; the first block simply omits the historical entries.

Attention integration. The reconstructed keys and values are prepended to the current block's keys and values. Every query can attend to the historical entries, while a causal mask restricts access within the current block:

$$\mathbf{O} = \text{softmax}\left(\frac{\mathbf{Q}_{\text{curr}}[\mathbf{K}_{\text{mem}}; \mathbf{K}_{\text{curr}}]^\top}{\sqrt{d_k}} + \mathbf{M}_{\text{causal}}\right)[\mathbf{V}_{\text{mem}}; \mathbf{V}_{\text{curr}}]. \tag{6}$$

In our from-scratch architecture, RoPE is applied to current queries and keys, and the reconstructed memory enters attention directly. The transfer configuration uses compact RoPE in the attention path; Appendix A.2 specifies that configuration. In both cases, the stored key state is formed from pre-RoPE features and is updated only after the current block has read the previous history.

Adapting the reader. Changing the sample points changes the information presented to attention without changing the stored coefficients. We evaluate uniform sampling across history and exponential sampling concentrated near the present. Their matrices can be precomputed from the chosen coordinates; full definitions appear in Appendix C.4. This gives a trained checkpoint a direct way to adjust its use of history without a parameter update, which we test in Section 3.1.

Algorithm 1 Elastic Memory Attention Block (From-Scratch Configuration)

Require: Input block $\mathbf{H}_i$; states $\mathbf{C}_i^{(K)}, \mathbf{C}_i^{(V)}$; zero-based block index i.

$\mathbf{Q}_{\text{raw}}, \mathbf{K}_{\text{raw}}, \mathbf{V}_{\text{curr}} \leftarrow \text{ProjectQKV}(\mathbf{H}_i)$
$\mathbf{Q}_{\text{curr}}, \mathbf{K}_{\text{curr}} \leftarrow \text{RoPE}(\mathbf{Q}_{\text{raw}}, \mathbf{K}_{\text{raw}})$
if $i > 0$ **then**
  $\mathbf{K}_{\text{mem}} \leftarrow \mathbf{R}_i \mathbf{C}_i^{(K)}$; $\mathbf{V}_{\text{mem}} \leftarrow \mathbf{R}_i \mathbf{C}_i^{(V)}$ ▷ Read history
else
  $\mathbf{K}_{\text{mem}}, \mathbf{V}_{\text{mem}} \leftarrow$ empty sequences
end if
$\mathbf{O} \leftarrow \text{CausalAttention}(\mathbf{Q}_{\text{curr}}, [\mathbf{K}_{\text{mem}}; \mathbf{K}_{\text{curr}}], [\mathbf{V}_{\text{mem}}; \mathbf{V}_{\text{curr}}])$
$\mathbf{C}_{i+1}^{(K)} \leftarrow \mathbf{P}_i \mathbf{C}_i^{(K)} + \bar{\mathbf{K}}_i \mathbf{K}_{\text{raw}}$ ▷ Write current keys
$\mathbf{C}_{i+1}^{(V)} \leftarrow \mathbf{P}_i \mathbf{C}_i^{(V)} + \bar{\mathbf{K}}_i \mathbf{V}_{\text{curr}}$ ▷ Write current values
return $\text{OutputProjection}(\mathbf{O}), \mathbf{C}_{i+1}^{(K)}, \mathbf{C}_{i+1}^{(V)}$

2.4 Scaling Memory without Additional Parameters

Elastic Memory separates the storage budget N from the backbone dimensions. For H heads, its key and value states contain $NH(d_k + d_v)$ elements per augmented layer. The reader independently selects M reconstructed entries for attention, and L specifies the current block length. These variables control representation capacity, historical attention width, and local context, respectively. Increasing N therefore enlarges the memory without adding model weights; for any fixed configuration, the recurrent state stays the same size as the processed sequence grows.

Table 1 compares this design with representative recurrent memories. Every Elastic Memory update and reconstruction operator is determined analytically, so the memory mechanism introduces **zero additional trainable parameters** in all evaluated settings. This allows more storage to improve historical representation without expanding the backbone or learning a larger compressor. Precomputed operators and temporary attention buffers are accounted for separately from the recurrent state in Appendix C.5.

Table 1: Recurrent memory designs. Memory size and additional trainable parameters are counted per layer; H is the number of attention heads. Experimental adaptations of these designs are described in Appendix A.1.

Model	Memory Size	Add. Params	Update	Retrieval
MemTrans.	$N_{\text{queue}} \times (d_{\text{key}} + d_{\text{value}}) \times H$	H	**FIFO queue**	**kNN+ cross-attn**
InfiniTrans.	$d_{\text{key}} \times d_{\text{value}} \times H$	H	**Hebb rule+ delta rule**	**linear-attn**
Melodi	$(N_{\text{mem.}} + N_{\text{sum.}}) \times d_{\text{value}} \times H$	$N_{\text{sum.}} \times d_{\text{value}} \times H$	**self-attn+ linear-mix.**	**cross-attn**
ElasticMem.	$N_{\text{HiPPO}} \times (d_{\text{key}} + d_{\text{value}}) \times H$	**0**	**function approx.**	**poly. sampling+ cross-attn**

3 Experiments

We test whether Elastic Memory makes better use of a limited state, turns additional storage into better predictions, and brings these capabilities to existing language models. The three studies use independent training regimes: pretraining from scratch, continued pretraining (CPT), and supervised

fine-tuning (SFT). Elastic Memory introduces no additional trainable parameters in any setting. Full protocols and supporting analyses appear in Appendices A and B.

3.1 Pretraining from Scratch: Quality and Scalability

Setup. We train a unified Llama-style architecture on books from PG-19 (Rae et al., 2020), mathematics from Proof-Pile (Azerbayev et al., 2023), and educational web text from FineWeb-Edu (Penedo et al., 2024). Each run uses 40B training tokens, 32K-token sequences, and 2K-token blocks. The main comparison uses a 100M-parameter backbone; model-scaling experiments extend it to 400M. We compare against Transformer++, a memory-free reference, and three representative memory architectures: Memorizing Transformer (Wu et al., 2022), Infini-Transformer (Munkhdalai et al., 2024), and Melodi (Chen et al., 2025). All methods share the backbone, data order, and base training recipe. Appendix A.1 details their memory injection and implementation differences. This setting isolates the contribution of historical representation within a common language-modeling framework.

We evaluate both average-token prediction and prediction that depends on distant history. PPL measures the former; LongPPL evaluates key tokens selected for their dependence on long-range context (Fang et al., 2025). The second metric is especially relevant to memory: a mechanism can improve frequent local predictions while doing little for tokens that need information outside the current block. Reporting both metrics tests whether the historical representation benefits language modeling broadly and supports the long-range dependencies it is designed to preserve. Unless otherwise stated, memory budgets count persistent state elements across all augmented layers, with a common 0.8M-element budget in the main comparison.

Table 2: Language modeling results on PG-19, Proof-Pile, and FineWeb-Edu. All datasets use 32K-token sequences. Base model contains 100M parameters. We report PPL and LongPPL (lower is better). Memory size and additional parameters are counted for the whole model.

Model	Mem. Size	Add. Params	PG-19		Proof-Pile		FineWeb-Edu	
			PPL	LongPPL	PPL	LongPPL	PPL	LongPPL
Transformer++	0	0	11.232	15.955	3.322	3.866	12.897	29.882
Mem. Transformer	0.8M	96	11.050	15.794	3.305	3.843	12.924	29.963
Infini-Transformer	0.8M	96	10.907	9.588	2.950	2.736	12.736	24.417
Melodi	0.8M	1.5M	10.684	10.478	2.940	2.980	12.509	23.532
Elastic Memory$_{\text{uni}}$	0.8M	0	10.692	**8.646**	2.958	**2.560**	12.419	**10.915**
Elastic Memory$_{\text{exp}}$	0.8M	0	**10.651**	8.885	**2.915**	2.586	**12.318**	11.932

Better predictions from a compact state. Elastic Memory improves both metrics across all three domains (Table 2). A single retrieval configuration, exponential sampling, already beats every baseline on all six dataset–metric combinations. Its PPL is 10.651 on PG-19, 2.915 on Proof-Pile, and 12.318 on FineWeb-Edu, against the strongest baseline values of 10.684, 2.940, and 12.509. The same configuration reduces LongPPL to 8.885, 2.586, and 11.932, respectively. Thus, its strength extends from average-token prediction to tokens that require distant context, using the same state budget and **no additional trainable parameters**. The improvement holds across books, mathematical text, and web documents, despite their different content and dependency patterns.

The largest gain is in long-context-sensitive prediction. On FineWeb-Edu, Elastic Memory's best LongPPL is **10.915 versus 23.532**, a **53.6% reduction**; the corresponding best results on PG-19 and Proof-Pile are 8.646 versus 9.588 and 2.560 versus 2.736. These gains show the value of retaining useful information throughout a document. Ordinary PPL averages over many tokens that can be predicted locally, whereas LongPPL concentrates on the predictions for which historical context matters. The improvement on both metrics therefore establishes the quality of the compressed representation, with particularly large benefits where the reader needs distant information. Uniform sampling supplies the strongest LongPPL results; we examine reader adaptation separately below.

Less storage, stronger prediction. On Proof-Pile, **1× Elastic Memory outperforms 16× Memorizing Transformer in PPL**: 2.92 versus 2.94 (Table 3). The recurrent states contain 0.8M and 12.8M elements, respectively, with the same 100M backbone. The comparison tests how effectively the architectures use their allocated storage: Memorizing Transformer receives sixteen times as many

Table 3: Memory scaling experiments on Proof-Pile. The base model contains 100M parameters. *Add.* denotes additional trainable parameters, *PPL* denotes perplexity, and *Long.* denotes LongPPL.

Model	1x Memory (0.8M)			2x Memory (1.6M)			4x Memory (3.2M)			8x Memory (6.4M)			16x Memory (12.8M)		
	Add.	PPL	Long.	Add.	PPL	Long.	Add.	PPL	Long.	Add.	PPL	Long.	Add.	PPL	Long.
MemT.	96	3.31	3.84	96	3.28	3.76	96	3.12	3.22	96	3.01	2.86	96	2.94	2.48
Melodi	1.5M	2.94	2.98	3.1M	2.93	2.97	6.4M	2.88	2.84	13.5M	2.84	2.67	30.1M	2.84	2.63
Elas.$_{\text{uni}}$	0	2.96	**2.56**	0	2.91	**2.39**	0	2.86	**2.28**	0	2.81	**2.19**	0	2.78	**2.14**
Elas.$_{\text{exp}}$	0	**2.92**	2.59	0	**2.86**	2.43	0	**2.82**	2.31	0	**2.78**	2.22	0	**2.75**	2.15

state elements. Elastic Memory achieves a better prediction score while representing that history with one-sixteenth of the state. This is a direct benefit of compression quality: a compact approximation can supply more useful context to the model than a much larger queue of historical entries.

Increasing the state budget then provides further gains. Across the five allocations from $1\times$ to $16\times$, PPL decreases from 2.92 to 2.75, and LongPPL from 2.59 to 2.15 for the same retrieval configuration. Both sampling configurations improve monotonically on both metrics. The backbone width and trainable parameter count remain unchanged throughout this sweep. Each state already represents the processed history at $1\times$; increasing polynomial order gives that representation more capacity to retain its detail. Additional storage is therefore converted into better prediction without requiring a larger language model. This makes memory capacity a useful performance axis alongside backbone scaling.

The advantage also holds against a learned compressor. Elastic Memory at $2\times$ memory outperforms Melodi at $4\times$ on both metrics, with PPL 2.86 versus 2.88 and LongPPL 2.43 versus 2.84; the same pattern holds at $4\times$ versus $8\times$. Both architectures compress the complete history at every reported budget, so these comparisons directly expose the quality obtained from each allocation. At $16\times$, Melodi adds 30.1M trainable parameters to the 100M backbone and reaches 2.84 PPL. Elastic Memory reaches 2.75 with no added weights. Its capacity growth improves historical representation without requiring an increasingly large learned summary module. Appendix A.1 provides the state formulas and context-coverage accounting behind these comparisons.

Table 4: Model scaling experiments on Proof-Pile, with model scales from 100M to 200M and 400M parameters. We report PPL and LongPPL (lower is better). Infini-Transformer's memory size is tied with head dimensions, thus other memory models adjust their memory size to align with it.

Model	1x Params (100M)			2x Params (200M)			4x Params (400M)		
	Mem.	PPL	LongPPL	Mem.	PPL	LongPPL	Mem.	PPL	LongPPL
Trans.++	0	3.322	3.866	0	3.226	3.757	0	3.206	3.730
MemTrans.	0.8M	3.305	3.843	1.2M	3.213	3.715	1.6M	3.187	3.689
InfiniTrans	0.8M	2.950	2.736	1.2M	2.796	2.484	1.6M	2.772	2.410
Melodi	0.8M	2.940	2.980	1.2M	2.840	2.857	1.6M	2.771	2.719
Elastic.$_{\text{uni}}$	0.8M	2.958	**2.560**	1.2M	2.824	**2.335**	1.6M	2.764	**2.238**
Elastic.$_{\text{exp}}$	0.8M	**2.915**	2.586	1.2M	**2.775**	2.364	1.6M	**2.737**	2.273

The advantage persists as the backbone grows. Table 4 increases the backbone from 100M to 200M and 400M parameters, aligning state budgets with Infini-Transformer at each scale. Elastic Memory retains the best PPL and LongPPL across the sweep. At 400M, the configuration with the lowest PPL achieves 2.737 PPL and 2.273 LongPPL, improving on the strongest baseline values of 2.771 and 2.410. A stronger backbone therefore continues to benefit from a better historical representation. The result complements memory scaling: increasing the memory budget improves prediction at fixed model size, while increasing model size preserves the advantage of the memory mechanism. Together, these sweeps show that Elastic Memory's gains extend beyond the base 100M comparison.

The quality improvement comes with competitive training throughput. At 100M and the base memory budget, Elastic Memory processes about $1.06\times$ as many samples per second as Transformer++. At 400M, it achieves **$1.52\times$ Melodi's training throughput** and remains close to Transformer++. These measurements matter because additional memory operations sit inside each training step: useful

historical context must be supplied efficiently to remain attractive as the backbone grows. Parallel block updates and reconstruction deliver strong prediction quality at these measured rates. Larger states incur additional update and reconstruction work; Appendix B.4 reports the complete training and evaluation throughput sweeps alongside the model and memory budgets.

Table 5: Retrieval adaptation without retraining on PG-19, Proof-Pile, and FineWeb-Edu. Labels specify sampling during training and evaluation: uni-exp switches uniform to exponential retrieval, and exp-uni switches in the opposite direction. Each entry reports *PPL / LongPPL*.

Model	PG-19	Proof-Pile	FineWeb-Edu
Elastic Memory$_{\text{uni}}$	10.69 / 8.65	2.96 / **2.56**	12.42 / **10.92**
Elastic Memory$_{\text{uni-exp}}$	10.66 / 8.97	2.93 / 2.61	12.38 / 12.15
Elastic Memory$_{\text{exp}}$	**10.65** / 8.89	**2.92** / 2.59	**12.32** / 11.93
Elastic Memory$_{\text{exp-uni}}$	10.70 / **8.60**	2.97 / 2.57	12.52 / 11.45

Better predictions without retraining. The polynomial reader provides an additional benefit after a model has been trained. Table 5 changes the sampling policy only at evaluation time, reusing the same checkpoint. Switching uniform-trained models to exponential retrieval improves PPL in all three domains; switching exponential-trained models to uniform retrieval improves LongPPL in all three. On PG-19, the latter switch reaches 8.60 LongPPL, below either model evaluated with its original policy. This intervention exposes a useful property of the representation: the stored coefficients support alternative ways of presenting history to attention. Improving the selected prediction metric requires neither new memory weights nor another training run. Appendix B.2 shows that the same directional benefits persist across memory and model scales.

Supporting analyses connect these results to the information held in the state. Increasing polynomial order from 540 to 8,640 reduces key/value reconstruction MSE from 0.45 to 0.27 while PPL improves from 2.92 to 2.75 (Appendix B.1). Under 50% all-layer local-context corruption, Elastic Memory also retains lower PPL and LongPPL than Memorizing Transformer (Appendix B.3). The first analysis measures representation fidelity; the second tests whether historical information remains useful when current-block features are disrupted. Both strengthen the evidence that the compressed state provides useful context for prediction.

3.2 Continued Pretraining: Quality and Efficiency

Setup. We adapt released Llama-2 7B and 13B checkpoints on 524.3M RedPajama tokens using 8K sequences, then evaluate PG-19 PPL from 2K to 32K. Elastic Memory uses uniform retrieval in an all-layer configuration. FreqKV (Kai et al., 2026), MemTransform, and Transformer++ share our CPT budget and evaluator. Table 6 also includes published full fine-tuning, LoCoCo (Cai et al., 2024a), and LongLoRA (Chen et al., 2024) results, marked with stars. Alongside PPL, we report steady-state training throughput during CPT on 8K sequences. Appendix A.2 gives the full configurations.

FreqKV-level quality with faster training. Elastic Memory matches the long-context quality of the strong FreqKV baseline while accelerating training. At 32K, their PPL values are 7.08 versus 7.02 at 7B and 6.42 versus 6.41 at 13B. At 7B, Elastic Memory trains at 64,498 tokens/s against FreqKV's 35,211, a **1.83× throughput gain**. The efficiency advantage also holds at 13B, with **1.71× training throughput**. These results make the practical benefit of the memory design clear: it delivers competitive long-context quality with substantially higher training throughput.

8K adaptation unlocks 32K language modeling. Elastic Memory reaches **four times the CPT sequence length** without expanding its recurrent state or using 32K training sequences. At 32K it obtains 7.08 PPL at 7B and 6.42 at 13B, compared with Transformer++'s 7.76 and 7.04: **an 8.8% reduction at both sizes**. This improvement follows just 1,000 CPT steps on the shorter sequences. Between 8K and 32K, Elastic Memory's PPL changes from 7.07 to 7.08 at 7B and improves from 6.44 to 6.42 at 13B. The practical result is access to a much longer evaluation context while retaining the language-modeling quality reached at the adaptation length.

Persistent history pays off beyond the recent window. The complete length sweep shows where the memory advantage becomes important. At 13B, Elastic Memory and Transformer++ both achieve

Table 6: PG-19 perplexity after CPT on 8K sequences, evaluated at 2K–32K. Throughput measures steady-state training tokens/s. Stars mark published references; other rows use our matched CPT protocol. Dashes denote unreported results. Bold marks the lowest PPL and highest throughput within each model size, including ties.

Size	Method	2048	4096	8192	16384	32768	Throughput
7B	Full FT*	7.55	7.21	**6.98**	–	–	–
	LoCoCo*	8.15	8.08	7.27	–	–	–
	LongLoRA*	7.70	7.35	7.14	–	–	–
	FreqKV	**7.45**	**7.12**	7.05	**7.03**	**7.02**	35,211
	MemTransform	**7.45**	**7.12**	7.16	7.17	7.18	**76,340**
	Transformer++	7.48	7.15	7.01	7.35	7.76	61,831
	Elastic Memory	**7.45**	**7.12**	7.07	7.08	7.08	64,498
13B	Full FT*	6.95	6.60	6.43	–	–	–
	LongLoRA*	7.03	6.73	6.58	–	–	–
	FreqKV	6.83	6.53	6.44	**6.42**	**6.41**	22,508
	MemTransform	6.83	6.53	6.55	6.55	6.55	**46,797**
	Transformer++	6.83	**6.52**	**6.38**	6.66	7.04	37,623
	Elastic Memory	**6.82**	**6.52**	6.44	**6.42**	6.42	38,578

6.52 PPL at 4K; their gap grows to 0.24 points at 16K and 0.62 at 32K. At 7B, Transformer++ degrades by 0.75 points between 8K and 32K, whereas Elastic Memory changes by only 0.01. Elastic Memory also improves on MemTransform at 32K at both sizes. As history exceeds the recent window, a persistent compressed state supplies information that a bounded-window model no longer retains. Together with the throughput gains over FreqKV, this establishes an efficient route to long-context adaptation of existing models.

3.3 Supervised Fine-Tuning: Long-Context Task Performance

Setup. We evaluate models after LongAlpaca SFT initialized from Llama-3-8B-Instruct. LongBench (Bai et al., 2024) covers 14 tasks in question answering, summarization, few-shot learning, and code completion, using a 50% retaining ratio. Needle-in-a-Haystack (NIAH) evaluates keyword retrieval over 310 length–depth combinations. These complementary tests examine whether a compact historical representation supports both task completion and access to a specific fact. Elastic Memory and FreqKV use our SFT checkpoints; the other LongBench rows reproduce published results, as detailed in Appendix A.3. All NIAH methods use a common SFT recipe and evaluation grid.

Table 7: LongBench results on Llama-3-8B-Instruct at a 50% retaining ratio. Elastic. denotes Elastic Memory; bold values mark the best score in each column. Elastic Memory and FreqKV use LongAlpaca SFT checkpoints; other rows reproduce published results (Appendix A.3).

Method	Single-Doc. QA			Multi-Doc. QA			Summarization			Few-shot			Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RB-P	
LaCache	22.95	28.91	40.99	46.34	36.26	23.02	16.92	16.02	15.59	74.00	**91.11**	32.97	55.09	49.31	39.25
D2O	**26.61**	28.29	38.85	44.95	36.57	22.90	27.66	23.34	25.62	74.00	90.31	41.79	57.75	54.54	42.36
GemFilter	23.60	29.26	43.71	42.18	36.35	**24.48**	17.75	16.46	15.70	73.00	90.79	33.30	53.19	38.54	38.45
FastKV	23.88	29.67	40.75	44.66	34.02	23.07	16.86	16.45	15.61	74.00	90.31	42.02	59.21	53.30	39.63
SnapKV	22.99	31.89	41.11	43.92	36.42	23.38	17.14	16.45	15.53	**74.50**	90.23	33.72	59.03	53.97	40.02
ThinK	21.98	30.97	41.90	45.10	34.18	22.51	27.73	23.38	26.40	**74.50**	90.16	41.50	62.05	**58.37**	42.91
PyramidKV	22.27	31.93	41.03	44.79	34.56	23.86	28.77	23.14	26.62	**74.50**	90.23	**42.53**	59.11	53.33	42.76
FreqKV	23.14	**41.60**	43.62	**46.48**	**40.54**	20.20	29.99	**23.67**	**28.58**	69.00	88.95	41.47	60.33	51.11	43.48
Elastic.	22.56	40.14	**46.51**	45.32	39.43	17.86	**30.97**	22.59	28.53	69.00	88.17	42.20	**62.09**	56.42	**43.70**

Leading LongBench performance, with strong code results. Elastic Memory achieves the **highest average score, 43.70**, across the 14 LongBench tasks (Table 7). The aggregate comparison covers

several types of generated outputs, from document answers and summaries to code completions. This extends the pretraining evidence beyond language-modeling loss: the reconstructed history supports useful behavior after instruction tuning. The strongest gains over the matched FreqKV comparison appear in code completion, where Elastic Memory improves LCC by 1.76 points and RepoBench-P by **5.31 points**. These two tasks average 59.26 for Elastic Memory versus 55.72 for FreqKV, showing a consistent advantage within that category.

The task-level results make the benefit of historical representation concrete. On RepoBench-P, Elastic Memory reaches 56.42 against FreqKV's 51.11, while its LCC score reaches 62.09. Outside code, it improves MultiFieldQA-en from 43.62 to 46.51 and achieves the strongest GovReport score in the table, 30.97. These gains span tasks that require extracting or continuing information from long inputs. Their contribution to the overall lead shows that compression preserves information useful to task generation, complementing the lower prediction losses observed in pretraining. The full task breakdown remains in the main text so that the aggregate result can be assessed alongside its component scores.

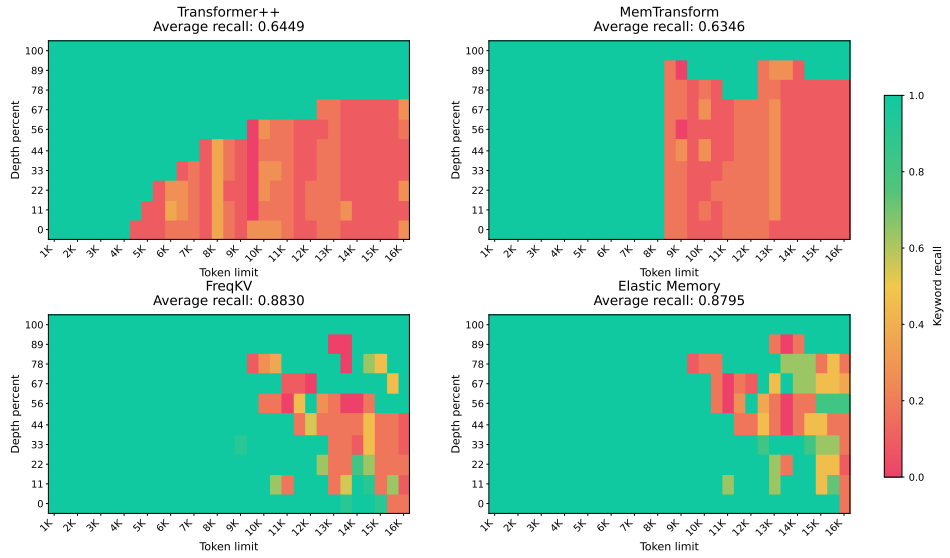


Figure 2: NIAH keyword recall after LongAlpaca SFT. All four methods use the same 31 context lengths (1K–16K), ten needle depths (0–100%), and shared color scale. Panel titles report average recall over the 310 cases.

Specific facts remain accessible after compression. NIAH directly probes information retrieval at different historical positions. Elastic Memory reaches **87.95% average keyword recall**, exceeding Transformer++ and MemTransform by 23.46 and 24.49 percentage points and closely matching FreqKV's 88.30% (Figure 2). The shared grid spans 31 lengths from 1K to 16K and ten insertion depths. Elastic Memory preserves broad coverage at longer lengths, where the window and raw-memory comparators lose access to many needles. This result answers a complementary question to LongBench: the compressed history can support retrieval of a designated fact as well as useful task-level generation. The reported metric measures answer-word recovery across the grid.

4 Conclusion

Elastic Memory turns a compact polynomial state into useful historical context for Transformer attention. Its parallel updates and sampling-based reconstruction provide scalable memory without additional trainable parameters. Controlled pretraining demonstrates strong prediction quality and effective use of increasing state capacity, while CPT and SFT extend the evidence to released language models and downstream tasks. These results establish better historical representation and independently scalable memory as practical routes to stronger long-context models.

AI USE STATEMENT

Generative AI tools assisted with coding and polishing the manuscript text. The authors take responsibility for the final code, claims, and written content.

ETHICS STATEMENT

All experiments use publicly available datasets. We do not identify any specific ethical concerns arising from this work.

REPRODUCIBILITY STATEMENT

Section 2 and Algorithm 1 specify the architecture and forward pass. Appendix A documents data preparation, training settings, baseline configurations, and evaluation protocols; Appendix B provides supporting experiments; and Appendix C gives the mathematical derivations and resource accounting. Code will be released to support reproduction of the results.

REFERENCES

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 11079–11091, 2022.

Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. Lococo: Dropping in convolutions for long context compression. *arXiv preprint arXiv:2406.05317*, 2024a.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024b.

Yinpeng Chen, DeLesley Hutchins, Aren Jansen, Andrey Zhmoginov, David Racz, and Jesper Andersen. MELODI: Exploring memory compression for long contexts. In *International Conference on Learning Representations (ICLR)*, 2025.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8220–8238, 2024.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3829–3846, 2023.

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 the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2978–2988, 2019.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.23771. arXiv:2410.23771.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1474–1487, 2020.

Ziwei He, Meng Yang, Minwei Feng, Jingcheng Yin, Xinbing Wang, Jingwen Leng, and Zhouhan Lin. Fourier transformer: Fast long range modeling by removing sequence redundancy with FFT operator. *arXiv preprint arXiv:2305.15099*, 2023.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 33248–33261, 2022.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. Fastkv: Decoupling of context reduction and kv cache compression for prefill-decoding acceleration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32167–32186, 2026.

Jushi Kai, Yixuan Wang, Boyi Zeng, Haoli Bai, Bo Jiang, Ziwei He, and Zhouhan Lin. FreqKV: Key-value compression in frequency domain for context window extension. In *International Conference on Learning Representations (ICLR)*, 2026.

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.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Meta AI. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024a. URL `https://arxiv.org/abs/2407.21783`.

Meta AI. Meta llama 3.1 8b model card. `https://huggingface.co/meta-llama/Meta-Llama-3.1-8B`, 2024b.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. URL `https://openreview.net/forum?id=n6SCkn2QaG`.

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. URL `https://openreview.net/forum?id=SylKikSYDH`.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, pp. 9355–9366. PMLR, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. *Neural Computation*, 4(1):131–157, 1992.

Dachuan Shi, Yonggan Fu, Xiangchi Yuan, Zhongzhi Yu, Haoran You, Sixu Li, Xin Dong, Jan Kautz, Pavlo Molchanov, et al. Lacache: Ladder-shaped kv caching for efficient long-context modeling of large language models. *arXiv preprint arXiv:2507.14204*, 2025.

Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13839–13857, 2026.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42661–42688, 2023.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, et al. D2o: Dynamic discriminative operations for efficient long-context inference of large language models. *arXiv preprint arXiv:2406.13035*, 2024. URL https://arxiv.org/abs/2406.13035.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

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

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. In *International Conference on Learning Representations*, volume 2025, pp. 56691–56709, 2025.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.06464.

APPENDIX CONTENTS

The appendix collects experimental protocols, supporting analyses, mathematical and implementation details, related work, and limitations. Page numbers link to the corresponding sections.

A Experimental Protocols 13
A.1 Pretraining from Scratch 13
A.2 Continued Pretraining 14
A.3 Supervised Fine-Tuning and Evaluation 15
B Additional Pretraining Analyses 16
B.1 Representation and Reconstruction Quality 16
B.2 Retrieval Adaptation without Retraining 17
B.3 Sensitivity to Local-Context Corruption 17
B.4 Training and Evaluation Efficiency 17
C Mathematical and Implementation Details 18
C.1 Polynomial Projection and HiPPO-LegS 18
C.2 Discretization and Initialization 19
C.3 Block Update Derivation 19
C.4 Polynomial Reconstruction and Sampling 20
C.5 Computational and Storage Accounting 20
D Related Work 20

A EXPERIMENTAL PROTOCOLS

We provide separate protocols for pretraining from scratch, continued pretraining, and supervised fine-tuning, corresponding to the three settings in Section 3.

A.1 PRETRAINING FROM SCRATCH

Data preparation. We use PG-19, Proof-Pile, and FineWeb-Edu with the Llama 2 tokenizer. In each corpus, we retain documents longer than 32,768 tokens, split them into non-overlapping 32,768-token sequences, and discard trailing incomplete segments. Each resulting corpus contains approximately 2B tokens. Models are trained for 20 epochs, totaling approximately 40B tokens per run, with the same data order across methods.

Backbone and baseline implementations. All methods use a unified Llama-style backbone with modifications to the attention or memory modules. Transformer++ provides the memory-free reference. For Memorizing Transformer, we replace kNN retrieval with end-to-end trainable dense attention, following the comparison setup of Chen et al. (2025). Infini-Transformer includes RMSNorm for training stability. Melodi uses intra-layer memory propagation to align its recurrent structure with the other methods. Memorizing Transformer and Elastic Memory augment the ninth layer; Infini-Transformer and Melodi augment every layer. The resulting state sizes and additional trainable parameters are counted across all augmented layers in the experimental tables; Table 1 gives the per-layer formulas.

Optimization and infrastructure. Training uses eight NVIDIA A800 GPUs, BF16, and FlashAttention2. A global batch of 16 sequences contains 524,288 tokens. Each memory-augmented layer processes a 32,768-token sequence in 2,048-token blocks. Table 8 lists the remaining hyperparameters. If a run exhibits loss spikes, it is restarted with twice as many warmup steps until training stabilizes.

Evaluation metrics. PPL is computed using sliding-window evaluation on contiguous text. LongPPL uses the released implementation of Fang et al. (2025) with default hyperparameters. Llama 3.1-8B selects the long-context-dependent key tokens; its native 128K context covers the 32K evaluation sequences (Meta AI, 2024b;a). PPL and LongPPL evaluate different token populations, so their absolute values should be interpreted separately.

Table 8: Base hyperparameters for pretraining from scratch.

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	2e-3
Adam β_1	0.9
Adam β_2	0.95
Adam ϵ	1e-7
Weight Decay	0.1
Max Gradient Norm	1.0
LR Scheduler	Cosine
Initial Warmup Ratio	0.01
Training Epochs	20
Batch Size per Device	2
Gradient Accumulation Steps	1
Context Length	32,768
Chunk Size	2,048
Precision	BF16
Random Seed	42

Memory budgets and context coverage. The $1\times$ budget is aligned with Infini-Transformer's model-wide recurrent state, accumulated over its augmented layers. Its per-layer state contains $d_{\text{key}}d_{\text{value}}H$ elements. Elastic Memory scales capacity through polynomial order N: 540, 1,080, 2,160, 4,320, and 8,640 for $1\times$, $2\times$, $4\times$, $8\times$, and $16\times$ memory. These counts describe persistent recurrent state rather than total GPU memory, which also includes activations and implementation buffers. In model-scaling experiments, the total state budget increases from 0.8M to 1.2M and 1.6M elements alongside the backbone.

Memorizing Transformer retains a finite FIFO queue, whose capacity grows with the allocated state budget. Infini-Transformer, Melodi, and Elastic Memory compress the complete history at every reported memory size. For these compressive methods, increasing capacity improves representation fidelity without changing the length of history represented.

A.2 Continued Pretraining

The CPT study starts from released Llama-2 7B and 13B checkpoints and uses a tokenized RedPajama-Data-1T-Sample subset. The adaptation recipe updates embeddings, normalization parameters, and the adapters enabled by each baseline configuration. Elastic Memory's compression and reconstruction operators remain fixed and add no trainable weights. Training lasts 1,000 optimizer steps, or 524.3M tokens. Table 9 specifies the optimization and evaluation settings.

Memory and positional configurations. The matched CPT runs use four sink tokens and a 4,096-slot state/cache budget. Memory-based methods allocate 2,046 memory entries and 2,046 local entries where applicable. The configurations are:

- **Elastic Memory**: all-layer spectral/ZOH states of dimension 2,046, FP32 recurrent-state storage, recursive slots, uniform reconstruction, an exact 4,096-token prefix, pre-RoPE key compression, and compact RoPE in the transfer attention path.
- **FreqKV**: the released all-layer DCT implementation, retaining ratio 0.5, local block length 2,046, cache size 4,096, and four sink tokens.
- **MemTransform**: all-layer raw K/V external memory with an exact reader, 2,046 memory entries, 2,046 local entries, an exact 4,096-token prefix, and aligned compact RoPE.
- **Transformer++**: four sink tokens and the most recent 4,092 tokens at each layer, without an external memory.

Table 9: Continued-pretraining and PG-19 evaluation protocol.

Item	Setting
Base checkpoints	Llama-2 7B and Llama-2 13B
Training corpus	tokenized RedPajama-Data-1T-Sample subset
Corpus inventory	168,685 rows, 8,192 tokens per row
Sequence length	8,192
GPUs	8
Per-device batch	1 sequence
Gradient accumulation	8 steps
Global batch	64 sequences
Tokens per optimizer step	524,288
CPT duration	1,000 optimizer steps / 524.3M tokens
Optimizer	AdamW
Learning rate	2×10^{-5}
Weight decay	0
Schedule	constant after 20 warmup steps
Precision	BF16; TF32 enabled
Attention kernel	FlashAttention
Trainable modules	embeddings/norms and explicitly retained adapters
Seeds	model 42; data 42
Checkpoint interval	every 100 steps
Activation checkpointing	disabled for static CPT runs
Evaluation corpus	PG-19 test split
Evaluation lengths	2K, 4K, 8K, 16K, 32K
Evaluation stride	256
Evaluation parallelism	8 shards; all GPUs at 32K

These all-layer transfer configurations adapt the released checkpoints, whereas the from-scratch Elastic Memory experiments use ninth-layer injection. State/cache slots in this study are distinct from the model-wide element counts used in the pretraining tables.

Published references. The starred Full FT, LoCoCo, and LongLoRA entries in Table 6 are published results at an 8K training length, not reruns under our evaluator. Unstarred entries use the common CPT protocol above. We retain all reported evaluation lengths and mark unreported entries with dashes.

Training throughput. Table 6 reports steady-state throughput in tokens per second. Train TPS counts tokens processed during CPT on 8K sequences. These measurements accompany the PG-19 perplexity evaluation and compare training efficiency at each model size.

A.3 Supervised Fine-Tuning and Evaluation

Initialization and adaptation. Both the LongBench and NIAH studies start from Llama-3-8B-Instruct. Elastic Memory and FreqKV are evaluated after LongAlpaca SFT. The NIAH comparison applies a common SFT recipe to all four methods: 6,284 examples, 16K input sequences, five epochs, 490 optimizer steps, and global batch 64 across eight GPUs. We use rank-8 LoRA on Q/K/V/O, trainable embeddings and normalization parameters, AdamW with learning rate 2×10^{-5}, 20 warmup steps, and BF16 SDPA.

LongBench. Table 7 reports 14 tasks at a 50% retaining ratio, covering single-document QA, multi-document QA, summarization, few-shot learning, and code completion. The average is the unweighted mean over all 14 task scores. Elastic Memory and FreqKV use our SFT checkpoints and evaluation pipeline. The comparison rows for LaCache (Shi et al., 2025), D2O (Wan et al., 2024), GemFilter (Shi et al., 2026), FastKV (Jo et al., 2026), SnapKV (Li et al., 2024), ThinK (Xu et al., 2025), and PyramidKV (Cai et al., 2024b) reproduce published values for the corresponding task definitions and aggregation. Their training procedures follow the respective sources.

Needle-in-a-Haystack. The evaluation uses 49 Paul Graham documents as background text. We test 31 context lengths from 1K to 16K and ten needle depths spanning 0–100%, giving 310 length–depth cases per method. Greedy decoding generates at most 30 new tokens. Keyword recall measures recovery of the 11 answer words, and panel means average this score over the full grid; it is not whole-answer exact-match accuracy. ROUGE-1 multiplied by 10 is an auxiliary evaluation metric. All panels in Figure 2 share identical axes and a common 0–1 color scale.

B Additional Pretraining Analyses

B.1 Representation and Reconstruction Quality

The feature sequences produced by a language model can contain structure that a low-order approximation captures efficiently. FreqKV reports concentration of key/value energy at low frequencies, particularly in deeper layers (Kai et al., 2026); related structure has also been studied in Transformer representations (He et al., 2023). These observations motivate examining the reconstructed features directly. We use synthetic signals with known ground truth and key/value sequences from our trained models.

Synthetic signals. We generate composite sine waves $f(t) = \sum_i A_i \sin(2\pi\omega_i t + \phi_i)$, compress them with HiPPO-LegS, and measure the reconstruction MSE. Table 10 compares polynomial orders and discretization methods. A state of order 32 reconstructs the smooth test signals with low error; ZOH improves on forward and backward discretization for the five-sine signal, while bilinear discretization has the lowest error in that comparison. Random noise retains much larger error even as state size grows. The results illustrate the advantage of polynomial compression for structured signals and motivate the language-model reconstruction analysis below.

Table 10: HiPPO reconstruction quality on synthetic signals. Smooth signals have low reconstruction error; random noise retains substantially larger error at the tested orders.

Signal Type	HiPPO Dim N	Sample Length	Discretization	Reconstruct MSE
1 sine wave	32	1024	ZOH	1.2e-5
3 sine waves	32	1024	ZOH	2.3e-4
5 sine waves	32	1024	ZOH	9.8e-4
5 sine waves	32	1024	forward	3.3e-3
5 sine waves	32	1024	backward	2.9e-3
5 sine waves	32	1024	bilinear	4.8e-4
random noise	32	1024	ZOH	0.97
random noise	128	1024	ZOH	0.90
random noise	512	1024	ZOH	0.72

Language-model keys and values. Table 11 compares the original key/value sequences with their reconstructions on Proof-Pile. As polynomial order increases from 540 to 8,640, reconstruction MSE decreases from 0.45 to 0.27, alongside a PPL improvement from 2.92 to 2.75. The consistent trend connects increased memory capacity to a more faithful representation of history and complements the language-modeling scaling results.

Table 11: Reconstruction quality vs. performance on Proof-Pile. As HiPPO dimension N increases, reconstruction fidelity improves (lower MSE), corresponding to better language modeling performance (lower PPL).

Memory Size	1x	2x	4x	8x	16x
HiPPO Dimension N	540	1080	2160	4320	8640
Reconstruction MSE	0.45	0.41	0.35	0.31	0.27
PPL	2.92	2.86	2.82	2.78	2.75

Compression and direct sampling. Polynomial coefficients aggregate historical features before the reader selects reconstruction positions. Retaining a subset of raw entries instead preserves those entries exactly and omits the rest. The coefficient state provides a global approximation with a reader that can be changed after compression. Our memory-scaling results demonstrate the effectiveness of this allocation, while reconstruction MSE measures approximation fidelity rather than exact recovery of every historical token.

B.2 Retrieval Adaptation without Retraining

Table 12 extends the sampling-policy comparison to every memory and model scale on Proof-Pile. A label such as uni-exp denotes uniform sampling during training and exponential sampling at evaluation; uni and exp retain their training policy. All policy switches reuse the same checkpoint. Uniform-to-exponential switching improves PPL throughout the sweep, while exponential-to-uniform switching improves or matches LongPPL. The intervention changes only the reconstruction coordinates: the trained weights and state-update operators are unchanged. Thus, an existing checkpoint can improve the selected prediction metric without another training run; Table 5 reports the complementary three-domain comparison.

Table 12: Retrieval adaptation without retraining on Proof-Pile when scaling model size or memory size. Labels specify training and evaluation sampling policies as in Table 5. Each entry reports *PPL / LongPPL*.

Model	Scaling Memory					Scaling Model	
	1x	2x	4x	8x	16x	2x	4x
Elastic Memory$_{\text{uni}}$	2.96 / **2.56**	2.91 / **2.39**	2.86 / **2.28**	2.81 / **2.19**	2.78 / **2.14**	2.82 / **2.34**	2.76 / **2.24**
Elastic Memory$_{\text{uni-exp}}$	2.93 / 2.61	2.88 / 2.44	2.84 / 2.33	2.80 / 2.25	2.77 / 2.18	2.80 / 2.39	2.75 / 2.28
Elastic Memory$_{\text{exp}}$	**2.92** / 2.59	**2.86** / 2.43	**2.82** / 2.31	**2.78** / 2.22	**2.75** / 2.15	**2.78** / 2.36	**2.74** / 2.27
Elastic Memory$_{\text{exp-uni}}$	2.97 / 2.57	2.91 / 2.42	2.86 / 2.30	2.82 / 2.21	2.79 / **2.14**	2.82 / 2.35	2.78 / 2.26

B.3 Sensitivity to Local-Context Corruption

We replace 25%, 50%, or 75% of the current block's key and value tokens with random noise. The layer-local condition corrupts only the memory-augmented layer; the all-layer condition applies corruption at every layer. Table 13 reports PPL and LongPPL on Proof-Pile.

Elastic Memory retains lower PPL and LongPPL than Memorizing Transformer at all reported all-layer corruption rates. At 50% corruption, exponential retrieval reaches 4.37 PPL versus 5.12 for Memorizing Transformer, while uniform retrieval reaches 4.08 LongPPL versus 5.99. These results show that the historical representation remains useful when current-block features are disrupted, complementing the reconstruction measurements with a prediction-level perturbation test.

Table 13: Local context corruption experiments on Proof-Pile under different corruption rates. Each entry reports *PPL / LongPPL*.

Model	Origin.	Layer-Local Corruption			All-Layer Corruption		
		25%	50%	75%	25%	50%	75%
Mem. Transformer	3.31 / 3.84	3.31 / 3.85	3.32 / 3.85	**3.32** / 3.85	3.96 / 4.61	5.12 / 5.99	34.39 / 28.42
Elastic Memory$_{\text{uni}}$	2.96 / **2.56**	2.99 / **2.63**	3.02 / **2.69**	**3.32** / 3.86	3.55 / **3.18**	4.55 / **4.08**	**30.53 / 25.26**
Elastic Memory$_{\text{exp}}$	**2.91** / 2.59	**2.93** / 2.66	**2.95 / 2.69**	**3.32 / 3.84**	**3.48** / 3.22	**4.37** / 4.10	30.57 / 25.27

B.4 Training and Evaluation Efficiency

Tables 14 and 15 report training and evaluation throughput on Proof-Pile using eight A800 GPUs and 32,768-token samples. Training throughput is the Weights & Biases `train_samples_per_second` measurement; evaluation throughput is `eval/samples_per_second`. All entries use samples per second. Dividing the training entries by the 100M Transformer++ value, 14.689, gives a common normalized comparison across memory and model scales.

Elastic Memory sustains approximately baseline training throughput at $1\times$ memory and remains close to Transformer++ when scaling the backbone. At 400M, exponential retrieval reaches 8.177 samples/s versus 5.370 for Melodi. Increasing memory to $16\times$ raises reconstruction cost, giving approximately $0.79\times$ the 100M Transformer++ training throughput. The evaluation measurements preserve much of the throughput of the memory-free model across the memory sweep.

Table 14: Raw training throughput (samples/second). Higher is better.

Model	1x	Scaling Memory 2x	4x	8x	16x	Scaling Model 2x	4x
Transformer++	14.689	-	-	-	-	**12.930**	**8.500**
Mem. Transformer	**15.844**	**15.774**	**15.677**	**15.514**	**15.237**	12.531	8.297
Infini-Transformer	9.359	-	-	-	-	7.572	5.249
Melodi	8.052	8.523	8.518	8.403	7.851	7.777	5.370
Elastic Memory$_{\text{uni}}$	15.503	15.420	15.153	14.370	11.673	12.445	8.152
Elastic Memory$_{\text{exp}}$	15.530	15.471	15.179	14.331	11.663	12.491	8.177

Table 15: Evaluation throughput (samples/second). Higher is better.

Model	1x	Scaling Memory 2x	4x	8x	16x	Scaling Model 2x	4x
Transformer++	6.212	-	-	-	-	**6.379**	**5.913**
Mem. Transformer	**6.608**	**6.544**	**6.463**	**6.523**	**6.455**	6.287	5.873
Infini-Transformer	6.013	-	-	-	-	6.146	5.609
Melodi	5.552	5.558	5.646	5.738	5.653	5.698	5.078
Elastic Memory$_{\text{uni}}$	6.406	6.512	6.457	6.428	6.106	6.322	5.798
Elastic Memory$_{\text{exp}}$	6.498	6.480	6.444	6.367	6.126	6.338	5.848

C Mathematical and Implementation Details

We give the projection dynamics inherited from HiPPO (Gu et al., 2020), derive the block update and polynomial reader used in Elastic Memory, and specify the associated resource accounting. Throughout, tokens and blocks are indexed from zero: token k occupies $[k, k+1)$, and block i spans $[iL, (i+1)L)$. The coefficient vector $c_k = c(k)$ represents the history before token k.

C.1 Polynomial Projection and HiPPO-LegS

Projection objective. For $t > 0$, the uniform measure on $[0, t]$ induces

$$\langle f, g \rangle_{\mu^{(t)}} = \frac{1}{t} \int_0^t f(x) g(x)\, dx. \tag{7}$$

The scaled Legendre functions $g_n^{(t)}(x) = \sqrt{2n+1} P_n(2x/t - 1)$ are orthonormal under this measure. The minimizer of Eq. 1 therefore has coefficients

$$c_n(t) = \frac{1}{t} \int_0^t f(x) g_n^{(t)}(x)\, dx. \tag{8}$$

This is an L_2-optimal projection onto the chosen polynomial space. It specifies approximation fidelity for the represented feature signal, rather than an optimum of the language-modeling objective.

Coefficient dynamics. Differentiating the integral gives

$$\frac{dc_n}{dt} = -\frac{c_n(t)}{t} + \frac{f(t) g_n^{(t)}(t)}{t} + \frac{1}{t} \int_0^t f(x) \frac{\partial g_n^{(t)}(x)}{\partial t}\, dx. \tag{9}$$

With $z = 2x/t - 1$, the basis derivative is $\partial_t g_n^{(t)}(x) = -\sqrt{2n+1}(z+1)P_n'(z)/t$. Using

$$(z+1)P_n'(z) = nP_n(z) + \sum_{m=0}^{n-1}(2m+1)P_m(z) \tag{10}$$

and $g_n^{(t)}(t) = \sqrt{2n+1}$ yields the HiPPO-LegS dynamics

$$\frac{dc(t)}{dt} = -\frac{1}{t}\mathbf{A}c(t) + \frac{1}{t}\mathbf{B}f(t), \tag{11}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ and $\mathbf{B} \in \mathbb{R}^N$ are

$$A_{nm} = \begin{cases} \sqrt{(2n+1)(2m+1)}, & n > m, \\ n+1, & n = m, \\ 0, & n < m, \end{cases} \qquad B_n = \sqrt{2n+1}. \tag{12}$$

The diagonal contribution is $n+1$: differentiating the basis contributes n, and differentiating the normalization of the measure contributes 1. These matrices are determined by the basis and contain no learned parameters.

C.2 Discretization and Initialization

Zero-order hold. Represent token feature f_k as constant over $[k, k+1)$. For $k \geq 1$, the continuous matrices are scalar multiples of the fixed matrix $\mathbf{A}$ and therefore commute across time. Their transition is

$$\mathbf{\Phi}(t_2, t_1) = \exp\left(-\mathbf{A}\log\frac{t_2}{t_1}\right) = \left(\frac{t_1}{t_2}\right)^{\mathbf{A}}. \tag{13}$$

Consequently, Eq. 2 uses

$$\bar{\mathbf{A}}_k = \left(\frac{k}{k+1}\right)^{\mathbf{A}}, \tag{14}$$

$$\bar{\mathbf{B}}_k = \int_k^{k+1}\left(\frac{\tau}{k+1}\right)^{\mathbf{A}}\frac{\mathbf{B}}{\tau}\,d\tau = \mathbf{A}^{-1}(\mathbf{I} - \bar{\mathbf{A}}_k)\mathbf{B}. \tag{15}$$

Here a matrix power denotes the corresponding matrix exponential. The expression follows from the time-varying LegS dynamics under the piecewise-constant input convention.

The first interval. The empty history has state $c_0 = 0$ and supplies no memory entries to attention. On $[0, 1)$, the input is constant, so its exact projection has $c_1 = (f_0, 0, \ldots, 0)^\top$. This also follows from the limits $\bar{\mathbf{A}}_0 = 0$ and $\bar{\mathbf{B}}_0 = \mathbf{A}^{-1}\mathbf{B} = (1, 0, \ldots, 0)^\top$. Handling this interval separately avoids evaluating $\log((k+1)/k)$ at $k = 0$. Block updates then use the same indexing convention from the first block onward.

C.3 Block Update Derivation

Unrolling the token recurrence. Let $a = iL$ and $b = (i+1)L - 1$ be the first and last token indices of block i. Repeated substitution in Eq. 2 gives

$$c_{b+1} = \left(\prod_{k=b}^{a}\bar{\mathbf{A}}_k\right)c_a + \sum_{j=a}^{b}\left(\prod_{k=b}^{j+1}\bar{\mathbf{A}}_k\right)\bar{\mathbf{B}}_j f_j. \tag{16}$$

Products place later updates on the left; an empty product is the identity. This convention also covers the last token's direct contribution.

Matrix formulation. Stack the block features into $\mathbf{F}_i \in \mathbb{R}^{L\times d}$ and define

$$\mathbf{P}_i = \prod_{k=b}^{a}\bar{\mathbf{A}}_k, \qquad (\bar{\mathbf{K}}_i)_{:,r} = \left(\prod_{k=b}^{a+r+1}\bar{\mathbf{A}}_k\right)\bar{\mathbf{B}}_{a+r}, \quad 0 \leq r < L. \tag{17}$$

Applying the recurrence to every feature channel gives $\mathbf{C}_{i+1} = \mathbf{P}_i\mathbf{C}_i + \bar{\mathbf{K}}_i\mathbf{F}_i$. Thus, the block expression is algebraically equivalent to the token recurrence, with floating-point differences determined by the implementation. The operators depend on absolute block position through the LegS dynamics and can be prepared for the configured context range.

C.4 Polynomial Reconstruction and Sampling

Reconstructing the signal. Given a coefficient state after processing $t > 0$ positions, the reconstructed feature at $x \in [0, t)$ is

$$\hat{f}(x) = \sum_{n=0}^{N-1} c_n(t)\sqrt{2n+1}P_n\left(\frac{2x}{t} - 1\right). \tag{18}$$

Evaluating this expression at M positions yields the matrix reader $\hat{\mathbf{F}} = \mathbf{RC}$ in Eqs. 4–5. Sampling controls which parts of the represented signal attention reads.

Sampling coordinates. The uniform schedule uses $x_j = jt/M$ for $0 \le j < M$. An exponential schedule can be written as

$$x_j = t\left(1 - \alpha^{M-1-j}\right), \qquad 0 < \alpha < 1, \quad 0 \le j < M. \tag{19}$$

This ordering runs from recent positions toward the beginning, with denser spacing near the present. Switching the schedule changes $\mathbf{R}$ while preserving the coefficient state and its update operators. If normalized coordinates x_j/t are fixed across blocks, the reconstruction matrix is independent of absolute history length and can be reused; more general position schedules use the corresponding matrices from a precomputed bank.

Reconstruction coordinates and attention positions. The basis is evaluated at specified historical coordinates, making reconstruction position-aware. The reconstructed values approximate the feature signal at those coordinates; this property alone does not guarantee a unique position code inside each vector. Our from-scratch attention uses reconstructed memory directly, with RoPE applied to current queries and keys. The CPT configuration uses compact RoPE in its transfer attention path, with pre-RoPE features still supplying the stored key state (Appendix A.2). Algorithm 1 specifies the from-scratch forward pass and omits the history reader when the history is empty.

C.5 Computational and Storage Accounting

Persistent state. For H attention heads, key width d_k, and value width d_v, the K/V coefficient states contain $NH(d_k + d_v)$ elements per augmented layer. Their size is independent of the processed history length. Model-wide budgets sum over all augmented layers, whereas Table 1 reports per-layer formulas. The 16× state comparison in Section 3.1 refers to these elements, not a 16× reduction in total GPU memory.

Update, reconstruction, and attention. Writing $d = H(d_k + d_v)$, a direct dense implementation of the block update costs $O(N^2 d + NLd)$ operations and reconstruction costs $O(MNd)$. Attention over L current queries and $M + L$ keys costs $O(L(M + L)d)$ up to the usual head-width constants. For fixed N, M, L, these costs do not grow with the number of previously processed blocks. Increasing N therefore changes the storage and update/reconstruction costs without changing the trainable parameter count. The empirical throughput measurements in Appendix B.4 account for the actual implementation.

Operators and temporary storage. The analytic operators are shared across sequences using the same configuration. Caching dense transitions and kernels for B blocks uses $O(B(N^2 + NL))$ elements; reconstruction matrices contribute $O(BMN)$ if each block has a distinct reader, or $O(MN)$ when a normalized schedule permits sharing. These operator banks, reconstructed entries, activations, and attention workspaces are separate from the per-sequence recurrent state. Extending a configured context range may require preparing further operators, but does not require enlarging that state or learning new memory weights.

D Related Work

External and recurrent memory. Transformer-XL (Dai et al., 2019) reuses hidden states across segments to extend the context available to attention. kNN-LM (Khandelwal et al., 2020) augments

predictions through retrieval from an external datastore, and Memorizing Transformer (Wu et al., 2022) incorporates retrieved representations into a language model's attention. Focused Transformer (Tworkowski et al., 2023) studies how to distinguish useful memories from distracting entries, while LongMem (Wang et al., 2023) uses a side network to access a memory bank. These approaches expand the information available beyond the current segment through stored representations and a retrieval mechanism. Elastic Memory instead maintains coefficients of a historical approximation and reconstructs the features needed by attention, making representation capacity configurable without retaining a growing collection of raw entries.

Associative memory. Fast weight models (Schmidhuber, 1992) represent recent experience through dynamically updated states. Linear Transformers can be interpreted as fast weight programmers, with the delta rule improving the management of key–value associations (Schlag et al., 2021). Gated DeltaNet adds a forgetting mechanism to this update (Yang et al., 2025), and Infini-Transformer combines compressive associative memory with local attention (Munkhdalai et al., 2024). Their matrix states provide compact access to history; in the architectures considered here, the state dimensions are tied to head count and feature widths. Elastic Memory introduces polynomial order as an additional capacity variable, allowing the recurrent state to grow while those backbone dimensions remain fixed.

Learned summary memory. Block-Recurrent Transformers (Hutchins et al., 2022), Recurrent Memory Transformer (Bulatov et al., 2022), and AutoCompressor (Chevalier et al., 2023) propagate compact representations between segments. Melodi (Chen et al., 2025) organizes memory through short- and long-term summaries. These methods learn to condense historical information into a bounded interface for future computation. Elastic Memory uses an analytic update and reconstruction to supply that interface. Its memory-capacity sweep therefore changes the size of the represented state without adding the trainable summary parameters that grow with Melodi's capacity in our comparison.

Function approximation and spectral compression. HiPPO (Gu et al., 2020) formulates recurrent memory as online polynomial projection; its LegS dynamics are the foundation of our coefficient update. Elastic Memory builds on this theory with blockwise KV compression and a polynomial reader for attention. Spectral methods provide another route to compact representations (He et al., 2023). In particular, FreqKV (Kai et al., 2026) repeatedly compresses a bounded KV cache using low-frequency DCT components. Elastic Memory maintains an evolving polynomial coefficient state and chooses reconstruction positions when reading it. Our comparison with FreqKV evaluates these alternative compression mechanisms after CPT and SFT.

Efficient context adaptation. LongLoRA (Chen et al., 2024) reduces the cost of long-context fine-tuning using shifted local attention, while LoCoCo (Cai et al., 2024a) learns convolutional context compression. Cache-selection and compression approaches such as D2O (Wan et al., 2024), SnapKV (Li et al., 2024), and ThinK (Xu et al., 2025) reduce the historical features retained for attention. These methods address complementary choices about adaptation cost and context allocation. Our CPT study examines whether a recurrent polynomial representation can be adapted using short training sequences, and the LongBench study assesses the resulting task performance against available compression baselines.