GPA: General Principled Framework for Linearizing Softmax Attention via KV Cache Approximation

Ning Wang[1,2,3]* **Zekun Li**[1,2,3,*] **Tongxin Bai**[3]
Man Yao[1,2] **Chunlei Men**[3]† **Guoqi Li**[1,2,†]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Beijing Academy of Artificial Intelligence
Beijing, China

Abstract

Transformers excel at sequence modeling, yet Softmax attention incurs quadratic complexity and unbounded KV cache growth. While linear attention offers a promising alternative, existing approaches lack systematic functional comparison with Softmax attention, rigorous error analysis, and a theoretically grounded improvement roadmap. We address this gap by framing linearization as KV-cache approximation and establishing a compositional pathway from Softmax attention to bounded-state models. The pathway contains five steps—redundancy elimination and counting, content–position decoupling, positional compression, inter-layer state propagation, and multi-level residual decomposition. We give a unified approximation statement that makes their assumptions, capacity requirements, and accumulated errors explicit. Building on this framework, we introduce GPA, a pretrained-weight-inheriting linear attention model. Across commonsense, LongBench, retrieval, NIAH, and BABILong evaluations, GPA consistently improves over converted GSA and Gated DeltaNet baselines; it is also compatible with intra-layer and cross-layer hybrid attention. Detailed latency measurements show constant decoding-state memory and length-independent per-token decoding cost after the context has been encoded. These results provide both a clearer theory-to-architecture map and practical guidance for converting pretrained Softmax models into efficient recurrent alternatives. Code: `https://github.com/fisalt/GPA`.

1 Introduction

The Transformer architecture [1] underpins modern deep learning, powering state-of-the-art models across language [2, 3, 4, 5, 6], vision [7, 8], and multimodal [9] domains. Yet two fundamental limitations persist: the *quadratic complexity* of Softmax attention with respect to sequence length, and the *unbounded growth* of the key-value (KV) cache during autoregressive inference. These constraints severely limit Transformer applicability in long-sequence tasks such as video understanding [10], genomic analysis [11], and extended-context applications [12].

Two major research directions have emerged to address these challenges. *KV-cache compression* [13, 14] reduces memory by selecting tokens or compressing sequence and channel dimensions, but a fixed compression ratio does not by itself prevent growth with context length. *Linear attention and recurrent linear models* instead aggregate history into fixed-size states through separable feature maps, gating, forgetting, or Delta-Rule updates [15, 16, 17, 18, 19, 20]. Recent work improves

*Equal contribution.

†Corresponding authors: Chunlei Men (`clmen@baai.ac.cn`) and Guoqi Li (`guoqi.li@ia.ac.cn`).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

attention concentration, signed interactions, norm preservation, Softmax–linear integration, and Mamba-inspired gating [21, 22, 23, 24, 25]. Despite this progress, fixed-state models still exhibit a pronounced recall–capacity trade-off and often require training from scratch or retaining selected Softmax computation.

In this work, we revisit the connection between Softmax and linear attention through the lens of *KV cache compression*. We propose a theoretically grounded pathway for compressing Softmax attention into linear models, structured around five principles. Each principle establishes (i) the necessity of a specific compression step, (ii) its equivalence to mechanisms in existing approaches, and (iii) the error introduced relative to Softmax attention. Collectively, these principles delineate what linear attention must retain, what it can safely discard, and how it fundamentally differs from Softmax attention.

This analysis yields both theoretical and practical benefits. Theoretically, it distinguishes exact finite-type aggregation from the controlled approximations required to reach a practical state budget, and it states how local errors propagate across layers. Practically, it enables pretrained Softmax-based language models to be converted into linear or hybrid variants with modest continued training. Empirically, GPA narrows the gap on retrieval, language understanding, and long-context reasoning while exposing the remaining limitations of purely fixed-state models.

Contributions. First, we formulate Softmax-to-linear conversion as a sequence of five composable KV-cache approximations and provide an end-to-end state and error guarantee rather than treating the propositions as isolated observations. Second, we derive GPA and explicitly separate theoretically motivated components (counting, positional-state compression, capacity selection, and residual state decomposition) from engineering refinements (sigmoid routing and multi-channel reading). Third, we evaluate the resulting model under broader retrieval, stress-test, hybrid-attention, and systems settings, including matched-memory SnapKV, S-NIAH, BABILong, learned-threshold, state-size, and kernel-level latency analyses.

2 Background, Preliminaries, and Positioning

2.1 Transformers and Softmax Attention

The Transformer relies on attention to compute contextualized representations. Given input $\boldsymbol{X} \in \mathbb{R}^{t\times d}$, it is projected into queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$. Attention is computed as

$$\boldsymbol{O} = \text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}} + \boldsymbol{M}^s\right)\boldsymbol{V}, \boldsymbol{o}_i = \sum_{j=1}^{i} \frac{\exp\left(\frac{\boldsymbol{q}_i \boldsymbol{k}_j^\top}{\sqrt{d}}\right)}{\sum_{h=1}^{i}\exp\left(\frac{\boldsymbol{q}_i \boldsymbol{k}_h^\top}{\sqrt{d}}\right)}\boldsymbol{v}_j, \quad (1)$$

where $\boldsymbol{M}^s_{ij} = 0$ if $i \geq j$ and $-\infty$ elsewhere. The autoregressive form $\boldsymbol{o}_i$ uses row vectors $\boldsymbol{q}_i$, $\boldsymbol{k}_j$, $\boldsymbol{v}_j$ of $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$. This formulation entails quadratic complexity and unbounded KV cache growth.

KV-cache compression retains or consolidates selected historical representations, whereas recurrent linear models aggregate history into fixed-size states. Gated, Delta-Rule, and multi-state updates refine this recurrent representation. Appendix A.1 retains the detailed formulations and their relation to the five principles.

2.2 Linearizing Pretrained LLMs and Hybrid Designs

Training recurrent architectures from scratch is expensive, motivating methods that reuse a pretrained Softmax Transformer. SUPRA performs scalable uptraining [26]; LoLCATs first aligns linear-attention outputs with the teacher and then applies low-rank fine-tuning [27]; and Liger reuses pretrained key projections to construct gated recurrent updates without adding a new feature-map module [28]. These methods primarily optimize the conversion procedure or recurrent parameterization. GPA instead asks which information in the original KV cache must be preserved, which information can be compressed, and how the resulting approximation errors compose.

Hybrid models preserve high-fidelity Softmax computation for selected tokens or layers. STILL routes salient tokens to sparse Softmax attention while summarizing the remainder with linear attention [29]. KL-guided layer selection restores the Softmax layers that most reduce teacher–student divergence [30], while Jet-Nemotron jointly searches full-attention placement, linear blocks,

and hardware-aware configurations [31]. GPA is orthogonal to these choices: its state can replace the linear branch of an intra-layer hybrid model or the converted layers of a cross-layer hybrid model. Sections 4 and Appendix A.4 evaluate both compositions.

3 Method

Existing KV cache compression methods lack general applicability, fail to bound the KV cache growth, and lack a clear error analysis relative to Softmax Attention. Linear attention approaches lack a comprehensive understanding of their components from the Softmax Attention perspective, leaving no clear improvement strategies to match or surpass Softmax Attention performance.

To address these limitations, we present five theoretical principles with experimental validation, establishing a general principled pathway to compress Softmax Attention into linear attention. Each theoretical node provides rigorous error analysis and demonstrates equivalence to existing model operations, offering references and guidance for future improvements.

3.1 Necessity of Redundancy Removal

Proposition 1 (Necessity of Deduplication and Storage Upper Bound)**.** *Let each component of key vectors $\mathbf{k}_t \in \mathbb{R}^{d_k}$ and value vectors $\mathbf{v}_t \in \mathbb{R}^{d_v}$ be stored with b bits (e.g., $b = 16$ for `float16`, $b = 4$ for `int4`). For an arbitrary finite prefix $\{(\mathbf{k}_t, \mathbf{v}_t)\}_{t=1}^{T}$, define the set of distinct key–value pairs and their multiplicities as*

$$\mathcal{U} = \big\{(\mathbf{u}_i, c_i)\big\}_{i=1}^{C}, \tag{2}$$

where $\mathbf{u}_i = [\mathbf{k}_{p_i}; \mathbf{v}_{p_i}] \in \mathbb{R}^{d_k+d_v}$ and $c_i \in \mathbb{N}$ is its occurrence count. Exact merging is performed only when the stored bit patterns are equal, $\mathbf{u}_i = \mathbf{u}_j$; cosine matching is an approximate extension discussed below. Then:

1. ***Upper bound on** C:*

$$C \;\le\; 2^{\,b\,(d_k+d_v)}. \tag{3}$$

Storing each c_i with $b_c = \lceil \log_2 T \rceil$ bits, the total bits required are

$$S_b = C \times \big(b\,(d_k + d_v) + b_c\big) \;\le\; 2^{\,b\,(d_k+d_v)} \times \big(b\,(d_k + d_v) + b_c\big). \tag{4}$$

2. ***Losslessness (Attention Equivalence)**: For any query $\mathbf{q} \in \mathbb{R}^{d_k}$,*

$$\sum_{t=1}^{T} \frac{\exp(\mathbf{q}^\top \mathbf{k}_t)}{\sum_{s=1}^{T} \exp(\mathbf{q}^\top \mathbf{k}_s)} \mathbf{v}_t \;=\; \sum_{i=1}^{C} c_i \frac{\exp(\mathbf{q}^\top \mathbf{k}_i)}{\sum_{j=1}^{C} c_j \exp(\mathbf{q}^\top \mathbf{k}_j)} \mathbf{v}_i. \tag{5}$$

3. ***Connection to Delta Rule**: The deduplication-with-counting mechanism is equivalent to a Delta Rule that independently stores new information (see Corollary A.6.3 in the appendix).*

Proof provided in Appendix A.6.

Generalization. Exact matching can be relaxed to cosine similarity above a threshold (e.g., 0.9). Two states above the threshold are consolidated, which acts as vector quantization and introduces key, value, and count distortion. The deployed GPA model does *not* use a manually fixed hard threshold: normalized Softmax-style or sigmoid assignment weights are learned jointly with the KV representations, and the resulting assignments are typically concentrated near 0.9. Fixed thresholds are used only to expose the accuracy–compression trade-off in controlled analysis (Table 5).

Complexity Implication. Let C be a sequence-length-independent upper bound on the number of distinct KV pairs. Exact causal Softmax attention can then be evaluated over the C types and their counts using $\mathcal{O}(NCd)$ total computation and $\mathcal{O}(Cd)$ vector storage (plus $\mathcal{O}(C \log N)$ count bits). Thus, for fixed C, it is linear-time and fixed-state in the number of stored vectors and is expressively equivalent to Softmax attention on that finite-precision prefix. This is a complexity and representation statement, not a claim of practical efficiency: the worst-case value $C \le 2^{b(d_k+d_v)}$ is enormous. Propositions 2–5 progressively reduce or approximate this loose bound.

3.2 Tokenization and Positional Information Decoupling

To achieve stronger compression, we analyze sequence modeling in LLMs. Tokenization limits distinct types to vocabulary size V (e.g., $\sim$32K for LLaMA2, $\sim$128K for Qwen2.5). Channel mixing does not affect type diversity; positional encoding and token mixing do. We thus optimize these operations leveraging first-layer tokenization.

Proposition 2 (Necessity of Positional Information Decoupling)**.** *Let vocabulary size be V, with word embeddings $\boldsymbol{e}_v \in \mathbb{R}^d$ ($v = 1, \ldots, V$) and embedding matrix $\boldsymbol{E}_{vocab} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_V]^\top \in \mathbb{R}^{V \times d}$. For an input sequence of length T with token indices $w_1, \ldots, w_T \in \{1, \ldots, V\}$, let the first-layer input be*

$$\tilde{\boldsymbol{X}}^{(1)} = PE(\boldsymbol{E}_{seq}, \boldsymbol{P}) \in \mathbb{R}^{T \times d}, \tag{6}$$

where $\boldsymbol{E}_{seq} = [\boldsymbol{e}_{w_1}, \ldots, \boldsymbol{e}_{w_T}]^\top$ and $\boldsymbol{P}$ is a positional encoding matrix. Then $\tilde{\boldsymbol{X}}^{(1)}$ can be losslessly compressed as a triple $(\boldsymbol{E}, \boldsymbol{I}, \tilde{\boldsymbol{P}})$ with:

- $\boldsymbol{E} = \boldsymbol{E}_{vocab}$ *(shared vocabulary embeddings, no extra storage);*
- $\boldsymbol{I} \in \mathbb{N}^{T \times 1}$, $\boldsymbol{I}_t = w_t$*: The index of $\boldsymbol{E}_{vocab}$ corresponding to the tth $\tilde{\boldsymbol{X}}^{(1)}$;*
- $\tilde{\boldsymbol{P}} \in \mathbb{R}^{T' \times d}$ *(positional information after decoupling, T' constant depending on method), such that $\tilde{\boldsymbol{X}}^{(1)} = PE\big(\boldsymbol{E}[\boldsymbol{I}, :],\ \boldsymbol{P}(\tilde{\boldsymbol{P}}, \boldsymbol{I})\big)$ with $\boldsymbol{E}[\boldsymbol{I}, :]$ the row selection. For RoPE, $T' = 1$.*

Properties:

1. ***Upper bound on position-independent content types**: $|\mathcal{U}_{\text{content}}| \leq V$, where $\mathcal{U}_{\text{content}} = \{\boldsymbol{e}_{w_t} \mid t = 1, \ldots, T\}$. This bound does not apply to the position-entangled rows $\tilde{\boldsymbol{x}}_t^{(1)}$ before decoupling.*
2. ***Storage size** (with b bits per component): $\underbrace{V \cdot d \cdot b}_{\textit{embeddings}} + \underbrace{T \cdot \lceil \log_2 V \rceil}_{\textit{indices}} + \underbrace{T' \cdot d \cdot b}_{\tilde{\boldsymbol{P}}}$.*

   For RoPE in LLMs, $\tilde{\boldsymbol{P}}$ reduces to a single row vector, giving: $V \cdot d \cdot b + T \cdot \lceil \log_2 V \rceil + 1 \cdot d_k \cdot b$.

Proof in Appendix A.7. This provides lossless compression for the first layer.

3.3 Positional Information Compression

Building on Proposition 2, we further compress positional information to achieve the same upper bound as the position-agnostic KV cache, enabling full control of the first-layer upper bound at the vocabulary size and laying the foundation for fixed-state linear models.

Proposition 3 (Necessity of Positional Information Compression)**.** *Assume RoPE with head dimension d (even). Define rotation angles*

$$\theta_j = \text{base}^{-\frac{2j}{d}}, \quad j = 0, 1, \ldots, \frac{d}{2} - 1, \tag{7}$$

with base > 1*. For position index m, let compressed positional information be stored as a linear superposition*

$$p_m^{(t)} = p_{m1} + p_{m2} + \cdots + p_{mt}, \tag{8}$$

representing accumulated positions up to some reference. The RoPE encoding $f_p(\boldsymbol{x}_m, \boldsymbol{p}(m))$ combines vector $\boldsymbol{x}_m$ with positional code $\boldsymbol{p}(m)$:

$$\begin{aligned} f_p(\boldsymbol{x}_m, \boldsymbol{p}(m)) = &[x_0 \quad x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{d-1} \quad x_d] \odot \boldsymbol{p}^c(m) \\ &+ [x_1 \quad -x_0 \quad x_3 \quad -x_2 \quad \ldots \quad x_d \quad -x_{d-1}] \odot \boldsymbol{p}^s(m), \end{aligned} \tag{9}$$

where

$$\begin{aligned} \boldsymbol{p}^c(m) &= \big[\cos(m\theta_0) \quad \cos(m\theta_0) \quad \ldots \quad \cos(m\theta_{d/2-1}) \quad \cos(m\theta_{d/2-1})\big], \\ \boldsymbol{p}^s(m) &= \big[\sin(m\theta_0) \quad \sin(m\theta_0) \quad \ldots \quad \sin(m\theta_{d/2-1}) \quad \sin(m\theta_{d/2-1})\big]. \end{aligned} \tag{10}$$

Consider the approximate attention score:

$$\tilde{A}_{n,s} = e^{f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))^\top \boldsymbol{k}_s} \left(t + f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))^\top f_p(\boldsymbol{k}_s, \boldsymbol{p}^{(t)}(s)) - f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))^\top \boldsymbol{k}_s \right), \tag{11}$$

where t is a scalar (e.g., count) and $\boldsymbol{p}^{(t)}(s)$ is the compressed positional information up to step t.Their upper bounds correspond one-to-one with $\boldsymbol{k}_s$, and the upper bound of $\boldsymbol{k}_s$ is the lexicon size following the tokenizer in Proposition 2. If base *is sufficiently large such that for all relevant relative positions $\Delta = n - s$ we have $|\Delta|\theta_j \leq 1$ for every $j \geq 2$, then the approximation error satisfies*

$$|A_{n,s} - \tilde{A}_{n,s}|_j = O\left(\frac{n}{\text{base}^{4j/d}} \right). \tag{12}$$

Thus, for large base*, $\tilde{A}_{n,s}$ closely approximates exact attention, with error decaying as* base *increases. For small j, the error may be larger but is diluted by dimension normalization.*

Proof in Appendix A.8. The intuition for decoupling and compressing separately is that complex information often emerges from interactions among simple components. It is thus more efficient to store these interactions than all raw details, analogous to recording a function instead of all input-output pairs. When compressing decoupled information, we focus on its equivalent effect at the time of separation; for example, positional information can be approximated by a linear gating mechanism.

This compression is most effective in the **first layer**. In deeper layers, token mixing become entangled with positional cues, so simple redundancy reduction cannot maintain the vocabulary-size bound. We therefore introduce a **layer-wise similarity constraint**: the **expected cosine similarity** between adjacent hidden states acts as a **contractive factor**, yielding a recursive bound that restricts intermediate-layer memory to a **constant multiple of the vocabulary size**, ensuring a fixed-state regime while preserving expressivity.

3.4 Inter-layer Similarity and State Propagation

Proposition 4 (Inter-layer Similarity and Recursive Compression Upper Bound)**.** *Consider a Transformer with residual connections. Let layer l input be $\boldsymbol{X}^{(l)} \in \mathbb{R}^{T \times d}$ (T: sequence length, d: dimension). Define $\mathcal{U}_l = \{\boldsymbol{u}_i^{(l)}\}_{i=1}^{M_l}$ as the set of distinct row vectors of $\boldsymbol{X}^{(l)}$, with $M_l = |\mathcal{U}_l|$. Suppose numerical precision ensures Euclidean distance between any two distinct vectors is at least $\varepsilon > 0$ (i.e., $\|\boldsymbol{u} - \boldsymbol{v}\| \geq \varepsilon$ for $\boldsymbol{u} \neq \boldsymbol{v}$). Set threshold $\tau = 1 - \varepsilon^2/8$; then any two distinct vectors cannot both have cosine similarity $\geq \tau$ with the same vector. (Proof: if $\|\boldsymbol{u} - \boldsymbol{c}\| \leq r$ and $\|\boldsymbol{v} - \boldsymbol{c}\| \leq r$ with $r = \sqrt{2(1-\tau)}$, then $\|\boldsymbol{u} - \boldsymbol{v}\| \leq 2r < \varepsilon$, a contradiction.)*

For layers l and $l-1$, define the quantization function

$$Q_{ij} = \begin{cases} 1 & \text{if } \cos(\boldsymbol{u}_i^{(l)}, \boldsymbol{u}_j^{(l-1)}) \geq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Let

$$\mathbb{E}[Q(X^{(l)}, X^{(l-1)})] = p_l = \frac{1}{M_l} \sum_{i=1}^{M_l} \max_j Q_{ij}, \tag{14}$$

the proportion of unique vectors in layer l that achieve threshold similarity with some vector in layer $l-1$. Then

$$M_l \leq \min\left(\frac{M_{l-1}}{p_l},\ T \right). \tag{15}$$

Symmetrically,

$$M_{l-1} \leq \min\left(\frac{M_l}{p_l'},\ T \right), \tag{16}$$

where p_l' is the reverse proportion. M_0 is bounded by the vocabulary size, T is original sequence length, and the final-layer bound M_L also approaches vocabulary size due to vocabulary projection.

Proof is provided in Appendix A.9. Proposition 4 gives a *conditional* recursive bound: it is informative when the measured matching ratio p_l is bounded away from zero and degenerates to the sequence-length bound as $p_l \to 0$. It does not derive the fixed capacity used by the implementation. GPA

therefore introduces a user-set capacity multiplier c_l and allocates at most $c_l M_{l-1}$ effective states before the subsequent multi-level approximation. Relative to the theoretical requirement M_{l-1}/p_l, the normalized capacity shortfall is

$$\delta_l^{\text{cap}}(c_l, p_l) = \frac{[M_{l-1}/p_l - c_l M_{l-1}]_+}{M_{l-1}/p_l} = [1 - c_l p_l]_+. \tag{17}$$

This quantity is zero only when the chosen capacity covers the conditional upper bound; otherwise it produces a capacity-collision error that must be reduced by fine-tuning and residual decomposition. Appendix A.4 reports inter-layer similarity across architectures and the sensitivity to c_l, state size, and decomposition depth. Even moderate scaling can still imply a very large state budget for a 128K vocabulary, motivating the multi-level approximation below.

3.5 Multi-level State Decomposition and Enhanced Reading

Proposition 5 (Necessity of Multi-level State Decomposition)**.** *For a fixed storage space, a* multi-level decomposition *with m levels is defined as:*

- *Each level $i = 1, \ldots, m$ has a vocabulary matrix $\boldsymbol{C}^{(i)} \in \mathbb{R}^{V_i \times d}$ (V_i prototypes).*
- *Any vector $\boldsymbol{x} \in \mathbb{R}^d$ is represented as $\hat{\boldsymbol{x}} = \sum_{i=1}^m \boldsymbol{C}_{k_i}^{(i)}$ with $k_i \in \{1, \ldots, V_i\}$.*

The recursive storage update is

$$\begin{aligned} \boldsymbol{S}_t^{k,(i)} &= \mathit{diag}\Big(1 - \bar{f}^{(i)}(\boldsymbol{k}_t^{(i)})^\top\Big)\, \boldsymbol{S}_{t-1}^{k,(i)} + \bar{f}^{(i)}(\boldsymbol{k}_t^{(i)})^\top \boldsymbol{k}_t^{(i)}, \\ \boldsymbol{k}_t^{(i+1)} &= \boldsymbol{k}_t^{(i)} - \bar{f}^{(i)}(\boldsymbol{k}_t^{(i)}) \boldsymbol{S}_t^{k,(i)}, \end{aligned} \tag{18}$$

where $\bar{f}^{(i)}$ typically uses softmax or sigmoid, e.g., $\bar{f}^{(i)}(\boldsymbol{k}_t^{(i)}) = \sigma(\boldsymbol{k}_t^{(i)} {\boldsymbol{C}^{(i)}}^\top)$ and $\boldsymbol{C}^{(i)} = \boldsymbol{S}_t^{k,(i)}$.

The following hold:

1. ***Representation capacity**: the number of index tuples is $\prod_{i=1}^m V_i$ (V^m if all $V_i = V$), and this is the number of distinct reconstructions when the sum map is injective.*
2. ***Error bound**: if each learned level is contractive in expectation, $\mathbb{E}\|\boldsymbol{r}^{(i)}\|_2^2 \le \rho_i \mathbb{E}\|\boldsymbol{r}^{(i-1)}\|_2^2$ with $0 \le \rho_i < 1$, then $\mathbb{E}\big[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2\big] \le \mathbb{E}\|\boldsymbol{x}\|_2^2 \prod_{i=1}^m \rho_i$.*

Increasing m reduces error exponentially but lowers efficiency due to serial computation.

Direct query–state multiplication can limit readout expressiveness even when storage capacity is adequate. We therefore study hierarchical reading, e.g., Softmax$(q_t {S^k}_t){S^v}_t$, and an enhanced version with learned transformations, $(\sigma(q_t {S^k}_t)W_\sigma){S^v}_t$, where $\sigma(x) = 1/(1+e^{-x})$. A multi-channel extension adds a direct query path: $(q_t W_r + \sigma(q_t {S^k}_t)W_\sigma){S^v}_t$. This empirical refinement is related to MVA's first-order vocabulary and gives progressive gains in Table 8; it is not required for the capacity result in Proposition 5.

3.6 Final Linear Model Update Rules

Integrating the five theoretical principles, we present the complete update rules.

1. Initialization.

$$q_t = f_p(x_t W_Q, r_t^{(i)}), k_{pt} = f_p(x_t W_K, r_t^{(i)}), k_t^{(0)} = x_t W_K \in \mathbb{R}^{1\times d}, v_t^{(0)} = x_t W_V \in \mathbb{R}^{1\times d},$$
$$S^{k(i)}_0 = 0 \in \mathbb{R}^{m\times d}, n_0^{(i)} = 0 \in \mathbb{R}^{1\times m}, E_t^{(0)} = I_m, S^{k(i)}_t = S^{kv(i)}_t[\ldots, :d_k], S^{v(i)}_t = S^{kv(i)}_t[\ldots, d_k : d_k + d_v].$$

2. Memory assignment and normalization (Theory 1, with state size control from Theories 4 and 5). $f^{(i)}(k_t^{(i)})$ is the retrieval score against stored states, $n_t^{(i)}$ the effective accumulation count, and $\bar{f}^{(i)}$ the normalized storage weight.

$$f^{(i)}(k_t^{(i)}) = \sigma(S^{k(i)}_{t-1} k_t^{(i)\top})^\top, \quad n_t^{(i)} = n_{t-1}^{(i)} + f^{(i)}(k_t^{(i)}), \quad \bar{f}^{(i)}(k_t^{(i)}) = \frac{f^{(i)}(k_t^{(i)})}{\max(n_t^{(i)}, 1)}. \tag{19}$$

$$m_t^{(i)} = \{k_t^{(i)}, v_t^{(i)}\}_{\text{dim}=-1}, {S^{kv}}_t^{(i)} = \text{diag}\left(1 - \bar{f}^{(i)}(k_t^{(i)})^\top\right) {S^{kv}}_{t-1}^{(i)} + \bar{f}^{(i)}(k_t^{(i)})^\top m_t^{(i)}. \tag{20}$$

3. Positional state update and residual propagation (Theories 2 & 3). The decoupled positional state ${S^p}_t^{(i)}$ is integrated with the position-free KV state.

$${S^p}_t^{(i)} = {S^p}_{t-1}^{(i)} + f^{(i)}(k_t^{(i)})^\top (k_{pt} - k_t^{(0)}). \tag{21}$$

4. Cross-Level Coupling and Attention Statistics. (Theories 3 & 4 & 5).
Theory 4's upper bound determines the number of hierarchical levels and level-wise state sizes. The residual between the current input and its projection onto the fixed-capacity stored state becomes the input for the next level: $m_t^{(i+1)} = m_t^{(i)} - f^{(i)}(k_t^{(i)}){S^{kv}}_t^{(i)}$. After partitioning, the inter-level connection $R_t^{(i+1)}$ is computed, and correct retrieval routing is enforced via $a_t^{(i)}$. Multi-channel reading uses terms such as $q_t W_r$ and $\sigma(q_t {S^k}_t^{(i)})W_\sigma$. The term $c_t^{(i)} = n_t^{(i)} + q_t {S^p}_t^{(i)\top}$ comes from Theory 3's Taylor approximation and can be generalized to data-dependent decay.

$$m_t^{(i+1)} = m_t^{(i)} - f^{(i)}(k_t^{(i)}){S^{kv}}_t^{(i)}, \quad e_t^{(i)} = q_t W_r + \sigma(q_t {S^k}_t^{(i)})W_\sigma. \tag{22}$$

$$R_t^{(i+1)}\left[f(k_t^{(i)}), f^{(i+1)}(k_t^{(i)})\right] = 1, \quad a_t^{(i)} = e_t^{(i)} R_t^{(i)\top}, \quad c_t^{(i)} = n_t^{(i)} + q_t {S^p}_t^{(i)\top}. \tag{23}$$

5. Final Aggregation and Output (Theories 4 & 5).
The final output is obtained by normalizing the hierarchical attention statistics $b_t^{(i)}$ and aggregating them through the inter-level topological connections.

$$b_t^{(i)} = \frac{\exp\left(\sum_i \ln(a_t^{(i)})\right)}{a_t^{(i)}} + e_t^{(i)}, T_t^{(i)} = R_t^{(i)}\left({S^v}_t^{(i)} \cdot e_t^{(i)\top} \cdot c_t^{(i)\top}\right), o_t = \sum_i \frac{b_t^{(i)}}{b_t^{(i)} \cdot e_t^{(i)} \cdot c_t^{(i)}} T_t^{(i)}. \tag{24}$$

3.7 Compositional Guarantee and Architecture Mapping

The five steps play different roles: Proposition 1 establishes a finite set of KV types and preserves multiplicity; Propositions 2 and 3 reduce the first-layer content bound and approximate positional effects; Proposition 4 propagates a conditional capacity requirement to deeper layers; and Proposition 5 approximates that capacity using a practical fixed budget. The following result states what their composition guarantees.

Theorem 1 (Compositional GPA approximation)**.** *For layer ℓ, let $\boldsymbol{o}_{\ell,t}^{\text{SA}}$ and $\boldsymbol{o}_{\ell,t}^{\text{GPA}}$ denote the exact Softmax and GPA outputs, and assume $\|\boldsymbol{q}_{\ell,t}\|_2 \le B_{\ell,Q}$ and $\|\boldsymbol{v}_{\ell,s}\|_2 \le B_{\ell,V}$. If the inter-layer matching ratios are $p_1, \dots, p_\ell > 0$, then the number of effective types required before fixed-budget approximation obeys*

$$M_\ell \le \min\left(T, \frac{|\mathcal{V}|}{\prod_{r=1}^{\ell} p_r}\right). \tag{25}$$

Let $\eta_{\ell,t}^K$ and $\eta_{\ell,t}^V$ collect deduplication, capacity-collision, and final residual-reconstruction errors. Let $\epsilon_{\ell,t}^{\text{cnt}}$, $\epsilon_{\ell,t}^{\text{pos}}$, $\epsilon_{\ell,t}^{\text{score}}$, and $\epsilon_{\ell,t}^{\text{read}}$ denote count, positional-score, enhanced-score, and value-readout errors. Then

$$\begin{aligned}\|\boldsymbol{o}_{\ell,t}^{\text{SA}} - \boldsymbol{o}_{\ell,t}^{\text{GPA}}\|_2 \le 2B_{\ell,V}\left(\frac{B_{\ell,Q}}{\sqrt{d_\ell}}\eta_{\ell,t}^K + \epsilon_{\ell,t}^{\text{cnt}} + \epsilon_{\ell,t}^{\text{pos}} + \epsilon_{\ell,t}^{\text{score}}\right) \\ + \eta_{\ell,t}^V + \epsilon_{\ell,t}^{\text{read}}.\end{aligned} \tag{26}$$

For H_ℓ residual levels with contraction ratios $\rho_{\ell,i}^K, \rho_{\ell,i}^V < 1$, the residual components of $\eta_{\ell,t}^K$ and $\eta_{\ell,t}^V$ decrease multiplicatively as $\prod_i \rho_{\ell,i}^K$ and $\prod_i \rho_{\ell,i}^V$. If subsequent residual blocks are locally Λ_r-Lipschitz and $e_{\ell,t}$ denotes the right-hand side above, the end-to-end deviation is bounded by

$$E_{L,t} \le \sum_{\ell=1}^{L} e_{\ell,t} \prod_{r=\ell+1}^{L} \Lambda_r. \tag{27}$$

With exact merging and counting, exact positional representation, sufficient capacity, and zero residual/readout error, GPA recovers exact Softmax attention over the finite-precision sequence. With fixed level sizes, storage is independent of T and total recurrent computation is linear in T.

Appendix A.11 provides the proof and defines each error term. The counting update, positional state, level sizes, residual decomposition, and relation matrix are derived from the five propositions. Sigmoid routing and multi-channel reading are engineering refinements: they improve empirical readout but are not needed for the bounded-state argument. This separation makes clear which claims are theoretical and which are implementation choices.

4 Experiments

We explore converting LLMs to linear models via weight inheritance, providing experimental support for each of the five theoretical principles. The integrated model, GPA, is evaluated comprehensively. We use lm-evaluation-harness [32], LongBench, S-NIAH, and BABILong [33]. Fine-tuning uses LoRA [34] for efficient parameter updates. Baselines include MVA, GSA, GLA, Gated DeltaNet, RetNet, and SUPRA [26].

Validation of the five principles. Appendix A.3 reports component-wise checks. For Theory 1, Table 5 shows that exact matching preserves all reported scores, while more permissive merging degrades ARC/MMLU/SAMSum despite perfect passkey retrieval. For Theories 2–3, Table 6 shows an accuracy cost from positional compression, partly recovered by the refined variant. For Theory 4, Table 7 shows ARC/MMLU gains with larger intermediate-layer budgets; Table 19 examines adjacent-layer similarity across backbones. For Theory 5, Table 8 shows that adding vocabulary interaction to MVA raises the 2K passkey score from 20 to 40, but does not improve every task.

Table 1: Comprehensive evaluation of GPA against state-of-the-art methods

Model	**Size**	**Tokens**	**ARC-e**	**ARC-c**	**Hella.**	**MMLU**	**Avg.**
Models trained from scratch (reference)							
RWKV6	7B	1.4T	73.6	44.0	75.2	43.9	58.0
Mamba	7B	1.2T	77.6	46.8	77.8	33.2	60.0
Llama2	7B	2T	76.4	46.2	76.0	45.5	60.2
Mistral	7B	?	80.8	54.0	81.1	62.4	66.6
Models via fine-tuning							
SUPRA	7B	+20B	74.6	42.3	74.8	28.0	-
RetNet	7B	+20B	73.3	39.9	72.9	26.1	51.9
GLA	7B	+20B	74.6	44.0	75.9	28.4	56.5
GSA	7B	+20B	75.9	43.9	76.5	32.4	57.7
MVA	7B	+20B	78.3	47.5	78.1	34.4	60.3
GPA (Ours)	7B	+8B	78.5	47.2	78.3	42.1	61.3

Table 2: Zero-shot and few-shot evaluation results on common language understanding benchmarks. ARC-c/e, HellaSwag, PIQA, and WinoGrande are evaluated in zero-shot setting, while MMLU uses 5-shot. AVG denotes the average accuracy across all tasks. A "+" token count denotes continued-training tokens.

Model	**ARC-c**	**ARC-e**	**Hella.**	**PIQA**	**Wino.**	**MMLU**	**AVG**
RWKV7-2.9B	48.7	81.0	76.4	79.7	72.8	55.0	68.93
GatedDeltaNet-1.3B	38.4	71.2	55.7	72.2	57.4	32.4	54.55
Mamba-2-2.7B	36.4	69.6	66.6	76.4	64.0	33.6	57.77
Qwen2.5-3B (Original)	45.0	77.4	73.5	78.6	68.5	65.7	68.12
Qwen2.5-3B-GPA (+10B)	45.2	78.0	71.4	78.2	66.8	39.2	63.13
Qwen2.5-14B (Original)	58.9	82.3	82.9	82.3	75.3	79.7	76.90
Qwen2.5-14B-GPA (+10B)	60.2	83.2	81.7	82.5	74.2	61.9	73.95
LLaMA-3-70B	64.1	86.9	81.6	84.8	80.6	82.0	80.00
LLaMA-3-70B-GSA (+12.5B)	47.6	72.6	74.9	64.5	59.4	42.1	60.18
LLaMA-3-70B-GPA (+10B)	65.2	87.8	79.8	85.6	78.9	58.4	75.95

4.1 GPA: Integrated Path-optimized Linear Attention

Building upon the validation of the individual principles, GPA combines all five components in a single recurrent block. Unless stated otherwise, it uses two memory levels with 64 states each (128 total), matching the state budget used for GSA and MVA. Table 18 evaluates decomposition depth and per-level state size. Relative to GSA, GPA adds explicit occurrence counts, a decoupled positional state, residual levels, and inter-level relations. Relative to MVA, it adds redundancy-aware updates, positional compression, theory-guided capacity selection, and a relation matrix that records how a residual at one level is represented at the next. Enhanced reading and sigmoid assignment are practical refinements rather than assumptions of Theorem 1.

Table 9 shows GPA-SWA improving the average over converted baselines across backbones; Table 10 reports a higher GPA-380M average with fewer training tokens. Table 20 shows task-dependent improvements from applying Principles 1–4 to SnapKV under a matched KV budget.

Table 3: (a) LongBench task scores: single-document QA (SQA), multi-document QA (MQA), summarization (Sum), few-shot learning (FS), and code completion (Code). (b) Passkey retrieval accuracy (fraction) at different context lengths.

(a) LongBench

Model	SQA	MQA	Sum	FS	Code	Avg.
Qwen2.5-7B	28.60	28.71	17.85	69.51	58.60	40.65
GSA-7B-Qwen (Ours)	22.72	22.05	15.68	67.32	55.34	36.62
GPA-7B-Qwen (Ours)	24.89	23.56	18.92	68.07	58.59	38.80
Qwen2.5-14B	36.25	33.85	20.12	74.60	63.42	45.65
GSA-14B-Qwen	31.02	30.74	16.34	70.60	63.01	42.34
GPA-14B-Qwen (Ours)	34.32	31.22	17.02	73.96	63.58	44.02

(b) Passkey retrieval

Model	256	512	1024	2048	4096	8192	16384	32768
Qwen	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GSA	1.0	0.8	0.7	0.5	0.3	0.4	0.1	0.1
GPA	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.8	0.8

Using the proposed method, we fine-tune pretrained model weights of varying scales and architectures, including Mistral-7B, Qwen2.5-3B, Qwen2.5-14B, and LLaMA3-70B, to obtain our GPA models. Since Mistral-7B was part of an earlier experiment, it was fine-tuned using the slimpajama dataset; the other models were fine-tuned using FineWeb-edu.

Fine-tuning Setup. We adopt the LoRA method for fine-tuning. LoRA is applied to all linear projection weights in the Transformer architecture. The rank of the low-rank adaptation matrices is set to 128. Training uses a constant learning rate of 1×10^{-4} with a warmup of 100 steps. Each training step processes a batch size of 0.5M tokens, and the total number of fine-tuning tokens is 10B.

Learned assignments and reproducibility. The deployed model uses learned, data-dependent Softmax-style or sigmoid assignments rather than a fixed similarity threshold; assignment activations are typically concentrated around 0.9. The fixed thresholds in Table 5 are controlled ablations only. Our end-to-end timing uses one NVIDIA A800 80GB GPU, 16 warm-up iterations, batch size 1, and Mistral-7B variants. The detailed kernel benchmark uses batch size 2, 16 query heads, 16 KV heads, head dimension 128, 32 layers, states $(64 + 64)$, and 128 decoding steps. Appendix A.4 reports the full configuration and component-wise latency.

Common-Sense Reasoning. Table 1 (Mistral-7B) and Table 2 (Qwen, LLaMA) show GPA recovers nearly 95% of original Transformer performance with only 10B tokens.

Long-Sequence Tasks. Tables 12 and 3 demonstrate GPA often surpasses the original attention models on long benchmarks, improving long-context modeling.

Retrieval Tasks. Table 3(b) shows GPA nearly perfectly solves passkey retrieval through 8K context after continued training, a capability previous converted linear-attention baselines lacked.

Efficiency Analysis. Figure 1 shows GPA's speed close to GSA at same state size, with much higher accuracy. Tables 21 and 22 provide decoding/prefill resource measurements and operation-wise latency profiles, respectively, for the reported attention benchmark.

Additional retrieval and long-context stress tests. On SWDE/SQuAD/FDA (Table 13), GPA-Qwen obtains 56.18/53.85/45.97, improving over GSA (45.76/41.92/19.26) and Gated DeltaNet (41.22/45.76/27.39). On S-NIAH-1 at 32K (Table 15), GPA reaches 91.2 versus 8.2 for GSA and 4.4 for Gated DeltaNet; the more difficult S-NIAH-2/3 tasks remain challenging for all purely linear models. Cross-layer hybridization closes much of this gap: retaining 12.5% Softmax layers raises S-NIAH-2 at 32K from 10.4 to 99.8 and MMLU from 56.7 to 63.6 (Table 16). With task-specific fine-tuning, GPA reaches 98–100 on BABILong Q1–Q5 at 16K (Table 17).

Compatibility with hybrid attention. GPA is complementary to both intra-layer token selection and cross-layer layer selection. Table 4 shows that combining GPA with STILL raises LLaMA-3-8B MMLU from 53.8 for GPA-SWA to 60.4 while retaining a 74.3 average without MMLU; KL-guided retention of a small number of Softmax layers yields the S-NIAH gains above. These results support GPA as a stronger linear component rather than claiming that a purely linear architecture always dominates a hybrid one.

Understanding the MMLU drop. Two effects are separable. First, continued pretraining itself shifts the model away from benchmark-specific capabilities: Qwen2.5-7B trained on FineWeb falls from 74.2 to 63.5 after 10B tokens and 60.9 after 20B, even without linearization; high-quality supervised data recovers part of the loss. Second, fixed-state approximation introduces an additional gap. Under a matched 20B-token FineWeb run, dense Qwen obtains 61.5, GPA 56.7, GSA 49.2, and Gated DeltaNet 45.4. GPA therefore does not eliminate the conversion gap, but reduces it substantially relative to alternative fixed-state conversions. Table 14 gives the complete controls.

Systems results. In the reported Triton attention benchmark, GPA decoding uses about 5 MiB of reported state and remains near 2.33 ms per benchmark step from 16K to 128K, whereas dense latency grows from 1.78 to 13.15 ms and reported KV cache from 258 to 2050 MiB. GPA becomes faster between 16K and 32K, but uses more prefill workspace. The dense cache figures are consistent with one attention layer rather than 32 layers; the measurements do not establish whole-model latency or memory.

Table 4: Compatibility with intra-layer hybrid attention on LLaMA-3-8B.

Model	Tokens (B)	PIQA	ARC-e	ARC-c	Hella.	Wino.	MMLU	Avg.	Avg. w/o MMLU
LLaMA-3-8B	–	79.5	80.0	53.3	79.1	73.1	65.3	71.8	73.0
LoLCATs	0.04	80.0	78.8	52.5	75.6	73.3	43.9	67.3	72.0
Liger-GLA [28]	0.02	80.3	81.1	52.5	76.3	72.0	43.4	67.6	72.4
STILL [29]	0.04	81.3	81.7	54.8	78.1	73.2	59.7	71.5	73.8
GPA-SWA (Ours)	0.04	81.2	**82.8**	54.5	**79.4**	**74.1**	53.8	71.0	**74.4**
GPA-STILL (Ours)	0.04	**81.3**	82.7	**55.2**	78.9	73.8	**60.4**	**72.0**	74.3

5 Conclusion

We chart the optimal path from Softmax to linear attention and verify, both theoretically and empirically, the pivotal roles of (i) redundancy removal, (ii) positional-code disentanglement & compression, (iii) tokenizer vocabulary reuse, (iv) layer-wise similarity, and (v) multi-vocabulary decomposition. Leveraging these insights, GPA sets a new efficiency-performance frontier: it matches or surpasses the best existing linearized models while consuming equal or fewer training tokens, offering a ready-to-use recipe for compressing large-language-model attention into a fixed-size, linear-complexity operator.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[2] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[3] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, et al. Mistral 7b, 2023.

[4] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

[9] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, et al. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

[10] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, et al. Video understanding with large language models: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

[11] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[12] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Trans. Softw. Eng. Methodol.*, July 2025. Just Accepted.

[13] Shi Luohe, Hongyi Zhang, Yao Yao, Zuchao Li, and hai zhao. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. In *First Conference on Language Modeling*, 2024.

[14] GAO WEI, Xinyu Zhou, Peng Sun, Tianwei Zhang, and Yonggang Wen. Rethinking key-value cache compression techniques for large language model serving. In *Eighth Conference on Machine Learning and Systems*, 2025.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 13–18 Jul 2020.

[16] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, et al. Simple linear attention language models balance the recall-throughput tradeoff, 2024.

[17] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56501–56523. PMLR, 2024.

[18] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

[19] Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, et al. Gated slot attention for efficient linear-time sequence modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 116870–116898, 2024.

[20] Ning Wang, Zekun Li, Tongxin Bai, Man Yao, Zhen Qin, and Guoqi Li. MVA: Linear attention with high-order query-keys integration and multi-level vocabulary decomposition. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 46473–46497. PMLR, 2025.

[21] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5961–5971, October 2023.

[22] Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Siyuan Pan, Pengfei Wan, et al. Agent attention: On the integration of softmax and linear attention. In *European Conference on Computer Vision*, 2024.

[23] Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. Polaformer: Polarity-aware linear attention for vision transformers, 2025.

[24] Weikang Meng, Yadan Luo, Liangyu Huo, Yingjian Li, Yaowei Wang, Xin Li, et al. Norm$\times$direction: Restoring the missing query norm in vision linear attention, 2025.

[25] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, et al. Demystify mamba in vision: A linear attention perspective. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[26] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, et al. Linearizing large language models, 2024.

[27] Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, et al. Lolcats: On low-rank linearizing of large language models, 2025.

[28] Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 32452–32466. PMLR, 2025.

[29] Weikang Meng, Liangyu Huo, Yadan Luo, Jiawen Guan, Jingyi Zhang, Yingjian Li, et al. STILL: Selecting tokens for intra-layer hybrid attention to linearize LLMs, 2026.

[30] Yanhong Li, Songlin Yang, Shawn Tan, Mayank Mishra, Rameswar Panda, Jiawei Zhou, et al. Distilling to hybrid attention models via KL-guided layer selection. In *International Conference on Learning Representations*, 2026.

[31] Yuxian Gu, Qinghao Hu, Shang Yang, Haocheng Xi, Junyu Chen, Song Han, et al. Jet-nemotron: Efficient language model with post neural architecture search, 2025.

[32] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, et al. A framework for few-shot language model evaluation, 12 2023.

[33] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, et al. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack, 2024.

[34] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, et al. Lora: Low-rank adaptation of large language models, 2021.

[35] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, et al. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 22947–22970, Red Hook, NY, USA, 2024. Curran Associates Inc.

[36] Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, et al. Deepseek-v3 technical report, 2025.

[38] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, et al. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[39] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc., 2023.

[40] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 21875–21895, 2024.

[41] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[42] Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Ruijie Zhu, Yiran Zhong, et al. Metala: Unified optimal linear approximation to softmax attention map, 2024.

[43] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, et al. Hgrn2: Gated linear rnns with state expansion, 2024.

[44] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, et al. Learning to (learn at test time): Rnns with expressive hidden states, 2025.

[45] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024.

[46] Kumar Krishna Agrawal, Long Lian, Longchao Liu, Natalia Harguindeguy, Boyi Li, Alexander Bick, et al. Atlas: Multi-scale attention improves long context image modeling, 2025.

[47] Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories, 2025.

[48] Nick Alonso, Tomas Figliolia, and Beren Millidge. Online vector quantized attention, 2026.

[49] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational Linguistics, 2018.

A Appendix

Acknowledgments

We thank the anonymous reviewers for their insightful comments and constructive suggestions, and the Area Chair for their careful assessment and efforts throughout the review process.

This work was supported by the Brain Science and Brain-like Intelligence Technology – National Science and Technology Major Project (2025ZD0215500) and the National Natural Science Foundation of China (62406322). We also thank FlagOS for its support.

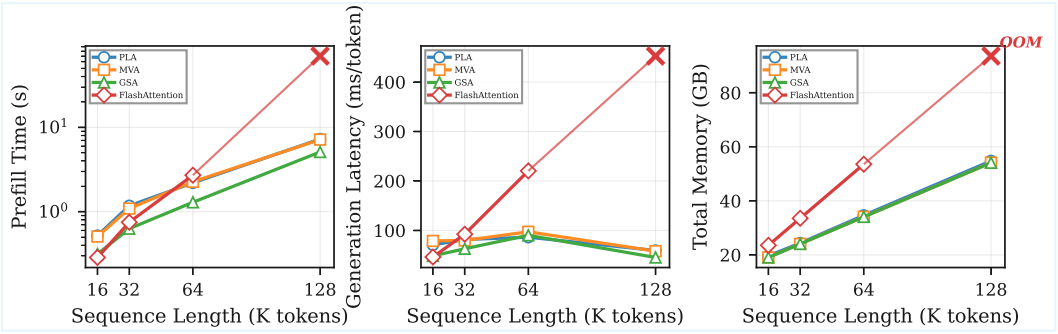

Figure 1: Inference efficiency comparison of GPA, MVA, GSA, and FlashAttention under different sequence lengths. The legacy legend *PLA* denotes GPA. OOM indicates out-of-memory errors.

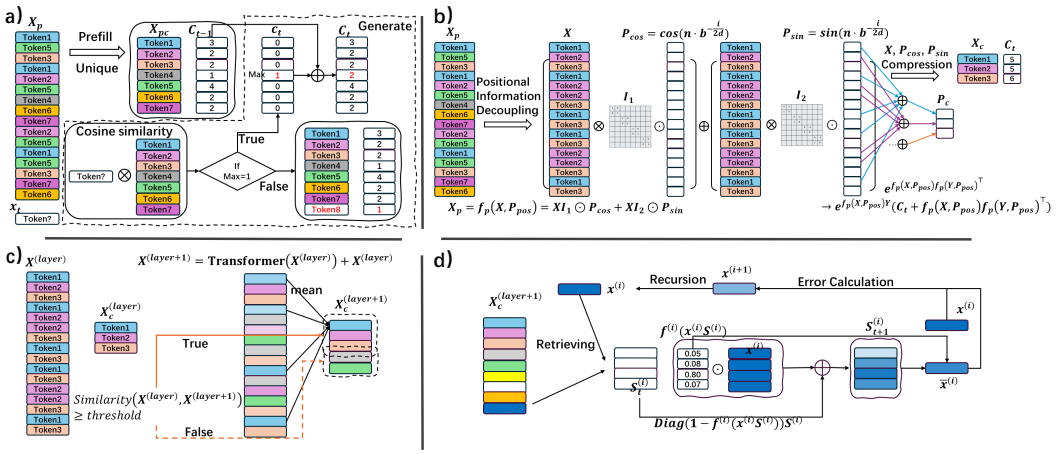

Figure 2: Conceptual illustration of the GPA memory hierarchy and its theoretical foundations. Panels (a)–(d) show redundancy elimination, positional decoupling and compression, inter-layer similarity, and multi-level state decomposition.

This figure illustrates the key components that enable bounded state size while preserving representational capacity, grounded in the theoretical analysis of Section 3.

(a) Redundancy elimination. During sequence growth, a large number of repeated tokens appear. The unbounded growth of the KV cache can be replaced by storing an identical token and recording its occurrence count. When the cosine similarity between tokens is greater than or equal to a threshold (e.g., 1 or 0.95), the two tokens are considered identical; the count is incremented. Otherwise, the new token is appended to the KV cache. This mechanism transforms the unbounded KV cache into a KV state with a bounded storage capacity.

(b) Position decoupling and position compression. After decoupling positional information from the sequence, the number of token types is significantly reduced. For instance, after decoupling positional information at the first layer, the number of token types equals the vocabulary size of the tokenizer. The positional information is then linearly compressed and used to compute attention scores (approximating a first-order Taylor expansion).

(c) Inter-layer similarity. Although the number of KV token types in subsequent layers increases substantially due to token mixing operations, it remains bounded by the previous layer's capacity due to the residual network structure. Consequently, the upper bound on the KV cache in subsequent layers can be controlled via the first-layer tokenizer vocabulary size. Specifically, if the cosine similarity between a token in a later layer and its corresponding token in the previous layer is below a threshold, it is compressed into a fixed state; otherwise, it is stored using a state size equal to that of the previous layer.

(d) Multi-level state decomposition. The upper bound imposed by the tokenizer vocabulary size offers limited advantage compared to current task-specific KV cache compression methods. We therefore approximate this bound using a multi-level decomposition. For example, a vocabulary size of $128K$ can be approximated by a three-level linear state with size 64 per level (since $64^3 > 128K$). The input feature $x^{(i)}$ is distributed across the states $S^{(i)}$ via retrieval scores; the information of $x^{(i)}$ in $S^{(i)}$ is then retrieved and reconstructed as $\bar{x}^{(i)}$. The residual error between $x^{(i)}$ and $\bar{x}^{(i)}$ is computed as $x^{(i+1)}$ and stored recursively.

A.1 Background Formulations

A.1.1 KV Cache Compression Methods

To alleviate KV cache growth, compression methods reduce redundancy via low-rank transformations or selection along the sequence or channel dimension.

Formally, given $\boldsymbol{X} \in \mathbb{R}^{t\times d}$ with projections $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{t\times d}$, the compressed cache $(\boldsymbol{K}^c, \boldsymbol{V}^c)$ of size $c \times d^r$ is

$$\boldsymbol{K}^c = \varphi\big(\phi(\boldsymbol{R}\boldsymbol{K})\boldsymbol{L}\big), \qquad \boldsymbol{V}^c = \varphi\big(\phi(\boldsymbol{R}\boldsymbol{V})\boldsymbol{L}\big), \tag{28}$$

where $\boldsymbol{R} \in \mathbb{R}^{c\times t}$ selects c tokens, $\boldsymbol{L} \in \mathbb{R}^{d\times d^r}$ compresses channels ($d^r = h \times d_k^r$ with h heads), and $\phi(\cdot)$, $\varphi(\cdot)$ are transformation and selection functions. Since t grows unboundedly, $\boldsymbol{R}$ is constructed recursively:

$$\boldsymbol{R} = f(\boldsymbol{R}'\boldsymbol{X}^\top) \in \mathbb{R}^{c\times t}, \tag{29}$$

where $\boldsymbol{R}' \in \mathbb{R}^{c\times d}$ defines a local window and $f(\cdot)$ specifies the selection strategy.

(1) **SnapKV** [35] compresses only along the sequence dimension: $\boldsymbol{L} = \boldsymbol{I}$, and $\boldsymbol{R} = f(\boldsymbol{Q}\boldsymbol{K}^\top)$ applies `top-`c over recent queries to select KV blocks, with $\phi(\cdot)$ gathering tokens by these indices. (2) **HeadKV** [36] extends SnapKV to channel compression: $\boldsymbol{R}$ selects tokens as above, while $\boldsymbol{L}$ is defined by an additional projection $\boldsymbol{L}'$ voting across heads, and $\varphi(\cdot)$ gathers heads by these votes. (3) **Multi-Head Latent Attention (MLA)** [37] removes token selection: $\phi(\cdot)$ and $\varphi(\cdot)$ are identity, $\boldsymbol{R} = \boldsymbol{I}$, and channel compression uses fixed $\boldsymbol{L} = (\boldsymbol{W}^{UK})^{-1}$. Leveraging associativity to fuse up-projection with the query-key product, MLA avoids explicit reconstruction of $\boldsymbol{K}$ and $\boldsymbol{Q}$, achieving memory savings and speedup.

Subsequent approaches including InfLLM [38], HO2 [39], and StreamLLM [40] are hybrids or reformulations of these principles.

A.1.2 Linear Models

Linear attention replaces the Softmax kernel with linearizable feature maps, reordering computations to achieve linear time and fixed-size state:

Parallel form.

$$\boldsymbol{O} = \mathrm{LA}\big(\phi(\boldsymbol{Q}), \phi(\boldsymbol{K}), \boldsymbol{V}\big) = \big(\big(\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top\big) \odot \boldsymbol{M}^l\big)\,\boldsymbol{V}, \tag{30}$$

Recursive form.

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \phi(\boldsymbol{k}_t)^\top \boldsymbol{v}_t, \qquad \boldsymbol{o}_t = \phi(\boldsymbol{q}_t)\,\boldsymbol{S}_t, \tag{31}$$

where $\boldsymbol{M}^l_{ij} = 1$ if $i \geq j$ and 0 elsewhere and $\boldsymbol{S}_t \in \mathbb{R}^{d_k \times d_v}$ is a fixed-size state. Many works improve $\phi(\cdot)$ [21, 41] or add auxiliary mechanisms.

However, this formulation suffers from state saturation. To address this, GLA introduces gating for dynamic forgetting, biasing toward recent context.

Gating / GLA. Gated Linear Attention (GLA) [17] applies multiplicative gates:

$$\boldsymbol{O} = \mathrm{GLA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{G}) = \mathrm{LA}(\boldsymbol{Q} \odot \boldsymbol{B},\ \boldsymbol{K}/\boldsymbol{B},\ \boldsymbol{V}), \boldsymbol{S}_t = \mathrm{diag}(\boldsymbol{g}_t)\, \boldsymbol{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t, \boldsymbol{o}_t = \boldsymbol{q}_t\, \boldsymbol{S}_t. \tag{32}$$

where the t-th row of $\boldsymbol{B}$ is $\boldsymbol{b}_t = \prod_{i=1}^t \boldsymbol{g}_i$ and $\boldsymbol{G} = \sigma(\boldsymbol{X}\boldsymbol{W}_g) \in \mathbb{R}^{n \times d_k}$. Models like MetaLA [42], HGRN2 [43], and GSA [19] also employ gating, but this offers coarse control over $\boldsymbol{S}_t$, motivating fast-weight updates.

Fast-weight / Delta Rule-style updates. These correct stored representations via prediction error:

$$\boldsymbol{v}_t^{\text{old}} = \boldsymbol{k}_t\, \boldsymbol{S}_{t-1}, \qquad \boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{g}_t \cdot \boldsymbol{k}_t^\top \big(\boldsymbol{v}_t - \boldsymbol{v}_t^{\text{old}}\big), \qquad \boldsymbol{o}_t = \boldsymbol{q}_t\, \boldsymbol{S}_t. \tag{33}$$

The update adds a correction proportional to the discrepancy between observed $\boldsymbol{v}_t$ and predicted $\boldsymbol{v}_t^{\text{old}}$, gated by $\boldsymbol{g}_t$. DeltaNet [18] parallelized this into a powerful linear model.

Optimization viewpoint. Viewing updates through an optimization lens yields dynamic weighting methods like TTT [44], Titans [45], and Atlas [46], sharing the objective:

$$\mathcal{L}_t(\boldsymbol{M}) \ = \ \sum_{i=1}^{t} \gamma_i \left\| \boldsymbol{M}\big(\phi(\boldsymbol{k}_i)\big) - \boldsymbol{v}_i \right\|_2^2, \tag{34}$$

where $\boldsymbol{M}(\cdot)$ maps key-features to value-predictions and $\{\gamma_i\}$ are weights. The memory $\boldsymbol{M}_t$ is updated via iterative optimization:

$$\boldsymbol{M}_t = \alpha_t\, \boldsymbol{M}_{t-1} + F(\boldsymbol{S}_t), \quad \boldsymbol{S}_t = \eta_t\, \boldsymbol{S}_{t-1} - \theta_t\, \nabla_{\boldsymbol{S}}\, \mathcal{L}_t\big(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t\big), \tag{35}$$

where $F(\cdot)$ aggregates statistics into primary memory $\boldsymbol{M}_t$, and the second line denotes a gradient-based corrective step for working state $\boldsymbol{S}_t$. This explains empirically successful rules and motivates algorithms minimizing per-step predictive error.

Extending the state budget. Even sophisticated updates may leave a single low-rank state insufficient for multi-scale dependencies. Recent works maintain multiple parallel or hierarchical states. **MoM** [47] and **MVA** [20] maintain multiple memory banks (e.g., short-term vs. long-term) and decompose the state into interacting sub-states:

$$\boldsymbol{S}_t^{(i)} = \mathrm{diag}(1 - \bar{f}^{(i)}(\boldsymbol{x}_t^{(i)})^\top)\boldsymbol{S}_{t-1}^{(i)} + \bar{f}^{(i)}(\boldsymbol{x}_t^{(i)})^\top \boldsymbol{x}_t^{(i)}, \qquad \boldsymbol{x}_t^{(i+1)} = f_1^{(i)}(\boldsymbol{x}_t^{(i)}, \boldsymbol{S}_t^{(i)}) \tag{36}$$

where f is typically σ and f_1 is a hybrid expert or delta function. This multi-state design narrows the gap to Softmax attention at the cost of additional complexity.

A.2 Expanded Related Work

Linear attention and expressiveness. Kernelized linear attention factorizes query–key similarity and reassociates $(\phi(Q)\phi(K)^\top)V$ as $\phi(Q)(\phi(K)^\top V)$, avoiding an explicit quadratic attention matrix [15, 41]. Its practical quality depends on more than reassociation. FLatten Transformer strengthens concentration and restores feature rank [21]; PolaFormer models same-sign and opposite-sign interactions [23]; NaLaFormer restores the query-norm signal that controls attention sharpness [24]; Agent Attention introduces a small set of agent tokens connecting Softmax and generalized linear attention [22]; and Mamba-inspired linear attention identifies forgetting and block design as central ingredients [25]. These studies improve the readout kernel or block. GPA instead treats the complete conversion as an approximation of the original KV cache and composes storage, position, capacity, and readout errors.

Recurrent and multi-state models. GLA uses data-dependent forget gates, while DeltaNet and Gated DeltaNet use prediction-error updates [17, 18]. MetaLA, HGRN2, and GSA improve recurrent expressiveness through Softmax approximation, hierarchical gating, or slot-based memory [42, 43, 19]. TTT and Titans interpret memory updates as online optimization [44, 45], while MoM and MVA expand capacity through multiple memories or residual vocabularies [47, 20]. GPA is related to vector quantization and Delta-Rule memories, but differs in explicitly retaining occurrence counts, positional states, theory-guided level capacity, and inter-level residual relations.

Linearizing and hybridizing pretrained LLMs. SUPRA, LoLCATs, and Liger reduce the cost of converting pretrained Softmax models through uptraining, teacher alignment, or reuse of key projections [26, 27, 28]. STILL preserves sparse Softmax attention for salient tokens inside each layer [29]. KL-guided layer selection and Jet-Nemotron instead retain or search selected full-attention layers [30, 31]. These approaches are complementary to GPA: the GPA state can replace their linear component, while their selected Softmax tokens or layers preserve high-fidelity access where a fixed state remains insufficient.

A.3 Experimental Validation of Theoretical Principles

We first conduct experiments with different parameters for each theoretical principle. The evaluation uses the passkey retrieval task, standard benchmarks from lm-evaluation-harness (ARC-C and MMLU), and the long-sequence SAMSUM dataset from LongBench for testing and guidance.

Theory 1: Redundancy Elimination with Similarity Thresholds.

Table 5: Results for Theory 1 with different similarity thresholds

Method	Fine-tuning tokens	Passkey (1K-8K)	ARC	MMLU	SAMSum
Mistral-7B-v0.1	–	100.0	54.0	62.4	43.6
Theory 1 ($t = 1$)	200M	100.0	54.0	62.4	43.6
Theory 1 ($t = 0.95$)	200M	100.0	53.4	60.7	42.9
Theory 1 ($t = 0.9$)	200M	100.0	51.8	57.6	41.5
Theory 1 ($t = 0.8$)	200M	100.0	49.7	49.2	38.3

Table 5 shows experiments with different similarity thresholds for Theory 1, where $t = 1$ indicates the threshold used for similarity discrimination in Theory 1 (e.g., $t = 0.9$ means tokens with cosine similarity ≥ 0.9 are considered identical). Results demonstrate that when similarity exceeds a certain level (e.g., 0.95), performance approaches that of the original model. This approach functions as a quantization process, indicating model insensitivity to token variations within certain ranges.

Theories 2 & 3: Positional Information Decoupling and Compression.

Table 6 presents experiments for Theories 2 and 3, exploring positional information separation, compression, and refined Taylor expansion approaches. Here, "depos" indicates decoupled positional encoding, while "de&cprpos" indicates decoupled and compressed positional encoding.

Table 6: Results for Theories 2 & 3 with different positional encoding strategies

Method	Fine-tuning tokens	Passkey (1K-8K)	ARC	MMLU	SAMSum
Theory 2&3 ($t = 0.95$, w/ depos)	500M	100.0	52.8	58.6	43.5
Theory 2&3 ($t = 0.95$, w/ de&cprpos)	500M	100.0	50.2	53.1	40.7
Theory 2&3 ($t = 0.95$, w/ de&cprpos-2)	500M	100.0	51.9	55.7	42.7

Theory 4: Inter-layer Scaling Factors.

Table 7 shows experiments for Theory 4 with different layer-wise scaling factors. Using parameters from previous theories ($t = 0.95$, w/ depos), $c_{\text{scale}} = 1.2$ indicates that the KV cache size at layer $l + 1$ is 1.2 times that of layer l, up to the midpoint of the total layers, after which the KV cache size remains constant. $c_{\text{scale-l8}} = 1.2$ & $c_{\text{scale-l16}} = 1.6$ indicates a scaling factor of 1.2 for the first 8 layers and 1.6 for layers 8-16.

Figure 3 illustrates the evolution of KV cache length across layers as predicted by Theory 4.

Table 7: Results for Theory 4 with different layer scaling factors

Method	Fine-tuning tokens	Passkey (1K-8K)	ARC	MMLU	SAMSum
Theory 4 (w/ $c_{\text{scale}} = 1.2$)	500M	100.0	46.2	52.0	40.8
Theory 4 (w/ $c_{\text{scale}} = 1.4$)	500M	100.0	51.7	57.8	43.2
Theory 4 (w/ $c_{\text{scale}} = 1.6$)	500M	100.0	53.4	60.1	42.7
Theory 4 (w/ $c_{\text{l8}} = 1.2$ & $c_{\text{l16}} = 1.6$)	500M	100.0	53.3	59.7	42.9

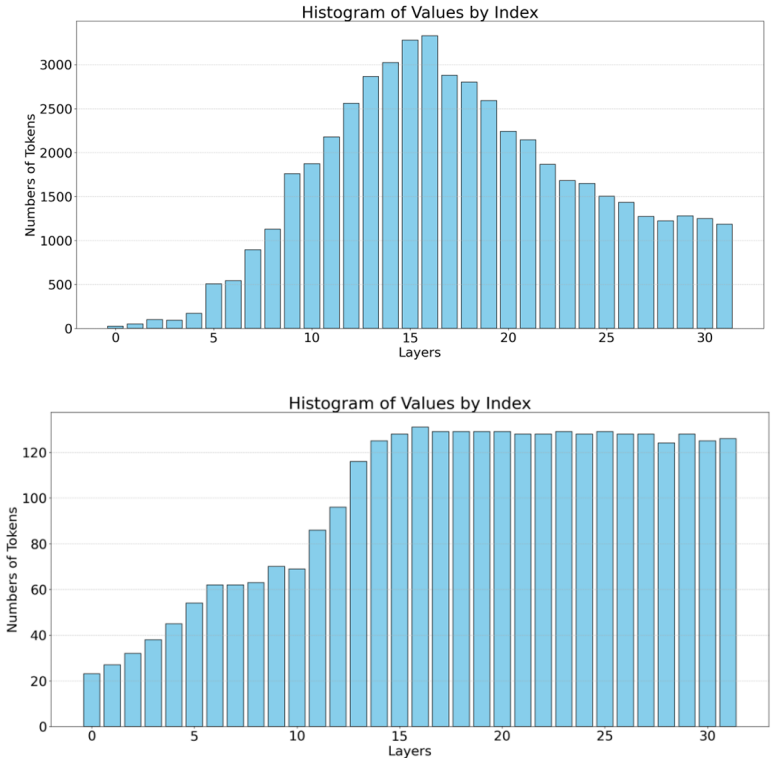

Figure 3: Per-layer KV state counts on the passkey task. The first panel applies redundancy consolidation (Theory 1); the second additionally applies inter-layer similarity (Theory 4), which controls the growth of later-layer states.

Theory 5: Enhanced Reading Mechanisms and Multi-state Configurations.

Table 8 presents experiments for Theory 5, examining various enhanced reading mechanisms and different state sizes. We use single and two-level vocabulary configurations, without positional separation for faster convergence, focusing solely on reading mechanism variations. Results show progressive performance improvement with enhanced reading capabilities, with our GPA approach building upon GSA by adding multi-channel reading and multi-state interaction, equivalent to incorporating vocabulary interaction(VI) into MVA.

Table 8: Results for Theory 5 with different reading mechanisms

Method	Fine-tuning tokens	Passkey (2K)	ARC	MMLU	SAMSum
Theory 5 (GSA)	500M	0.0	31.7	22.3	18.9
Theory 5 (GSA w/ sigmoid)	500M	0.0	33.5	23.5	18.9
Theory 5 (GSA + MetaLA)	500M	10.0	35.6	24.1	21.7
Theory 5 (MVA)	500M	20.0	38.2	25.6	24.9
Theory 5 (GPA: MVA + VI)	500M	40.0	39.4	26.2	24.7

Table 9: Performance comparison of different models across various benchmarks.

Model	Tokens (B)	PIQA	ARC-e	ARC-c (norm)	Hella. (norm)	Wino.	MMLU (5-shot)	Avg.	Avg. w/o MMLU
Mistral 7B	–	82.1	80.9	53.8	81.0	74.0	62.4	72.4	74.4
Mistral 7B SUPRA	100	80.4	75.9	45.8	77.1	70.3	34.2	64.0	69.9
Mistral 7B LoLCATs [27]	**0.04**	81.5	**81.7**	54.9	80.7	74.0	51.4	70.7	74.5
Mistral 7B GPA-SWA (Ours)	**0.04**	**82.1**	81.5	**55.4**	**81.1**	**74.2**	**54.2**	**71.4**	**74.9**
Llama 3 8B	–	79.9	80.1	53.3	79.1	73.1	66.6	72.0	73.1
Llama 3 8B Hedgehog	**0.04**	77.4	71.1	40.6	66.5	54.3	24.2	55.7	62.0
Llama 3 8B LoLCATs	**0.04**	80.9	81.7	**54.9**	79.7	**74.1**	52.8	70.7	74.2
Llama 3 8B GPA-SWA (Ours)	**0.04**	81.2	**82.8**	54.5	79.4	74.1	53.8	**71.0**	**74.4**

Table 10: Performance comparison of different attention variants across benchmarks.

Model	Tokens (B)	Params	PIQA	Hella.	Wino.	ARC-e	ARC-c	Avg.
std att	50	480M	66.4 ± 0.1	40.5 ± 0.1	52.3 ± 0.7	52.8 ± 0.4	29.1 ± 0.5	48.21
sw-nope	50	480M	66.7 ± 0.4	40.9 ± 0.1	52.6 ± 0.6	53.1 ± 0.1	28.4 ± 0.4	48.35
sw-ovq [48]	50	480M	66.6 ± 0.2	41.1 ± 0.2	52.4 ± 0.2	52.7 ± 0.1	28.7 ± 0.4	48.30
GPA-380M (Ours)	**25**	**380M**	**67.6**	40.1	**52.6**	**57.2**	28.6	**49.22**

Reported $\pm$ values are retained for the reference baselines; no uncertainty estimate is supplied for GPA. The uncertainty type is not specified in the available comparison.

Table 11: Summary of Theoretical Principles for Softmax-to-Linear Attention Compression

Theory	Mechanism	Function	Error Bound	Equivalent Mechanism
T1	Redundancy Elimination/ Quantization	Controls state upper bound	Lossless for exact matches	Delta Rule, Quantization
T2	Positional Information Decoupling	Reduces first-layer state count	Lossless	Tokenizer vocabulary + positional decoupling
T3	Positional Information Compression	Controls positional dimension error	$O\left(\frac{n}{\text{base}^{4j/d}}\right)$	Gating mechanism
T4	Inter-layer Similarity	Controls intermediate layer state count	Controllable (empirically validated)	Residual connection + similarity propagation
T5	Multi-level State Decomposition	Fixed-size state representation	Exponential decay	Vocabulary decomposition + enhanced reading

Table 12: Long-context evaluation and training-efficiency comparison.

(a) Long-context evaluation

Model	Qasper	NarrativeQA	QMSum
Models trained from scratch			
RWKV6	9.2	14.4	1.1
Mamba	5.6	27.9	0.8
Mistral	25.8	25.1	5.0
Fine-tuned from Mistral-7B (10B tokens)			
RetNet	11.1	0.0	0.0
GLA	18.4	17.2	9.0
GSA	18.8	19.2	10.0
MVA	20.7	20.4	9.58
GPA	22.3	21.2	10.7

(b) Training resources

Method	Memory (MiB)	Time (s/iter.)
MetaLA	36,317	75.08
GSA	37,619	81.67
MVA w/ VD	40,096	105.79
GPA w/ VD	41,278	118.67

A.4 Additional Retrieval, Hybrid, and Systems Experiments

Additional retrieval benchmarks. Following the reviewer suggestion, we evaluate converted Qwen2.5-7B models on SWDE, SQuAD [49], and FDA. GPA improves consistently over the two fixed-state baselines, although a gap to the dense model remains.

Table 13: Retrieval accuracy on additional benchmarks.

Model	SWDE	SQuAD	FDA
Qwen2.5-7B	92.80	69.09	81.20
GSA-Qwen-7B	45.76	41.92	19.26
Gated DeltaNet-Qwen-7B	41.22	45.76	27.39
GPA-Qwen-7B (Ours)	**56.18**	**53.85**	**45.97**

MMLU controls. Table 14 separates continued-training drift from the architecture-conversion gap. The two FineWeb-20B dense values, 60.9 and 61.5, were reported in separate rebuttal comparisons; the supplied records do not establish the reason for their difference. Continued pretraining causes a substantial drop even in the dense model, while GPA retains more of the matched dense result than GSA or Gated DeltaNet. High-quality supervised tuning recovers part, but not all, of the lost accuracy.

Table 14: MMLU analysis. "CPT" denotes continued pretraining and "SFT" supervised fine-tuning.

Model	Data	Added tokens	MMLU
Qwen2.5-7B	–	–	74.2
Qwen2.5-7B-SFT	unspecified SFT	1B	72.6
Qwen2.5-7B-CPT	FineWeb	10B	63.5
Qwen2.5-7B-CPT	FineWeb	20B	60.9
Qwen2.5-7B-CPT+SFT	Tulu-3 + NP-SFT-v1	2B	64.6
Qwen2.5-7B-CPT	SlimPajama	10B	52.9
Qwen2.5-7B-CPT (matched run)	FineWeb	20B	61.5
GPA-Qwen-7B-CPT	FineWeb	20B	**56.7**
GSA-Qwen-7B-CPT	FineWeb	20B	49.2
Gated DeltaNet-Qwen-7B-CPT	FineWeb	20B	45.4
Mistral-7B	–	–	62.4
Mistral-7B-CPT	SlimPajama	10B	44.5
GPA-Mistral-7B-CPT	SlimPajama	10B	**42.6**
GSA-Mistral-7B-CPT	SlimPajama	100B	37.8

Intra-layer hybridization. Table 4 compares token-level hybrid attention. GPA-STILL combines the GPA linear state with STILL's salient-token Softmax branch and produces the strongest MMLU and overall average among converted models. This supports complementarity rather than treating hybrid attention as a competing formulation.

S-NIAH and cross-layer hybridization. The purely linear GPA model strongly improves S-NIAH-1 but still loses accuracy on the multi-key variants. Following KL-guided layer selection [30], retaining 12.5% of Softmax layers recovers nearly all S-NIAH-2 accuracy while improving MMLU.

Table 15: S-NIAH stress tests. Converted models use 20B fine-tuning tokens in this comparison. The separate 12.5% hybrid comparison is given in Table 16.

(a) S-NIAH-1

Model	2K	8K	32K	128K	256K
Qwen2.5-7B	100	100	100	46.5	0.0
GSA-Qwen-7B	98.8	35.6	8.2	1.6	0.6
Gated DeltaNet-Qwen-7B	97.2	31.8	4.4	0.8	1.4
GPA-Qwen-7B (Ours)	**100**	**98.8**	**91.2**	**90.0**	**88.6**

(b) S-NIAH-2 and S-NIAH-3

Model	S-NIAH-2			S-NIAH-3		
	2K	8K	32K	2K	8K	32K
Qwen2.5-7B	100	100	100	97.8	99.2	100
GSA-Qwen-7B	24.4	16.2	1.8	0.4	0.0	0.0
Gated DeltaNet-Qwen-7B	37.2	11.8	0.4	1.2	0.0	0.0
GPA-Qwen-7B (Ours)	**68.0**	**32.6**	**10.4**	**25.4**	**10.8**	**5.4**

Table 16: Cross-layer hybrid GPA on difficult S-NIAH settings and MMLU.

Model	S-NIAH-2			S-NIAH-3			MMLU
	2K	8K	32K	2K	8K	32K	
Qwen2.5-7B	100	99.4	99.2	97.8	99.2	100	74.2
GPA-Qwen-7B	68.0	32.6	10.4	25.4	10.8	5.4	56.7
GPA-Qwen-7B-12.5%	**100**	**100**	**99.8**	**99.0**	**96.8**	**89.6**	**63.6**

BABILong. Without task-specific training, fixed-state models remain below dense attention on multi-hop reasoning. Following the BABILong/RMT protocol, however, GPA recovers 98–100 accuracy across Q1–Q5 and matches or exceeds the fine-tuned dense backbones.

Table 17: BABILong accuracy at 16K context. Fine-tuned rows follow the task-specific protocol.

Model	Q1	Q2	Q3	Q4	Q5
Qwen2.5-7B	70	43	73	74	73
GSA-Qwen-7B	21	15	18	17	32
Gated DeltaNet-Qwen-7B	19	11	22	14	28
GPA-Qwen-7B	**35**	**27**	**31**	**32**	**54**
Qwen2.5-7B-finetune	100	72	86	96	94
GPA-Qwen-7B-finetune	100	**100**	**98**	**100**	**98**
Mistral-7B-finetune	100	49	36	56	74
GPA-Mistral-7B-finetune	100	**100**	**96**	**100**	**97**

State-size and depth sensitivity. All converted Qwen2.5-3B variants below use 3B tokens sampled from FineWeb. Increasing either depth or per-level state size improves some tasks, but no single configuration dominates, exposing a real capacity–efficiency trade-off rather than an invariant optimum.

Table 18: Sensitivity to decomposition depth and state size.

Model	ARC-c	ARC-e	HellaSwag	PIQA	WinoGrande
Qwen2.5-3B	45.0	77.4	73.5	78.6	68.5
GPA (2×64)	41.5	74.6	65.1	72.3	60.1
GPA (4×64)	**42.6**	74.0	65.7	**74.2**	**62.5**
GPA (2×128)	42.1	**75.9**	**66.5**	73.9	61.6

Inter-layer similarity across scales. Table 19 reports representative adjacent-layer cosine similarities averaged over multiple randomly sampled texts. Except for the first transition, similarity is high across architectures and generally increases with depth in LLaMA-3-70B. This evidence supports using p_l as a measurable capacity signal, but does not turn the conditional bound into a universal constant.

Table 19: Representative adjacent-layer cosine similarity. A dash denotes a nonexistent layer pair.

Layer pair	LLaMA-3-70B	Qwen2.5-3B	LLaMA-3.1-8B
L0–L1	0.7310	0.4411	0.5933
L1–L2	0.8908	0.9043	0.7600
L7–L8	0.9209	0.9302	0.8528
L15–L16	0.9152	0.9415	0.8976
L23–L24	0.9107	0.9601	0.9477
L30–L31	0.9253	0.9289	0.9367
L34–L35	0.9239	0.9590	–
L47–L48	0.9772	–	–
L55–L56	0.9809	–	–
L63–L64	0.9811	–	–
L71–L72	0.9712	–	–
L78–L79	0.8523	–	–

Matched-memory KV compression. Under an equivalent 1,024-token storage budget, adding GPA Principles 1–4 to SnapKV [35] raises RepoBench-P (RBP) from 51.87 to 52.67 and improves Qasper, HotpotQA, 2WikiMQA, and GovReport. This shows that the principles can also improve a Softmax-preserving cache compressor.

Table 20: LongBench under an equivalent 1,024-token KV budget.

(a) NarrativeQA through GovReport

Method	NtrvQA	Qasper	HotpotQA	2WikiMQA	Musique	GovReport
Mistral (all KV)	26.82	33.06	42.77	27.33	19.27	32.85
SnapKV	25.54	29.51	40.94	25.70	19.42	25.89
SnapKV + GPA P1–P4	25.67	**31.85**	**41.92**	**26.71**	18.98	**29.35**

(b) QMSum through RepoBench-P

Method	QMSum	MultiNews	TREC	SAMSum	LCC	RBP
Mistral (all KV)	24.25	27.06	71.0	42.98	55.51	52.88
SnapKV	23.82	26.11	69.5	42.06	55.65	51.87
SnapKV + GPA P1–P4	**24.12**	26.11	69.0	**42.64**	55.58	**52.67**

Reported attention-system efficiency. The reported dense cache grows with context, while GPA keeps a fixed 5 MiB state. GPA is slower at 16K but becomes faster from 32K onward in this benchmark. The table also reports more prefill workspace for GPA; peak workspace and persistent decoding state are distinct resource costs.

Table 21: Reported attention benchmark on one NVIDIA A800 80GB. Batch size 2, 16 query/KV heads, head dimension 128, GPA states $(64+64)$, and 128 decoding steps. The supplied configuration lists 32 layers, but the dense cache figures are consistent with one layer; whole-model resource use is not established.

(a) Decoding and reported state/cache

Method	Context	GPU latency (ms)	Tokens/s	Reported state/cache (MiB)	Decode workspace (MiB)
Dense	16K	1.7782	1121.89	258	260.13
GPA	16K	2.3432	849.43	5	5.60
Dense	32K	3.3608	594.37	514	516.13
GPA	32K	2.3270	855.27	5	5.60
Dense	64K	6.6200	301.93	1026	1028.13
GPA	64K	2.3285	854.69	5	5.60
Dense	128K	13.1532	152.00	2050	2052.13
GPA	128K	2.3404	850.38	5	5.60

(b) Prefill

Method	Context	Prefill GPU (ms)	Prefill workspace (MiB)
Dense	16K	18.3831	640
GPA	16K	24.0957	2048
Dense	32K	51.3480	1280
GPA	32K	50.6516	4096
Dense	64K	187.4776	2560
GPA	64K	99.1206	8192
Dense	128K	735.0504	5120
GPA	128K	202.1429	16384

Table 22: GPA latency breakdown. Percentages are relative to end-to-end latency.

	Generation				Prefill			
Stage	32K ms	%	64K ms	%	32K ms	%	64K ms	%
Q/K/V projection	0.1089	4.69	0.1099	4.76	6.8295	13.61	12.9315	12.97
L0 codebook affinity	0.1655	7.12	0.1633	7.07	2.4099	4.80	4.9484	4.96
L0 chunk query (S^K)	0.2548	10.96	0.2508	10.86	7.2576	14.47	14.0173	14.06
L0 score projection	0.0810	3.49	0.0808	3.50	0.3865	0.77	0.6825	0.68
L0 chunk query (S^V)	0.2433	10.47	0.2409	10.43	6.2333	12.43	12.6779	12.72
Residual K/V projection	0.1285	5.53	0.1301	5.63	1.4033	2.80	2.8033	2.81
L1 codebook affinity	0.1767	7.61	0.1754	7.59	2.4045	4.79	4.9730	4.99
L1 chunk query (S^K)	0.2554	10.99	0.2569	11.12	6.7719	13.50	13.7106	13.75
L1 score projection	0.0808	3.48	0.0810	3.51	0.3441	0.69	0.6785	0.68
State0–1 relation	0.2402	10.34	0.2358	10.21	6.7733	13.50	13.6295	13.67
L1 relation merge	0.0092	0.39	0.0096	0.41	0.0035	0.01	0.0035	0.00
L1 chunk query (S^V)	0.2289	9.85	0.2254	9.76	6.1786	12.32	12.3800	12.42
Level output sum	0.0264	1.14	0.0260	1.13	0.4563	0.91	0.9047	0.91
Output projection	0.0909	3.91	0.0904	3.91	2.6487	5.28	5.2842	5.30
Unaccounted/Python	0.2332	10.03	0.2333	10.10	0.0617	0.12	0.0622	0.06
End-to-end	2.3237	100	2.3095	100	50.1626	100	99.6871	100

A.5 Limitations

The strict vocabulary-size bound from Propositions 2 and 3 applies to the first layer; deeper-layer control depends on the measured matching ratios in Proposition 4. If p_l is small, the theoretical bound degenerates. The implementation therefore uses a user-chosen capacity c_l, which is an external approximation and requires continued training; it is not implied by the proposition.

Fixed-state compression also remains weaker than full attention on difficult multi-key retrieval and reasoning. GPA substantially improves over other purely linear conversions on S-NIAH-2/3, but its absolute performance is still low without hybrid layers or targeted fine-tuning. Hybrid GPA and BABILong fine-tuning show that the gap is reducible, but they add Softmax computation or task-specific training. Moreover, the current parallel prefill kernel trades persistent-state savings for temporary workspace, so constant decoding memory should not be confused with minimal peak prefill memory. Finally, most accuracy tables report the deterministic lm-evaluation-harness point estimates from a single converted checkpoint; efficiency values are averages over repeated timed runs, but complete run-level variance is not available for every legacy baseline.

A.6 Proof of Proposition 1

A.6.1 Proof of the Storage Upper Bound

Without loss of generality, we first consider the case where both batch size and number of heads equal 1. Assume that the numeric type has bit-width b and the head dimension is d. For the infinitely growing key and value sequences, denote them as matrices of shape $\mathbb{R}^{t\times d}$, and concatenate them along the feature dimension into

$$\boldsymbol{S}^{KV} \in \mathbb{R}^{t\times(d_k+d_v)}, \quad \boldsymbol{S}^{K} = \boldsymbol{S}^{KV}[:, : d_k], \quad \boldsymbol{S}^{V} = \boldsymbol{S}^{KV}[:, d_k : d_k + d_v].$$

Let the count matrix be $\boldsymbol{c} \in \mathbb{R}^{t\times 1}$.

Proof. Since each numerical component is represented with b bits, each component can take 2^b distinct values. The dimensions of a vector $\mathbf{u} \in \mathbb{R}^{d_k+d_v}$ are independent; therefore, the total number of possible distinct vectors C is

$$\prod_{m=1}^{d_k+d_v} 2^b = 2^{b(d_k+d_v)}.$$

The distinct key-value pairs appearing in the sequence must belong to a subset of this finite set, hence $C \le 2^{b(d_k+d_v)}$. Storing each unique vector requires $b(d_k + d_v)$ bits, and each count requires b_c bits. Thus, the upper bound on total storage S_b is $C \times (b(d_k + d_v) + b_c)$, and combining this with the bound on C yields the desired inequality. □

A.6.2 Proof of Attention Equivalence

Proof. Let $\mathcal{I}_i = \{t \mid (\mathbf{k}_t, \mathbf{v}_t) = (\mathbf{k}_i, \mathbf{v}_i)\}$ be the set of time indices where the i-th unique key-value pair appears; then $|\mathcal{I}_i| = c_i$. The original attention output can be written as

$$\mathbf{o} = \sum_{t=1}^{T} \frac{e^{\mathbf{q}^\top \mathbf{k}_t}}{Z} \mathbf{v}_t, \quad Z = \sum_{t=1}^{T} e^{\mathbf{q}^\top \mathbf{k}_t}.$$

Grouping the summation by unique pairs:

$$\mathbf{o} = \sum_{i=1}^{C} \sum_{t\in\mathcal{I}_i} \frac{e^{\mathbf{q}^\top \mathbf{k}_t}}{Z} \mathbf{v}_t.$$

Since for any $t \in \mathcal{I}_i$, we have $\mathbf{k}_t = \mathbf{k}_i$ and $\mathbf{v}_t = \mathbf{v}_i$, it follows that

$$\sum_{t\in\mathcal{I}_i} \frac{e^{\mathbf{q}^\top \mathbf{k}_t}}{Z} \mathbf{v}_t = c_i \frac{e^{\mathbf{q}^\top \mathbf{k}_i}}{Z} \mathbf{v}_i.$$

Moreover, the denominator can be expressed as $Z = \sum_{i=1}^{C} \sum_{t \in \mathcal{I}_i} e^{\mathbf{q}^\top \mathbf{k}_t} = \sum_{i=1}^{C} c_i e^{\mathbf{q}^\top \mathbf{k}_i}$. Substituting these expressions yields

$$\mathbf{o} = \sum_{i=1}^{C} c_i \frac{e^{\mathbf{q}^\top \mathbf{k}_i}}{\sum_{j=1}^{C} c_j e^{\mathbf{q}^\top \mathbf{k}_j}} \mathbf{v}_i,$$

which is precisely the attention output computed from the compressed representation. This completes the proof. □

A.6.3 Comparison with the Delta Rule Formulation

Proof. We now describe the KV cache storage process using the deduplication mechanism in terms of the update rules of linear attention. Consider the unique-filtered sequence as the state $\boldsymbol{S}_t^{(K)}$, where t denotes the current time step or the row index up to which the sequence has been processed. $\boldsymbol{S}_t^{(K)}$ represents the KV cache maintained at time t. For linear attention, the size of $\boldsymbol{S}_t^{(K)}$ is fixed; for standard KV cache, each new element is concatenated to the previous cache. For the deduplication-based KV cache, its update rule can be expressed as:

$$\Delta(\boldsymbol{k}_t) = \boldsymbol{k}_t - Q\big(\boldsymbol{k}_t {\boldsymbol{S}_{t-1}^{(K)}}^\top\big)\, \boldsymbol{S}_{t-1}^{(K)}, \qquad \Delta(\boldsymbol{v}_t) = \boldsymbol{v}_t - Q\big(\boldsymbol{v}_t {\boldsymbol{S}_{t-1}^{(V)}}^\top\big)\, \boldsymbol{S}_{t-1}^{(V)}.$$

If $\Delta(\boldsymbol{k}_t) \le 1 - \text{threshold}$, then $\boldsymbol{S}_t^{(K)} = \boldsymbol{S}_{t-1}^{(K)}$. Otherwise, if $\Delta(\boldsymbol{k}_t) > 1 - \text{threshold}$, we update via concatenation:

$$\boldsymbol{S}_t^{(K)} = \text{concat}\big(\boldsymbol{S}_{t-1}^{(K)}, \boldsymbol{k}_t\big).$$

For the unique operation, the effective threshold is 1. And the Q function is defined as:

$$\mathrm{Q}(a_{ij}) = \begin{cases} 1 & \text{if } a_{ij} \text{ is max} \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

When the state size is manually limited to m, as in linear attention, once $\boldsymbol{S}_t^{(K)}$ reaches size m, further growth is prohibited. In this case, information differences must be integrated into the previous state via gating, yielding an update analogous to the Delta Rule:

$$\boldsymbol{S}_t^{(K)} = \big(1 - \beta \cdot Q(\boldsymbol{k}_t {\boldsymbol{S}_{t-1}^{(K)}}^\top)^\top\big)\, \boldsymbol{S}_{t-1}^{(K)} - \beta \cdot Q(\boldsymbol{k}_t {\boldsymbol{S}_{t-1}^{(K)}}^\top)^\top\, \Delta(\boldsymbol{k}_t).$$

When the key–value pairs are stored jointly as matrix states, the update becomes

$$\boldsymbol{S}_t^{(KV)} = \big(1 - \beta \cdot Q(\boldsymbol{k}_t {\boldsymbol{S}_{t-1}^{(K)}}^\top)^\top\big)\, \boldsymbol{S}_{t-1}^{(KV)} - \beta \cdot Q(\boldsymbol{k}_t {\boldsymbol{S}_{t-1}^{(K)}}^\top)^\top\, \Delta(\boldsymbol{v}_t),$$

which is essentially equivalent to the Delta Rule update from the fast-weight literature, except that $\phi(\boldsymbol{k}_t)$ is replaced by $Q(\boldsymbol{k}_t \boldsymbol{S}_{t-1}^{(K)})$. Moreover, MVA demonstrates that $Q(\boldsymbol{k}_t \boldsymbol{S}_{t-1}^{(K)})$ can be substituted with $Q(\boldsymbol{k}_t \boldsymbol{W}_c)$ to achieve comparable performance, and with carefully chosen Q functions, this becomes exactly equivalent to using $\phi(\boldsymbol{k}_t)$. □

A.7 Proof of Proposition 2

We provide a constructive proof.

A.7.1 Construction of the Compressed Representation

Proof. Given the original first-layer input $\tilde{\boldsymbol{X}}^{(1)}$, we construct the triple $(\boldsymbol{E}, \boldsymbol{I}, \tilde{\boldsymbol{P}})$ as follows:

Vocabulary embedding matrix $\boldsymbol{E}$: Directly take the pre-trained word embedding matrix $\boldsymbol{E}_{\text{vocab}}$. Since this matrix remains unchanged during inference, it does not need to be stored repeatedly in the compressed representation (it can be considered as shared parameters).

Index vector $\boldsymbol{I}$: For each position t, record its token index w_t, i.e. $\boldsymbol{I}_t = w_t$. Because $w_t \in \{1, \ldots, V\}$, each index can be stored using $\lceil \log_2 V \rceil$ bits.

Positional information matrix $\tilde{\boldsymbol{P}}$: To reconstruct $\tilde{\boldsymbol{X}}^{(1)}$ from $(\boldsymbol{E}, \boldsymbol{I})$, we need to retain the effect of positional encoding. Define

$$\tilde{\boldsymbol{P}} \;=\; \mathrm{PE}^{-1}\big(\tilde{\boldsymbol{X}}^{(1)},\; \boldsymbol{E}[\boldsymbol{I}, :]\big),$$

where PE^{-1} is the inverse operation of the positional encoding function (i.e., given the output and the word embeddings, recover the positional information). For common positional encoding schemes, PE^{-1} exists and is easy to compute:

- For absolute additive encoding: $\tilde{\boldsymbol{X}}^{(1)} = \boldsymbol{E}[\boldsymbol{I}, :] + \boldsymbol{P}$, then $\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{X}}^{(1)} - \boldsymbol{E}[\boldsymbol{I}, :] = \boldsymbol{P}$.
- For RoPE, we do not need to reconstruct the original sequence explicitly; instead we reconstruct the computation equivalent to attention losslessly, which yields a practically usable compression ratio. For completeness, we also sketch the reconstruction of the original sequence.

□

A.7.2 Lossless Reconstruction (RoPE Case)

Proof. Consider an input sequence of length T with token indices $w_1, \ldots, w_T \in \{1, \ldots, V\}$. Suppose the model employs RoPE positional encoding, whose core operation is to rotate each pair of dimensions $(2j, 2j+1)$ by an angle $\theta_j = \text{base}^{-2j/d}$, $j = 0, \ldots, d/2 - 1$. Denote the rotation parameters as $\tilde{\boldsymbol{P}} = \Theta = \{\theta_j\}_{j=0}^{d/2-1}$. RoPE combines an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^d$ with a position index m into a rotated vector $\tilde{\boldsymbol{x}}_m = f_{\text{RoPE}}(\boldsymbol{x}, m, \Theta)$ defined by

$$\begin{cases} \tilde{x}_{m,2j} = x_{2j}\cos(m\theta_j) - x_{2j+1}\sin(m\theta_j), \\ \tilde{x}_{m,2j+1} = x_{2j}\sin(m\theta_j) + x_{2j+1}\cos(m\theta_j), \end{cases} \qquad j = 0, \ldots, \frac{d}{2} - 1.$$

The first-layer input is the matrix obtained by applying RoPE to the word embedding sequence:

$$\tilde{\boldsymbol{X}}^{(1)} = [\tilde{\boldsymbol{e}}_{w_1,1}, \ldots, \tilde{\boldsymbol{e}}_{w_T,T}]^\top \in \mathbb{R}^{T\times d},$$

where $\tilde{\boldsymbol{e}}_{w_t,t} = f_{\text{RoPE}}(\boldsymbol{e}_{w_t}, t, \Theta)$.

To verify that the compressed representation $(\boldsymbol{E}, \boldsymbol{I}, \Theta)$ can losslessly recover $\tilde{\boldsymbol{X}}^{(1)}$, it suffices to note that for any position t, the vector $\tilde{\boldsymbol{e}}_{w_t,t}$ is uniquely determined by $\boldsymbol{e}_{w_t}$, t, and Θ via the RoPE definition. Hence, given $\boldsymbol{E}$, $\boldsymbol{I}$, and Θ, we can reconstruct the entire $\tilde{\boldsymbol{X}}^{(1)}$ position by position, exactly matching the original definition. Thus the reconstruction is lossless.

Moreover, in subsequent attention computations we do not need to reconstruct the full matrix explicitly. For any query position n and key position s, the (unnormalized) attention score is

$$A_{n,s} = \exp\left(\frac{1}{\sqrt{d}}\tilde{\boldsymbol{e}}_{w_n,n}^\top \tilde{\boldsymbol{e}}_{w_s,s}\right).$$

The inner product $\tilde{\boldsymbol{e}}_{w_n,n}^\top \tilde{\boldsymbol{e}}_{w_s,s}$ can be expressed using only $\boldsymbol{e}_{w_n}$, $\boldsymbol{e}_{w_s}$, the relative position $\Delta = n - s$, and the parameters Θ. Indeed, from the inner product property of RoPE:

$$\begin{aligned} \tilde{\boldsymbol{e}}_{w_n,n}^\top \tilde{\boldsymbol{e}}_{w_s,s} = \sum_{j=0}^{d/2-1} \big[&(e_{w_n,2j}e_{w_s,2j} + e_{w_n,2j+1}e_{w_s,2j+1})\cos(\Delta\theta_j) \\ &+ (e_{w_n,2j}e_{w_s,2j+1} - e_{w_n,2j+1}e_{w_s,2j})\sin(\Delta\theta_j)\big]. \end{aligned} \tag{38}$$

Thus, knowing the word embeddings $\boldsymbol{e}_{w_n}$, $\boldsymbol{e}_{w_s}$ (obtained from $\boldsymbol{E}$ via the indices) and the relative position Δ (computed from the position indices) together with θ_j from Θ, we can compute this inner product. This means the entire attention computation can be performed directly on the compressed representation, without storing the rotated vectors for each position or reconstructing the full matrix. □

A.7.3 Upper Bound on the Number of Distinct Row Vectors

Proof. Before positional encoding, every row of $\boldsymbol{E}_{\text{seq}}$ is selected from the V rows of $\boldsymbol{E}_{\text{vocab}}$. Hence $|\mathcal{U}_{\text{content}}| \leq V$, with equality when all vocabulary items occur. RoPE can map the same content vector to different position-entangled vectors, so no vocabulary-size bound is claimed for the rows of $\tilde{\boldsymbol{X}}^{(1)}$ themselves. The role of the decomposition $(\boldsymbol{E}, \boldsymbol{I}, \Theta)$ is precisely to keep the bounded content types separate from the position-dependent transformation. □

A.7.4 Storage Size Analysis

Proof. The vocabulary embedding matrix $\boldsymbol{E}$ occupies $V \cdot d \cdot b$ bits, but these parameters already exist in the model and need no extra storage in the compressed representation (they are merely referenced).

The index vector $\boldsymbol{I}$ requires storing T integers, each in the range $[1, V]$, hence $T\lceil \log_2 V \rceil$ bits.

The rotation parameters Θ contain $d/2$ scalars, each of b bits, totalling $(d/2) \cdot b$ bits, which is independent of the sequence length T.

Therefore, the total storage overhead of the compressed representation is $T\lceil \log_2 V \rceil + (d/2) \cdot b$ bits, compared to the original input which uses $T \cdot d \cdot b$ bits. This achieves a compression from $O(Td)$ to $O(T \log V + d)$, which is particularly significant when $d \gg \log V$. □

A.8 Proof of Proposition 3

A.8.1 Error Estimate of Proposition 3

Proof. We provide a step-by-step derivation that combines the original reasoning with necessary definitions and rigorous estimates.

1. Notation and basic identities. Let $\tilde{\boldsymbol{q}}_n = f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))$ and $\tilde{\boldsymbol{k}}_s = f_p(\boldsymbol{k}_s, \boldsymbol{p}(s))$ denote the vectors after applying RoPE. The exact unnormalized attention score is

$$A_{n,s} = \exp\left(\frac{1}{\sqrt{d}} \tilde{\boldsymbol{q}}_n^\top \tilde{\boldsymbol{k}}_s\right).$$

Using the RoPE inner product formula,

$$\tilde{\boldsymbol{q}}_n^\top \tilde{\boldsymbol{k}}_s = \sum_{j=0}^{d/2-1} \Big[(q_{n,2j}k_{s,2j} + q_{n,2j+1}k_{s,2j+1})\cos(\Delta\theta_j) + (q_{n,2j}k_{s,2j+1} - q_{n,2j+1}k_{s,2j})\sin(\Delta\theta_j)\Big],$$

where $\Delta = n - s$. For convenience we also denote the unrotated query–key inner product as

$$\begin{aligned}\tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s = \sum_{j=0}^{d/2-1} \big[&q_{n,2j}k_{s,2j}\cos(n\theta_j) - q_{n,2j+1}k_{s,2j}\sin(n\theta_j) \\ &+ q_{n,2j}k_{s,2j+1}\sin(n\theta_j) + q_{n,2j+1}k_{s,2j+1}\cos(n\theta_j)\big].\end{aligned} \tag{39}$$

2. Small-angle expansion. Because base is large, θ_j is small for $j \geq 2$ under the assumption $|\Delta|\theta_j \leq 1$. We expand cos and sin:

$$\cos(\Delta\theta_j) = 1 - \frac{(\Delta\theta_j)^2}{2} + O((\Delta\theta_j)^4), \qquad \sin(\Delta\theta_j) = \Delta\theta_j + O((\Delta\theta_j)^3).$$

Substituting these into $\tilde{\boldsymbol{q}}_n^\top \tilde{\boldsymbol{k}}_s$ yields

$$\tilde{\boldsymbol{q}}_n^\top \tilde{\boldsymbol{k}}_s = \tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s + \tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s),$$

where $\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s$ collects the first-order effects of rotation. More explicitly,

$$\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s) = \sum_j \tilde{\boldsymbol{q}}_{nj}^\top \boldsymbol{k}_{sj} \Big[(\cos(\Delta\theta_j) - 1) + \sin(\Delta\theta_j)\Big],$$

with each term behaving as $O(\Delta\theta_j)$ plus higher-order corrections.

3. Exponential expansion. Write the exact score as

$$A_{n,s} = \exp\left(\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s\right) \exp\left(\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\right).$$

Expand the second exponential using the Taylor series $e^u = 1 + u + O(u^2)$:

$$\exp\left(\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\right) = 1 + \frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s) + O\left(\frac{1}{d}\big(\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\big)^2\right).$$

Therefore

$$A_{n,s} = e^{\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s}\left(1 + \frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\right) + e^{\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s} \cdot O\left(\frac{1}{d}\big(\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\big)^2\right).$$

4. Error estimate. The error is bounded by the remainder term:

$$|A_{n,s} - \tilde{A}_{n,s}| \le e^{\frac{1}{\sqrt{d}}\tilde{\boldsymbol{q}}_n^\top \boldsymbol{k}_s} \cdot \frac{C}{d}\big(\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)\big)^2,$$

where C is an absolute constant (coming from the exponential factor and the bound on the remainder). From the small-angle expansion,

$$\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s) = \sum_j \tilde{\boldsymbol{q}}_{nj}^\top \boldsymbol{k}_{sj}\left[-\tfrac{1}{2}(\Delta\theta_j)^2 + \Delta\theta_j + O((\Delta\theta_j)^3)\right].$$

For the j-th dimension pair, the contribution to $\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s)$ is

$$\tilde{\boldsymbol{q}}_{nj}^\top \boldsymbol{k}_{sj}\left[(\cos(\Delta\theta_j) - 1) + \sin(\Delta\theta_j)\right].$$

Using the Taylor expansions of $\cos$ and $\sin$, this term becomes

$$\tilde{\boldsymbol{q}}_{nj}^\top \boldsymbol{k}_{sj}\left[-\frac{(\Delta\theta_j)^2}{2} + \Delta\theta_j\right].$$

Substituting into the error bound $O\left(\frac{1}{d}(\tilde{\boldsymbol{q}}_n^\top (\tilde{\boldsymbol{k}}_s - \boldsymbol{k}_s))^2\right)$, the overall error is of order

$$O\left(C_{qk}(\Delta\theta_{j+m})^2\right),$$

with

$$C_{qk} = \frac{1}{d}\sum_m \tilde{\boldsymbol{q}}_{nj}^\top \boldsymbol{k}_{sj}\, \tilde{\boldsymbol{q}}_{nm}^\top \boldsymbol{k}_{sm}.$$

Since $\theta_j = \mathrm{base}^{-2j/d}$, the largest term for $j \ge 2$ is $\mathrm{base}^{-4/d}$, and the sum over j is dominated by the smallest index in the range considered. For a fixed dimension index j, the error contributed by that frequency is $O(\Delta^2 \mathrm{base}^{-4j/d})$. Because Δ can be as large as the sequence length n (if we consider the query position n), we obtain an overall error bound

$$|A_{n,s} - \tilde{A}_{n,s}|_j = O\left(\frac{n}{\mathrm{base}^{4j/d}}\right),$$

where the notation $\mathrm{base}^{4j/d}$ is understood as the dominant decay rate. The factor n arises from the possible worst-case relative position; in practice the error may be smaller due to averaging. □

A.8.2 Approximate Attention of Proposition 3

The first term on the right-hand side is exactly $\tilde{A}_{n,s}$ if we identify

$$f_p(\boldsymbol{q}_n, \boldsymbol{p}(n)) = \tilde{\boldsymbol{q}}_n, \quad f_p(\boldsymbol{k}_s, \boldsymbol{p}^{(t)}(s)) = \tilde{\boldsymbol{k}}_s,$$

and interpret the scalar t as an additive term coming from the linear superposition (e.g., representing a cumulative sum of positional increments). In practice, the expression

$$t + f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))^\top f_p(\boldsymbol{k}_s, \boldsymbol{p}^{(t)}(s)) - f_p(\boldsymbol{q}_n, \boldsymbol{p}(n))^\top \boldsymbol{k}_s$$

captures the first-order effect together with a baseline t that can be absorbed into the normalization. Thus $\tilde{A}_{n,s}$ is precisely the first-order Taylor approximation of $A_{n,s}$.

Proof. The original attention computation can be expressed as:

$$\begin{aligned}&\exp\left(\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\right)\\&\quad\cdot\sum_{s}\exp\left\{\begin{array}{l}\dfrac{1}{\sqrt{d}}\displaystyle\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1)\\+\displaystyle\sum_{j=0}^{d}\frac{(-1)^{j+1}}{\sqrt{d}} q_{nj}\cdot k_{s,(j+\frac{d}{2})\%d}\cdot(\sin[(n-s)\theta_j])\end{array}\right\}\end{aligned}\tag{40}$$

Let us define the residual term:

$$\begin{aligned}r(\theta_1,\ldots,\theta_{d/2-1}) = &\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}k_{sj}\big(\cos((n-s)\theta_j)-1\big)\\&+\frac{1}{\sqrt{d}}\sum_{j=0}^{d}(-1)^{j+1}q_{nj}k_{s,(j+d/2)\bmod d}\sin((n-s)\theta_j).\end{aligned}\tag{41}$$

Since $\theta_j = \text{base}^{-\frac{2j}{d}}$, when base is large, we have the approximations:

$$\cos\theta_j = 1-\frac{\theta_j^2}{2}+O(\theta_j^4)\tag{42}$$

$$\sin\theta_j = \theta_j + O(\theta_j^3)\tag{43}$$

Following the CRG NTK method which has been proven to extend context window length and achieve excellent performance, we extend base to very large values (e.g., base $= 2^{40}\times 10000$). In this regime, $\cos\theta_j$ and $\sin\theta_j$ become very small, particularly for dimensions with larger j values. Since $\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}$ is typically on the order of 1, we can apply Taylor expansion to $e^{r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})}$:

$$e^{r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})} = 1 + r(\theta_1,\ldots,\theta_{\frac{d}{2}-1}) + O(r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})^2)\tag{44}$$

After base expansion, $r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})$ becomes small ($O(\theta_j)$ first-order term), making the higher-order terms $O(\theta_j^2)$ negligible, particularly in the latter half of the feature dimensions where $x_{nj}\cdot x_{sj}$ is diluted to near-zero values.

Based on the Taylor expansion of the exponential function when the input is close to zero, we retain only the linear small term $x_{nj}\cdot x_{sj}\cdot(\cos[(n-s)\theta_j]-1)$. The original attention can thus be approximated as:

$$\begin{aligned}&e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}}\sum_{s}\left(1+\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1)\right.\\&\qquad\left.+\frac{1}{\sqrt{d}}\sum_{j=0}^{d}(-1)^{j+1}q_{nj}\cdot k_{s,(j+(-1)^j)}\cdot(\sin[(n-s)\theta_j])\right)\end{aligned}\tag{45}$$

Alternatively, we can use the formulation:

$$\begin{aligned}&e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} x_{nj}x_{sj}}\sum_{s}\left(1+\frac{1}{\sqrt{d}}\sum_{j=0}^{d} x_{nj}x_{sj}\big(\cos[(n-s)\theta_j]-1\big)\right.\\&\qquad\left.+\frac{1}{\sqrt{d}}\sum_{j=0}^{d}(-1)^{j+1}x_{nj}x_{s,(j+(-1)^j)}\big(\sin[(n-s)\theta_j]-(n-s)\theta_j\big)\right)\end{aligned}\tag{46}$$

The residual term $r(\theta_1, \ldots, \theta_{\frac{d}{2}-1})$ can be expressed using linear attention, or we can first linearly superimpose the positional encoding before applying it to the K state sequence, enabling the entire formulation to be implemented with linear models.

For finer compression approximation, we can partition the dimensions into multiple segments and perform linear expansion separately:

$$\begin{aligned} e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}} \sum_{s} \prod_{p=0}^{d/m} \Bigg(& 1 + \frac{1}{\sqrt{d}} \sum_{j=p\cdot m}^{p\cdot m+m} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1) \\ & + \frac{1}{\sqrt{d}} \sum_{j=p\cdot m}^{p\cdot m+m} (-1)^{j+1} q_{nj}\cdot k_{s,(j+(-1)^j)}\cdot(\sin[(n-s)\theta_j]) \Bigg) \end{aligned} \tag{47}$$

We also explore variants that preserve the positional encoding for queries while applying $(\cos -1)$ transformation to the key sequence's positional encoding:

$$\begin{aligned} & \exp\left\{ \begin{array}{l} \frac{1}{\sqrt{d}} \sum_{j=0}^{d} (q_{nj}\cos(n\theta_j) - q_{n,(j+\frac{d}{2})\%d}\cdot\sin[n\theta_j]) \\ \qquad \cdot k_{sj} \end{array} \right\} \\ & \cdot \sum_{s} \exp\left\{ \begin{array}{l} \frac{1}{\sqrt{d}} \sum_{j=0}^{d} (q_{nj}\cos(n\theta_j) - q_{n,(j+\frac{d}{2})\%d}\cdot\sin[n\theta_j]) \\ \qquad \cdot (k_{sj}(\cos(s\theta_j)-1) - k_{s,(j+\frac{d}{2})\%d}\cdot\sin[s\theta_j]) \end{array} \right\} \end{aligned} \tag{48}$$

Applying the same Taylor expansion yields:

$$e^{f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top} \sum_{s} \left(1 + f_p(\mathbf{q}_n,\mathbf{p}(n)) f_p(\mathbf{k}_s,\mathbf{p}(s))^\top - f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top\right) \tag{49}$$

After simplification, we use the following formulation in our implementation, which achieves comparable or even better performance:

$$e^{f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top} \left(t + f_p(\mathbf{q}_n,\mathbf{p}(n)) f_p(\mathbf{k}_s,\mathbf{p}^{(t)}(s))^\top - f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top\right) \tag{50}$$

This positional encoding decoupling is functionally equivalent to the gating mechanisms in state-of-the-art models.

Proposition 3 establishes a rigorous foundation for compressing positional information while maintaining theoretical error bounds. The linear superposition approach for positional encoding enables efficient storage while the error analysis provides guarantees for practical deployment. The connection to gating mechanisms bridges theoretical compression techniques with established architectural components. □

A.9 Proof Sketch of Proposition 4

Proof. **1. Bounded difference from residual connections.** Each Transformer layer consists of residual connections and sublayers:

$$\boldsymbol{X}^{(l)} = \boldsymbol{X}^{(l-1)} + \mathrm{Attn}(\boldsymbol{X}^{(l-1)}) + \mathrm{FFN}\big(\boldsymbol{X}^{(l-1)} + \mathrm{Attn}(\boldsymbol{X}^{(l-1)})\big).$$

For simplicity, we focus on the main components and use Lipschitz continuity. The attention mechanism and the FFN are both Lipschitz continuous under common activation functions; thus there exist constants L_1, L_2 such that

$$\|\,\mathrm{Attn}(\boldsymbol{X}^{(l-1)})\|_F \le L_1 \|\boldsymbol{X}^{(l-1)}\|_F, \quad \|\,\mathrm{FFN}(\boldsymbol{Y})\|_F \le L_2 \|\boldsymbol{Y}\|_F.$$

Combined with bounded norms, there exists a constant $L > 0$ such that for each position t,

$$\|\boldsymbol{x}_t^{(l)} - \boldsymbol{x}_t^{(l-1)}\| \le L. \tag{1}$$

(Here $\boldsymbol{x}_t^{(l)}$ denotes the t-th row of $\boldsymbol{X}^{(l)}$.)

2. Mapping from original positions to unique vectors. Each original position t corresponds to a row vector $\boldsymbol{x}_t^{(l)}$ in layer l, which belongs to some unique vector $\boldsymbol{u}_i^{(l)} \in \mathcal{U}_l$. Similarly, $\boldsymbol{x}_t^{(l-1)}$ belongs to some $\boldsymbol{u}_j^{(l-1)} \in \mathcal{U}_{l-1}$. From (1), for this position t we have

$$\|\boldsymbol{u}_i^{(l)} - \boldsymbol{u}_j^{(l-1)}\| \leq L. \tag{2}$$

3. Relation between cosine similarity and Euclidean distance. Let θ be the angle between $\boldsymbol{u}_i^{(l)}$ and $\boldsymbol{u}_j^{(l-1)}$; then

$$\cos\theta = \frac{\langle \boldsymbol{u}_i^{(l)}, \boldsymbol{u}_j^{(l-1)} \rangle}{\|\boldsymbol{u}_i^{(l)}\| \|\boldsymbol{u}_j^{(l-1)}\|}.$$

By the law of cosines,

$$\|\boldsymbol{u}_i^{(l)} - \boldsymbol{u}_j^{(l-1)}\|^2 = \|\boldsymbol{u}_i^{(l)}\|^2 + \|\boldsymbol{u}_j^{(l-1)}\|^2 - 2\|\boldsymbol{u}_i^{(l)}\| \|\boldsymbol{u}_j^{(l-1)}\| \cos\theta.$$

Using the unit norm assumption ($\|\boldsymbol{u}_i^{(l)}\| = \|\boldsymbol{u}_j^{(l-1)}\| = 1$) and (2), we obtain

$$2(1 - \cos\theta) \leq L^2,$$

hence

$$1 - \cos\theta \leq \frac{L^2}{2}, \qquad \cos\theta \geq 1 - \frac{L^2}{2}.$$

This shows that each $\boldsymbol{u}_i^{(l)}$ has a high cosine similarity with the $\boldsymbol{u}_j^{(l-1)}$ corresponding to the same position. More generally, define

$$\sigma_i = \max_j \frac{\langle \boldsymbol{u}_i^{(l)}, \boldsymbol{u}_j^{(l-1)} \rangle}{\|\boldsymbol{u}_i^{(l)}\| \|\boldsymbol{u}_j^{(l-1)}\|} \geq 1 - \frac{L^2}{2}.$$

Thus $\rho_l = \frac{1}{M_l} \sum_i \sigma_i$ satisfies $\rho_l \geq 1 - \frac{L^2}{2}$.

4. One-to-one correspondence of high-similarity points. Consider those $\boldsymbol{u}_i^{(l)}$ for which there exists a j such that $\cos(\boldsymbol{u}_i^{(l)}, \boldsymbol{u}_j^{(l-1)}) \geq \tau$. Let $\mathcal{I} \subseteq \{1, \ldots, M_l\}$ be the set of such indices; then $|\mathcal{I}| = p_l M_l$. For each $i \in \mathcal{I}$, choose one such $j(i)$. The choice of τ guarantees that no two distinct $\boldsymbol{u}_i^{(l)}$ can be simultaneously similar to the same $\boldsymbol{u}_j^{(l-1)}$ (otherwise their distance would be less than ε, contradicting the precision assumption). Therefore the mapping $i \mapsto j(i)$ is injective, and we have

$$|\mathcal{I}| \leq M_{l-1},$$

i.e.,

$$p_l M_l \leq M_{l-1}. \tag{3}$$

5. Recursive upper bound. From (3) we obtain $M_l \leq M_{l-1}/p_l$. Since M_l cannot exceed the sequence length T (each position contributes at most one unique vector), we conclude

$$M_l \leq \min\left(\frac{M_{l-1}}{p_l},\ T\right). \tag{4}$$

For practical use, the trade-off among threshold, performance, and efficiency is controlled by $\rho_l = \frac{1}{M_l} \sum_i \sigma_i$.

6. Reverse inequality. By symmetry, considering the vectors in layer $l-1$ that have similarity $\geq \tau$ with some vector in layer l, we obtain $p_l' M_{l-1} \leq M_l$, i.e., $M_{l-1} \leq M_l / p_l'$. Together with the length bound this gives the reverse inequality. □

Remark. The threshold τ in the proposition depends on the numerical precision ε. In practice, ε can be determined by the quantization bit width (e.g., for b-bit quantization, it is the minimal distance between distinct quantized values, as in the deduplication approach described in Proposition 1). The quantity p_l can be interpreted as an average matching proportion, which may be estimated empirically or analytically. This recursive relationship indicates that the ratio of unique vector counts between adjacent layers is roughly inversely proportional to the high-similarity proportion, providing a theoretical foundation for hierarchical compression. When p_l is close to 1, the numbers of unique vectors in consecutive layers are nearly equal; when p_l is small, the next layer may produce many more new vectors, but this is ultimately bounded by the total sequence length.

Practical Implementation and Extensions

Since the similarity lower bound can approach zero, potentially leading to intermediate layer states approaching the original sequence length, we implement a practical constant scaling factor c in our experiments. This approach is motivated by several empirical observations:

1. Beyond a certain sequence length, attention between new query tokens and stored states becomes sparse (as observed in NSA, MoBA, etc.) 2. Many tasks can be completed with fixed-size state representations (as demonstrated in SnapKV, GLA, GSA) 3. The thinking process during inference often operates within bounded state spaces

We therefore treat $c_l \in [1, 2)$ as a user-set capacity hyperparameter: the layer may add $(c_l - 1)M_{l-1}$ states, for total capacity $c_l M_{l-1}$. This value is not derived from Proposition 4. Comparing it with the conditional requirement M_{l-1}/p_l gives the normalized shortfall $[1 - c_l p_l]_+$ from the main text. A nonzero shortfall is lossy and is absorbed into $\epsilon_{l,t}^{\mathrm{cap},K}$ and $\epsilon_{l,t}^{\mathrm{cap},V}$ in Theorem 1, which is why fine-tuning is required. Earlier layers empirically benefit from larger capacities, while later layers can often use smaller values.

As shown in Figure 3, this configuration achieves 100% performance on the passkey task. Note that the passkey task involves substantial noise insertion, resulting in high compression ratios; more complex tasks will naturally exhibit lower compression efficiency.

For handling the stored thinking states, we employ a GSA-like approach that preserves softmax operations:

$$\text{Storage}(X_{<t}) = \text{Softmax}(CX_{<t}^{\top})X_{<t} \tag{51}$$

where C is a learnable parameter functioning as a dynamic vocabulary. While fixed-ratio scaling provides complete upper bound control, it constitutes a lossy compression scheme.

After applying these compression steps, the sequence state size (X length or KV length) at each layer becomes $O(CV)$, where C is a constant multiple of the vocabulary size. However, practical challenges remain: for models like Qwen2.5 and LLaMA3 with vocabulary sizes around 128K, setting $c = 1.5$ for a 32-layer model results in an upper bound of approximately $1.5^{16} \times 128\text{K} \approx 82\text{M}$ states. This necessitates further compression strategies.

Inspired by MVA's vocabulary decomposition and MoM's functional partitioning approaches, we introduce multi-memory states to approximate the bounds established in Proposition 4 while maintaining fixed-size representations.

Corollary 1. *The inter-layer similarity mechanism enables adaptive compression ratios across different network depths, with early layers accommodating more state diversity and later layers leveraging the vocabulary projection for efficient compression. This aligns with the observed "thinking" pattern in transformer architectures.*

A.10 Proof of Proposition 5

Let $X \in \mathbb{R}^{n \times d}$ and let $\{C^{(i)}\}_{i=1}^{c}$ be vocabulary matrices, where $C^{(i)} \in \mathbb{R}^{m_i \times d}$ contains m_i prototypes. Multi-level residual decomposition represents each element x_j as

$$\hat{x}_j = \sum_{i=1}^{c} C_{k_j^{(i)}}^{(i)}, \tag{52}$$

where $k_j^{(i)} \in \{1, \dots, m_i\}$.

Proof. **Part 1: Representation Capacity**

The number of index tuples follows directly from the product rule:

$$\text{number of tuples} = \prod_{i=1}^{c} m_i. \tag{53}$$

Thus there are at most $\prod_i m_i$ reconstructions. Equality requires the mapping from index tuples to vector sums to be injective; without that condition, collisions can reduce the effective capacity. This qualification is important because the product is a nominal capacity, not an unconditional guarantee that training uses every path distinctly.

Part 2: Error Analysis

Define residuals recursively:

$$r_j^{(0)} = x_j \tag{54}$$

$$r_j^{(i)} = r_j^{(i-1)} - C^{(i)}_{k_j^{(i)}} \quad \text{for } i = 1, 2, \dots, c \tag{55}$$

The final approximation is:

$$\hat{x}_j = \sum_{i=1}^{c} C^{(i)}_{k_j^{(i)}} = x_j - r_j^{(c)} \tag{56}$$

Thus, the approximation error is $\|x_j - \hat{x}_j\|_2 = \|r_j^{(c)}\|_2$.

At each level choose the prototype minimizing the residual norm:

$$k_j^{(i)} = \arg \min_{k \in \{1, \dots, m_i\}} \|r_j^{(i-1)} - C_k^{(i)}\|_2. \tag{57}$$

Assume the learned codebook at level i is contractive over the residual distribution, i.e., for some $0 \le \rho_i < 1$,

$$\mathbb{E}\|r_j^{(i)}\|_2^2 \le \rho_i \, \mathbb{E}\|r_j^{(i-1)}\|_2^2. \tag{58}$$

Repeated substitution gives

$$\mathbb{E}\|x_j - \hat{x}_j\|_2^2 = \mathbb{E}\|r_j^{(c)}\|_2^2 \le \mathbb{E}\|x_j\|_2^2 \prod_{i=1}^{c} \rho_i. \tag{59}$$

The multiplicative decrease therefore follows from the explicit contraction assumption; it is not a consequence of independence or the chain rule. □

Corollary 2 (Trade-off between Representation Capacity and Error)**.** *For equal level size m and a fixed number of stored prototypes $B = mc$, the capacity-only objective m^c is maximized in the continuous relaxation at $m = e$ and $c = B/e$ (hence near integer $m = 3$). This objective ignores the serial latency of increasing c; practical designs therefore use few wider levels and select the point empirically (Table 18).*

Proof. Using $c = B/m$, the log-capacity is

$$\log(\text{capacity}) = \frac{B}{m} \log m. \tag{60}$$

Differentiating with respect to m gives

$$\frac{d}{dm}\left(\frac{B\log m}{m}\right) = \frac{B(1-\log m)}{m^2}, \tag{61}$$

which vanishes at $m = e$ and is positive before and negative after that point. □

Multi-level decomposition can therefore provide multiplicative nominal capacity and multiplicative residual reduction, subject respectively to injectivity and contraction. These conditions make the scope of the claim explicit.

In practice, the vocabularies $\{C^{(i)}\}$ are learned to minimize the overall reconstruction error, and the independence assumption between levels may be relaxed through joint optimization, potentially yielding even better performance than the theoretical bounds suggest.

Remark. In practice, the vocabularies $\{\boldsymbol{C}^{(i)}\}$ are learned to minimize the overall reconstruction error, and the linear independence assumption can be relaxed through joint optimization, potentially yielding even better performance than the theoretical bounds suggest. The recursive update formulas in the proposition correspond to a greedy residual encoding scheme, which is closely related to orthogonal matching pursuit (OMP) and vector quantization in hierarchical codebooks.

A.11 Proof of Theorem 1

Proof. We prove the state, local-error, and cross-layer statements in turn.

Effective-state recursion. Exact merging in Proposition 1 replaces all occurrences of an identical pair by one representative and its count without changing attention. Proposition 2 gives $M_0 \le |\mathcal{V}|$ for first-layer content types. Applying Proposition 4 recursively yields

$$M_\ell \le \min\left(T, \frac{M_0}{\prod_{r=1}^{\ell} p_r}\right) \le \min\left(T, \frac{|\mathcal{V}|}{\prod_{r=1}^{\ell} p_r}\right). \tag{62}$$

This is a required-capacity bound before enforcing a practical fixed budget; any shortfall is recorded in the capacity-collision terms below.

Key and value reconstruction. Write the effective key and value perturbations after deduplication, capacity truncation, and H_ℓ residual levels as

$$\eta^K_{\ell,t} \le \epsilon^{\mathrm{dup},K}_{\ell,t} + \epsilon^{\mathrm{cap},K}_{\ell,t} + B_{\ell,K}\left(\prod_{i=1}^{H_\ell} \rho^K_{\ell,i}\right)^{1/2}, \tag{63}$$

$$\eta^V_{\ell,t} \le \epsilon^{\mathrm{dup},V}_{\ell,t} + \epsilon^{\mathrm{cap},V}_{\ell,t} + B_{\ell,V}\left(\prod_{i=1}^{H_\ell} \rho^V_{\ell,i}\right)^{1/2}. \tag{64}$$

The product terms follow directly from Proposition 5's contraction assumption. Count distortion and Proposition 3's score approximation contribute $\epsilon^{\mathrm{cnt}}_{\ell,t}$ and $\epsilon^{\mathrm{pos}}_{\ell,t}$; the practical enhanced reader contributes $\epsilon^{\mathrm{score}}_{\ell,t}$ and $\epsilon^{\mathrm{read}}_{\ell,t}$.

Single-layer output perturbation. Let $z_s = \boldsymbol{q}_{\ell,t}\boldsymbol{k}^\top_{\ell,s}/\sqrt{d_\ell}$ and $\tilde{z}_s$ be its GPA approximation after the count and positional corrections. By Cauchy–Schwarz,

$$\|z - \tilde{z}\|_\infty \le \frac{B_{\ell,Q}}{\sqrt{d_\ell}}\eta^K_{\ell,t} + \epsilon^{\mathrm{cnt}}_{\ell,t} + \epsilon^{\mathrm{pos}}_{\ell,t} + \epsilon^{\mathrm{score}}_{\ell,t}. \tag{65}$$

The Softmax map satisfies $\|\mathrm{softmax}(z) - \mathrm{softmax}(\tilde{z})\|_1 \le 2\|z - \tilde{z}\|_\infty$. Multiplying by values bounded by $B_{\ell,V}$ and adding direct value/readout perturbations gives the single-layer inequality in Theorem 1.

Cross-layer propagation. Replace GPA blocks by their Softmax counterparts one layer at a time and telescope the final hidden-state difference. If block r amplifies an incoming perturbation by at most Λ_r, the local layer-ℓ error is multiplied by at most $\prod_{r=\ell+1}^{L} \Lambda_r$. Summing the telescoping terms gives

$$E_{L,t} \leq \sum_{\ell=1}^{L} e_{\ell,t} \prod_{r=\ell+1}^{L} \Lambda_r. \tag{66}$$

Finally, exact merging/counting, exact positional representation, sufficient joint-path capacity, and zero residual/readout error make every local term zero, recovering exact attention. With fixed H_ℓ and state sizes, each token performs a constant amount of recurrent work and stores a sequence-length-independent number of vectors, yielding linear total time and constant vector-state memory in T. □

A.12 Enhanced Reading Mechanisms and Unified Formula

Proposition 6 (Necessity of Enhanced Reading Mechanisms)**.** *Many recurrent approaches read memory through a direct query–state product. Our ablations indicate that limited readout expressiveness, in addition to storage capacity, contributes to the remaining gap with Softmax attention. We therefore evaluate a hierarchical, multi-channel reader; this component is an empirical refinement and is separated from the bounded-state guarantee.*

We propose a sophisticated reading approach:

$$R_t^{(i+1)} \left[f(k_t^{(i)}), f^{(i+1)}(k_t^{(i)}) \right] = 1 \tag{67}$$

$$d_t^{(i)} = q_t {S^k}_t^{(i)\top} \tag{68}$$

$$e_t^{(i)} = \frac{\exp\left(d_t^{(i)}\right)}{\sum_{j=0}^{d-1} \exp\left(d_{tj}^{(i)}\right)} \tag{69}$$

$$a_t^{(i)} = e_t^{(i)} R_t^{(i)\top} \tag{70}$$

$$b_t^{(i)} = \frac{\exp\left(\sum_i \ln(a_t^{(i)})\right)}{a_t^{(i)}} \tag{71}$$

$$c_t^{(i)} = \left(n_t^{(i)} + q_t {S^p}_t^{(i)\top}\right) \tag{72}$$

$$T_t^{(i)} = R_t^{(i)} \left({S^v}_t^{(i)} \odot e_t^{(i)\top} \odot c_t^{(i)\top}\right) \tag{73}$$

$$o_t = \sum_i o_t^{(i)} = \sum_i \frac{b_t^{(i)}}{b_t^{(i)} \odot e_t^{(i)} \odot c_t^{(i)}} T_t^{(i)} \tag{74}$$

This mechanism implements a hierarchical access pattern through multiple channels. For comparison, the GSA reading mechanism Softmax$(q_t {S^k}_t){S^v}_t$ represents the simplest form of indirect reading. Our enhanced version replaces the Softmax with a sigmoid activation followed by learned transformations:

$$(\sigma(q_t {S^k}_t) W_\sigma) {S^v}_t, \quad \textit{where } \sigma(x) = \frac{1}{1+e^{-x}} \tag{75}$$

Further extending to multiple reading channels:

$$(q_t W_r + \sigma(q_t {S^k}_t) W_\sigma) {S^v}_t \tag{76}$$

This approach is equivalent to the first-order vocabulary case of MVA and gives a modest, consistent improvement (Table 8). With multi-level vocabularies, Softmax, perceptron, and multi-channel readers further improve over a single-state readout.

For completeness, the following equations give the version obtained directly from the five principles, without optional engineering refinements such as sigmoid routing. **Initial conditions:**

$$q_t = f_p(x_t W_Q, r_t^{(i)}), k_{pt} = f_p(x_t W_K, r_t^{(i)}), k_t^{(0)} = x_t W_K \in \mathbb{R}^{1\times d}, v_t^{(0)} = x_t W_V \in \mathbb{R}^{1\times d},$$
$$S^{k}{}_0^{(i)} = 0 \in \mathbb{R}^{m\times d}, n_0^{(i)} = 0 \in \mathbb{R}^{1\times m}, E_t^{(0)} = I_m \in \mathbb{R}^{m\times m},$$

Iterative process:

$$\begin{aligned} f^{(i)}(k_t^{(i)}) &= F(S^{k}{}_{t-1}^{(i)} k_t^{(i)\top})^\top, \quad F(x) = \begin{cases} 1 & \text{if } x_i \text{ is maximum} \\ 0 & \text{otherwise} \end{cases} \text{ or Softmax}(x) \\ n_t^{(i)} &= n_{t-1}^{(i)} + f^{(i)}(k_t^{(i)}), \quad \bar{f}^{(i)}(k_t^{(i)}) = \frac{f^{(i)}(k_t^{(i)})}{\max(n_t^{(i)}, 1)} \end{aligned} \tag{77}$$

$$\begin{aligned} S^{k}{}_t^{(i)} &= \text{diag}\left(1 - \bar{f}^{(i)}(k_t^{(i)})^\top\right) S^{k}{}_{t-1}^{(i)} + \bar{f}^{(i)}(k_t^{(i)})^\top k_t^{(i)} \\ S^{v}{}_t^{(i)} &= \text{diag}\left(1 - \bar{f}^{(i)}(k_t^{(i)})^\top\right) S^{v}{}_{t-1}^{(i)} + \bar{f}^{(i)}(k_t^{(i)})^\top v_t^{(i)} \end{aligned} \quad \text{(Theory 1)} \tag{78}$$

$$\begin{aligned} S^{p}{}_t^{(i)} &= S^{p}{}_{t-1}^{(i)} + f^{(i)}(k_t^{(i)})^\top (k_{pt} - k_t^{(0)}) \\ k_t^{(i+1)} &= k_t^{(i)} - f^{(i)}(k_t^{(i)}) S^{k}{}_t^{(i)} \\ v_t^{(i+1)} &= v_t^{(i)} - f^{(i)}(k_t^{(i)}) S^{v}{}_t^{(i)} \end{aligned} \quad \text{(Theories 2 \& 3)} \tag{79}$$

$$R_t^{(i+1)}\left[f(k_t^{(i)}), f^{(i+1)}(k_t^{(i)})\right] = 1 \quad \text{(Theory 5)} \tag{80}$$

$$\begin{aligned} d_t^{(i)} &= q_t S^{k}{}_t^{(i)\top}, \quad e_t^{(i)} = \text{softmax}(d_t^{(i)}) \\ a_t^{(i)} &= e_t^{(i)} R_t^{(i)\top}, \quad b_t^{(i)} = \frac{\exp\left(\sum_i \ln(a_t^{(i)})\right)}{a_t^{(i)}} \end{aligned} \quad \text{(Theories 4 \& 5)} \tag{81}$$

$$\begin{aligned} c_t^{(i)} &= n_t^{(i)} + q_t S^{p}{}_t^{(i)\top} \\ T_t^{(i)} &= R_t^{(i)}\left(S^{v}{}_t^{(i)} \odot e_t^{(i)\top} \odot c_t^{(i)\top}\right) \\ o_t &= \sum_i \frac{b_t^{(i)}}{b_t^{(i)} \odot e_t^{(i)} \odot c_t^{(i)}} T_t^{(i)} \end{aligned} \quad \text{(Theories 4 \& 5)} \tag{82}$$

Chunk-wise Parallel Form

For minibatch processing:

$$R_t^{(i+1)} = R_{t-1}^{(i+1)} + \left(f^{(i)}(K^{(i)})\right)^\top \left(f^{(i+1)}(K^{(i+1)})\right) \tag{83}$$

$$R_t^{(i+1)} = R_{t-1}^{(i+1)} + (Y - R_{t-1}^{(i+1)}) \odot \left(f^{(i)}(K^{(i)})\right)^\top \left(f^{(i+1)}(K^{(i+1)})\right) \tag{84}$$

$$N^{(i)} = \text{CumSum}\left(f^{(i)}(K^{(i)})\right) \tag{85}$$

$$\bar{f}^{(i)}(K^{(i)}) = \frac{f^{(i)}(K^{(i)})}{N^{(i)}} \tag{86}$$

$$D^{(i)} = \text{GLA}^\top\left(Q, K^{(i)}, \bar{f}^{(i)}(K^{(i)}), 1 - \bar{f}^{(i)}(K^{(i)})\right) \tag{87}$$

$$E^{(i)} = \text{softmax}(D^{(i)}) \tag{88}$$

$$A^{(i)} = E^{(i)}(R^{(i)})^\top \tag{89}$$

$$C^{(i)} = \text{GLA}^{\top}\left(Q, f_p(K^{(i)}, P) - K^{(i)}, \bar{f}^{(i)}(K^{(i)}), 1 - \bar{f}^{(i)}(K^{(i)})\right) + N^{(i)} \tag{90}$$

$$T^{(i)} = R^{(i)}\text{GLA}\left(C^{(i)} \odot E^{(i)}, \bar{f}^{(i)}(K^{(i)}), V^{(i)}, 1 - \bar{f}^{(i)}(K^{(i)})\right) \tag{91}$$

$$O^{(i)} = \frac{B^{(i)}}{B^{(i)} \odot E^{(i)} \odot C^{(i)}} \odot T^{(i)} \tag{92}$$

$$O = \sum_i O^{(i)} \tag{93}$$

Proof Sketch. The multi-level decomposition proposition builds upon the following insights:

1. **Error Analysis**: The multiplicative residual bound follows by recursively applying the per-level contraction inequality from Proposition 5. It does not require independent errors and does not follow from the chain rule.
2. **Capacity Analysis**: The storage capacity $\prod_{i=1}^{m} C_i$ results from the combinatorial nature of hierarchical representations. Each level provides a separate "alphabet" of size C_i, and the total number of expressible states is their product.
3. **Enhanced Reading Mechanism**: The sophisticated reading approach enables:
   - Cross-level information integration through the R matrix
   - Adaptive weighting through the $b_t^{(i)}$ terms
   - Position-aware modulation through the $c_t^{(i)}$ terms

The unified model integrates all five theoretical principles:

- Theory 1: Redundancy elimination through the $f^{(i)}$ functions
- Theories 2 & 3: Positional information compression through $k_{pt} - k_t^{(0)}$
- Theory 4: Inter-layer similarity through the hierarchical structure
- Theory 5: Multi-level decomposition and enhanced reading

Experimental validation shows that intermediate layers benefit from more states (m larger), while earlier and later layers can use fewer states. For fair comparison with existing work, we use single-level decomposition against GSA and two-level decomposition against MVA. □

Proposition 5 and Proposition 6 together provide a comprehensive framework for efficient state management in linear attention models. The multi-level approach offers exponential error reduction while the enhanced reading mechanism ensures effective information retrieval from the compressed representations.

Corollary 3. *For a model with m levels and vocabulary sizes C_i, the total number of parameters required for the reading mechanism scales as $O(\sum_{i=1}^{m} C_i d^2)$, providing a favorable trade-off between expressivity and efficiency compared to the $O(Nd^2)$ scaling of standard attention.*

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: For more details, please refer to the Introduction and Abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: For more details, please refer to A.5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: For example, Prop 1 and proof A.6

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The method and major experimental settings are described, but runnable training/evaluation scripts, complete per-run configurations, and checkpoints are not yet available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The repository address is `https://github.com/fisalt/GPA`; complete reproduction artifacts are not included in this version.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [No]

Justification: The experimental setup reports LoRA rank, state sizes and depth, datasets, token budgets, and major optimization settings; complete configurations for every supplemental run are not yet provided.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Most GPA results are point estimates without repeated-seed uncertainty or significance tests; we do not claim statistically significant differences.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Timing tests specify an NVIDIA A800 80GB GPU and benchmark configurations, but per-training-run GPU-hours and total project compute are not fully documented.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [N/A]

   Justification: [N/A]

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper discusses technical efficiency and modeling limitations, but does not provide a full discussion of positive and negative societal impacts.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer:[N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: Existing assets are cited, but a complete record of their versions, licenses, and applicable terms is not provided.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The model is described in Section 3, but executable code, checkpoints, and full release documentation have not yet been published.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM assistance was used in manuscript editing and organization, not as a component of the proposed architecture or for generating experimental measurements.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.