PLA: A Principled Path from Softmax Attention to Linear Models via KV Cache Compression

Anonymous authors
Paper under double-blind review

Abstract

Transformers, despite their remarkable sequence modeling capabilities, are fundamentally constrained by the quadratic complexity of Softmax attention and the unbounded growth of the key–value (KV) cache. Replacing Softmax attention with linear variants has emerged as a promising direction, yet existing approaches lack a systematic functional comparison with Softmax attention, clear error analysis, and a theoretically guided roadmap for improvement. In this work, we approach the problem from the perspective of KV cache compression and present a theoretically grounded pathway from Softmax attention to linear models. Our analysis reveals five critical components: redundancy elimination, tokenizer-level quantization and positional information separation, positional information compression, inter-layer similarity, and multi-state decomposition. For each, we provide succinct theoretical justification, derive error bounds, and demonstrate equivalence to existing mechanisms. Building on this pathway, we introduce PLA, a linearized attention model that inherits pretrained weights and achieves state-of-the-art performance. Notably, PLA surpasses strong baselines such as MVA and GSA on multiple benchmarks while requiring only 80% of the fine-tuning resources. Our findings provide both theoretical clarity and practical guidance for advancing linear attention, highlighting a principled route towards efficient and scalable alternatives to Softmax attention.

1 Introduction

The Transformer architecture Vaswani et al. (2017) has become the backbone of modern deep learning, powering state-of-the-art models in language Touvron et al. (2023); Jiang et al. (2023); DeepSeek-AI et al. (2024); Dubey et al. (2024); Yang et al. (2025), vision Dosovitskiy et al. (2020); Han et al. (2022), and multimodal Yin et al. (2024) domains due to its remarkable sequence modeling capability. Despite these successes, Transformers face two fundamental limitations: 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 issues severely constrain the applicability of Transformers in long-sequence modeling tasks, such as video understanding Tang et al. (2025), genomic sequence analysis Jumper et al. (2021), and other domains requiring extended context Jiang et al. (2025).

To address these limitations, two major lines of research have emerged: *KV cache compression* Luohe et al. (2024); WEI et al. (2025) and *linear attention* Katharopoulos et al. (2020); Hua et al. (2022); Qin et al. (2024a). KV cache compression methods aim to reduce memory usage by compressing the stored states across either the sequence or the channel dimension. For example, some approaches design task-specific prompts that select a fixed set of relevant KV entries to retain, thus improving efficiency at the cost of generality. Others Hu et al. (2021) apply low-rank or dimensionality reduction techniques along the channel dimension, yielding constant compression ratios but failing to fundamentally address the ever-growing cache size in long contexts.

Linear attention methods, in contrast, replace the Softmax kernel with kernelized approximations, thereby reordering computations to achieve linear complexity. Crucially, such models can recursively maintain a fixed-size state, resolving the KV cache growth problem. Recent works Yang et al. (2024b;a) further en-

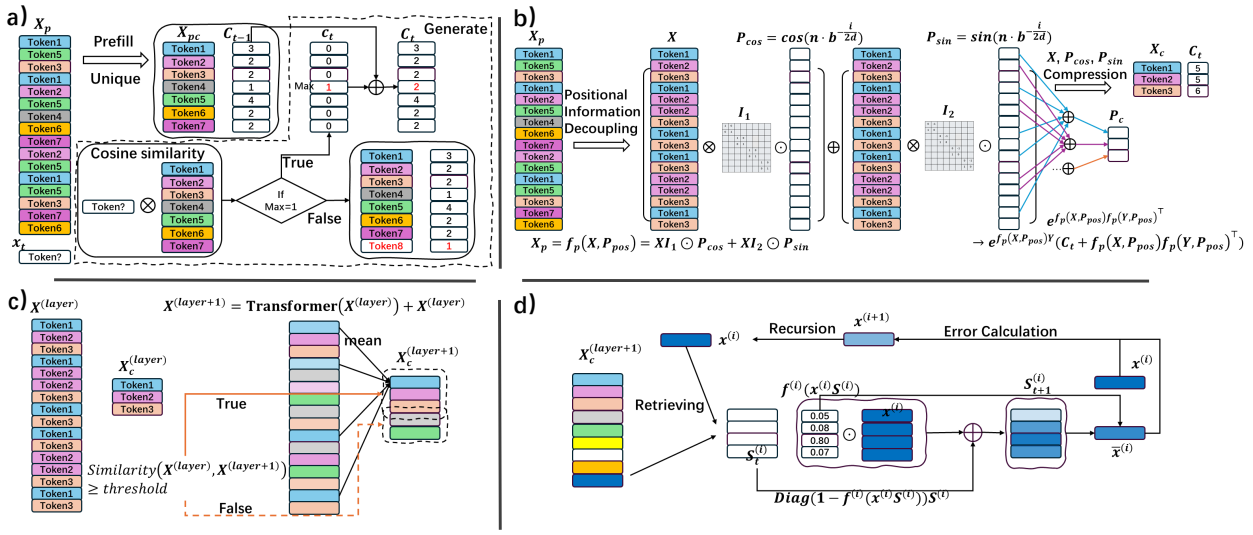

Figure 1: **Conceptual illustration of the RSA memory hierarchy and its theoretical foundations.** This figure illustrates the key components that enable bounded state size while preserving representational capacity, grounded in the theoretical analysis of Section 3. **(a) Redundancy elimination.** During 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.

hance these models with additional mechanisms such as gating functions and delta-rule updates to improve expressiveness. However, linear attention models still exhibit significant drawbacks: they often suffer from limited retrieval and reasoning capacity, exhibit noticeable performance gaps relative to Softmax attention, and typically require training from scratch or hybridization with Softmax attention to achieve competitive results.

In this work, we revisit the connection between Softmax attention and linear attention from the perspective of *KV cache compression*. We propose what we argue to be the current optimal and theoretically grounded pathway for compressing Softmax attention into linear models. This pathway is structured around five theoretical principles, each demonstrating (i) the necessity of a specific compression step, (ii) its equivalence to mechanisms in existing approaches, and (iii) the error it introduces relative to Softmax attention. Taken

together, these principles provide a clear functional blueprint of what linear attention should retain, what it can safely discard, and how it differs fundamentally from Softmax attention.

Our analysis yields both theoretical and practical benefits. First, it clarifies the essential components required to bridge the gap between Softmax and linear attention, guiding future designs of efficient architectures. Second, it enables the transformation of pretrained Softmax-based large language models into linear variants with significantly reduced fine-tuning cost. Empirically, we show that our approach achieves state-of-the-art performance, narrowing the gap between linear and Softmax attention especially on tasks where existing linear models struggle, such as retrieval, few-shot reasoning, and complex logical inference.

2 Background and Preliminaries

2.1 Transformers and Softmax Attention

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

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

where $\boldsymbol{M}$ denotes the causal mask with $\boldsymbol{M}_{ij} = 1$ if $i \geq j$ o.w. $-\infty$ and d is the feature dimension used for normalization. Equivalently, the autoregressive form can be written as $\boldsymbol{o}_i$, where $\boldsymbol{q}_i$, $\boldsymbol{k}_j$, and $\boldsymbol{v}_j$ are the i-th or j-th row vectors of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, respectively. While highly effective, this formulation entails quadratic complexity in sequence length and requires storing all past key–value pairs, leading to unbounded KV cache growth during inference.

2.2 KV Cache Compression Methods

To alleviate the quadratic growth of the key–value (KV) cache, a large body of work explores *KV cache compression*. The central idea is to reduce redundancy in the cache by performing low-rank transformations or selection operations along the sequence dimension or the channel dimension.

Formally, given an input $\boldsymbol{X} \in \mathbb{R}^{t\times d}$ and its 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 defined as

$$\boldsymbol{K}^c = \varphi\big(\phi(\boldsymbol{R}\boldsymbol{K})\boldsymbol{L}\big), \qquad \boldsymbol{V}^c = \varphi\big(\phi(\boldsymbol{R}\boldsymbol{V})\boldsymbol{L}\big), \tag{2}$$

where $\boldsymbol{R} \in \mathbb{R}^{c\times t}$ selects c tokens from the sequence, and $\boldsymbol{L} \in \mathbb{R}^{d\times d^r}$ compresses the channel dimension. Here $d^r = h \times d_k^r$, with h denoting the number of heads and d_k^r the per-head compressed dimension. The operators $\phi(\cdot)$ and $\varphi(\cdot)$ denote transformation and selection functions, respectively.

Because the sequence length t grows without bound during autoregressive generation, $\boldsymbol{R}$ is usually constructed recursively, i.e.,

$$\boldsymbol{R} = f(\boldsymbol{R}'\boldsymbol{X}^\top) \in \mathbb{R}^{c\times t}, \tag{3}$$

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

(1) **SnapKV** Li et al. (2025) applies compression only along the sequence dimension. Specifically, $\boldsymbol{L}$ is the identity matrix, so no channel compression is applied. $\boldsymbol{R} = f(\boldsymbol{Q}\boldsymbol{K}^\top)$ is defined via a `top-`c operator over the most recent queries and obtains the indexes of the corresponding KV block. $\phi(\cdot)$ gather the corresponding key tokens by these indices. The same procedure is applied to $\boldsymbol{V}$. (2) **HeadKV** Fu et al. (2025) extends SnapKV by compressing along the channel dimension. In this case, $\boldsymbol{R}$ again selects tokens as in SnapKV, while $\boldsymbol{L}$ is defined through an additional projection $\boldsymbol{L}'$, which serves as a voting mechanism across attention heads. $\varphi(\cdot)$ gather the corresponding key heads by these votes. (3) **Multi-Head Latent Attention (MLA)** DeepSeek-AI et al. (2025) approach removes explicit token selection. Both $\phi(\cdot)$ and $\varphi(\cdot)$ are set to the identity. No sequence compression is applied, i.e., $\boldsymbol{R} = \boldsymbol{I}$. Instead, channel compression is performed

with a fixed projection $\boldsymbol{L} = (\boldsymbol{W}^{UK})^{-1}$. By leveraging the associativity of matrix multiplication to fuse the up-projection with the query-key product, MLA avoids explicitly reconstructing $\boldsymbol{K}$ and $\boldsymbol{Q}$, achieving substantial memory savings and computational speedup.

Numerous subsequent approaches, including InfLLM Xiao et al. (2024a), HO2 Zhang et al. (2023), and StreamLLM Xiao et al. (2024b), can be understood as hybrids or equivalent reformulations of the above principles.

2.3 Linear Models

Linear attention replaces the Softmax kernel with linearizable feature maps, which permits re-ordering the computations among queries, keys and values and thereby achieves linear time and fixed-size state:

Parallel form.

$$\boldsymbol{O} = \mathrm{LA}\big(\phi(\boldsymbol{Q}), \phi(\boldsymbol{K}), \boldsymbol{V}\big) = \big(\big(\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top\big) \odot \boldsymbol{M}\big)\, \boldsymbol{V}, \tag{4}$$

Recursive form.

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \phi(\boldsymbol{k}_t)^\top \boldsymbol{v}_t, \qquad \boldsymbol{o}_t = \phi(\boldsymbol{q}_t)\, \boldsymbol{S}_t, \tag{5}$$

where $\boldsymbol{S}_t \in \mathbb{R}^{d_k \times d_v}$ is a fixed-size state matrix maintained across time steps. By keeping $\boldsymbol{S}_t$ bounded, linear attention attains constant memory during autoregressive inference. Many works focus on improving the choice of $\phi(\cdot)$ Han et al. (2023); Choromanski et al. (2022) or introducing auxiliary mechanisms to enhance expressiveness.

However, this formulation is prone to state saturation, which dilutes the attention mechanism. To address this, methods like GLA introduce a gating mechanism that enables dynamic forgetting in the state $\boldsymbol{S}_t$, thereby promoting a bias towards more recent context.

Gating / GLA. Gated Linear Attention (GLA) Yang et al. (2024b) applies multiplicative gates to control the contribution of new tokens and the persistence of prior state:

$$\boldsymbol{O} = \mathrm{GLA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{G}) = \mathrm{LA}(\boldsymbol{Q} \odot \boldsymbol{B},\ \boldsymbol{K}/\boldsymbol{B},\ \boldsymbol{V}), \boldsymbol{S}_t = \mathrm{diag}(\boldsymbol{g}_t)\, \boldsymbol{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t, \boldsymbol{o}_t = \boldsymbol{q}_t\, \boldsymbol{S}_t. \tag{6}$$

where the t-th row of $\boldsymbol{B}$ is $\boldsymbol{b}_t = \prod_{i=1}^{t} \boldsymbol{g}_i$ and $\boldsymbol{G} = \sigma(\boldsymbol{X}\boldsymbol{W}_g) \in \mathbb{R}^{n \times d_k}$. While models like MetaLA Chou et al. (2024), HGRN2 Qin et al. (2024b), and GSA Zhang et al. (2024) also employ gating, this approach offers a relatively coarse control over the state S_t, failing to fully address information redundancy. This limitation motivated the development of more sophisticated updates, such as those inspired by fast weights.

Fast-weight / Delta Rule-style updates. These methods aim to correct the stored representation based on prediction error:

$$\boldsymbol{v}_t^{\mathrm{old}} = \boldsymbol{k}_t\, \boldsymbol{S}_{t-1}, \qquad \boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{g}_t \cdot \boldsymbol{k}_t^\top \big(\boldsymbol{v}_t - \boldsymbol{v}_t^{\mathrm{old}}\big), \qquad \boldsymbol{o}_t = \boldsymbol{q}_t\, \boldsymbol{S}_t. \tag{7}$$

Intuitively, the update adds a correction proportional to the discrepancy between the newly observed value $\boldsymbol{v}_t$ and the state-predicted value $\boldsymbol{v}_t^{\mathrm{old}}$, gated by $\boldsymbol{g}_t$. Building on this, DeltaNet Yang et al. (2024a) parallelized the formulation to create a powerful linear model.

Optimization viewpoint. Furthermore, viewing the update through an optimization lens led to dynamic weighting methods like TTT Sun et al. (2025), Titans Behrouz et al. (2024), and Atlas Agrawal et al. (2025), which share the unified objective:

$$\mathcal{L}_t(\boldsymbol{M}) \;=\; \sum_{i=1}^{t} \gamma_i \left\| \boldsymbol{M}\big(\phi(\boldsymbol{k}_i)\big) - \boldsymbol{v}_i \right\|_2^2, \tag{8}$$

where $\boldsymbol{M}(\cdot)$ denotes a (possibly parametric) mapping from key-features to value-predictions and $\{\gamma_i\}$ are weighting coefficients. One may then update the memory/map $\boldsymbol{M}_t$ by performing a few steps of iterative optimization (e.g., gradient descent-GD or Muon Jordan et al. (2024)-style). A compact schematic of such an optimization-inspired update is:

$$\boldsymbol{M}_t = \alpha_t\, \boldsymbol{M}_{t-1} + F(\boldsymbol{S}_t), \quad \boldsymbol{S}_t = \eta_t\, \boldsymbol{S}_{t-1} - \theta_t\, \nabla_{\boldsymbol{S}}\, \mathcal{L}_t\big(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t\big), \tag{9}$$

where $F(\cdot)$ aggregates the current statistics into the primary memory $\boldsymbol{M}_t$, and the second line denotes a gradient-based (or similar) corrective step for the working state $\boldsymbol{S}_t$. This optimization viewpoint explains a number of empirically successful update rules and motivates algorithms that explicitly minimize per-step predictive error.

Extending the state budget. Even with sophisticated updates, a single low-rank state may remain insufficient to capture complex, multi-scale dependencies. Recent works therefore maintain multiple parallel or hierarchical states, each specialized for different temporal ranges or functional roles. For instance, **MoM** Du et al. (2025) and **MVA** Wang et al. (2025) maintain multiple memory banks (e.g., short-term vs. long-term) and/or decompose the state into several sub-states that interact during read/write.

$$\boldsymbol{S}_t^{(i)} = \mathrm{diag}(1 - \bar{f}^{(i)}(\boldsymbol{x}_t^{(i)})^\top)\boldsymbol{S}_{t-1}^{(i)} + \bar{f}^{(i)}(\boldsymbol{x}_t^{(i)})^\top \boldsymbol{x}_t^{(i)}, \qquad \boldsymbol{x}_t^{(i+1)} = f_1^{(i)}(\boldsymbol{x}_t^{(i)}, \boldsymbol{S}_t^{(i)}) \tag{10}$$

where f function is generally taken as σ, while the f_1 function is taken as a hybrid expert or delta function. This multi-state design can close much of the performance gap to Softmax attention, at the cost of additional architectural and algorithmic complexity.

3 Method

Existing KV cache compression methods either lack general applicability or fail to address the unbounded growth of KV cache, while also lacking a clear error analysis compared to Softmax Attention. Furthermore, current linear attention approaches lack a comprehensive understanding of their components and mechanisms from the perspective of Softmax Attention, resulting in the absence of clear improvement strategies to match or even surpass Softmax Attention performance.

To address these limitations, this paper presents five theoretical principles with corresponding experimental validation, establishing an optimal pathway for compressing Softmax Attention into linear attention. Each theoretical node provides rigorous error analysis and demonstrates equivalence to existing model operations and mechanisms, thereby offering valuable references and guidance for future improvement strategies.

3.1 Necessity of Redundancy Removal

Theorem 1 (Necessity of Redundancy Removal)**.** *For any sequence stored in the KV cache that exhibits translation-invariant properties, the application of redundancy removal operations (e.g., unique filtering) together with a counting mechanism can bound the storage size. Specifically, the upper bound is given by*

$$C \;\leq\; 2^{\,b_k \times d_k + b_v \times d_v} \times \big(b_k \times d_k + b_v \times d_v + b_c\big), \tag{11}$$

where only the unique KV vectors $\boldsymbol{u}_i$ and their counts c_i are stored as a set of unique pairs $\{(\boldsymbol{u}_i, c_i)\}_{i=1}^{C}$, with $i \;\leq\; 2^{\,b_k \times d_k + b_v \times d_v}$.

Here b denotes the bit-width of the number type for the KV vectors (e.g., $b = 16$ for `float16`*, $b = 4$ for* `int4`*), d_k and d_v are the dimensions of the key and value vectors (e.g., head_dim $= 128$ in Qwen and LLaMA), and b_c is the number of bits required to maintain counts. This compression is* lossless*, and the redundancy-removal mechanism is functionally equivalent to the Delta Rule used in existing fast-weight models.*

A detailed proof is provided in the Appendix A.1.

Generalization. The unique operation can also be generalized by relaxing exact matching to cosine similarity. For example, using a threshold of 0.9 instead of 1.0 can still preserve performance in A.1. This opens the possibility of balancing efficiency and accuracy by tuning the threshold. Furthermore, we propose a more general and stronger compression approach: when two tokens exhibit cosine similarity above a threshold, we treat them as identical and replace them by their average. This can be interpreted as a quantization process, which also provides noise reduction.

Complexity Implication. Let the upper bound of the KV cache be C. According to the above reasoning, Softmax attention can be interpreted as a linear model with complexity $\mathcal{O}(N \times C \times d)$, where $C = 2^{2bd}$. Since C is extremely large, the naive bound is impractical in comparison to specialized task-optimized methods. Nonetheless, the key insight here is that redundancy removal (via unique or Delta Rule operations) is *necessary* to compress an unbounded state into a bounded one.

3.2 Tokenization and Positional Information Decoupling

To achieve stronger compression, we conduct a deeper analysis of sequence modeling in LLMs. The input to an LLM undergoes tokenization, which constitutes a strong quantization that limits the number of distinct types to the vocabulary size V_T (e.g., 32K for LLaMA2). Subsequent channel mixing operations in the LLM do not affect the number of distinct types; rather, it is the positional encoding and token mixing operations that impact type diversity. We therefore optimize the input-output characteristics of these two operations, leveraging the first-layer tokenization to achieve stronger compression and reduce the upper bound of the state storage requirement.

We introduce two optimizations specifically targeting positional encoding operations:

Theorem 2 (Necessity of Positional Information Decoupling)**.** *For tokenizer-based models, the input sequence and KV cache at the first layer, after positional information decoupling, can be losslessly represented with an upper bound of $C = V_T$, where V_T is the vocabulary size. This is achieved by storing a tensor of size V_T along with indices for each vector. The limitation of this approach is that the indices grow unbounded with sequence length, theoretically only compressing storage by a factor of the dimension size.*

Due to positional constraints, the upper bound remains identical to Theorem 1. We subsequently address this limitation through positional information compression. The detailed proof is provided in Appendix A.2. This approach provides lossless compression for the first layer.

3.3 Positional Information Compression

Building upon the upper bound established in Theorem 2, we further optimize the positional information representation to achieve the same upper bound as the position-agnostic KV cache in Theorem 2. This allows us to completely control the first-layer upper bound at the vocabulary size level, laying the foundation for fully fixed-size state linear models.

Theorem 3 (Necessity of Positional Information Compression)**.** *With the compression method of positional encoding described below, the upper bound of the KV cache and the positional information in the first layer reaches the vocabulary size. For positions k_m and v_m at index m, we store the compressed positional information as a linear superposition $p_m^{(t)} = p_{m1} + p_{m2} + \cdots + p_{mt}$. The approximation error between the following attention formulation and the original attention is $O\left(\frac{n}{base^{\frac{2j}{d}}}\right)$:*

$$e^{f_p(\mathbf{q}_n, \mathbf{p}(n))\mathbf{k}_m^\top} \left(t + f_p(\mathbf{q}_n, \mathbf{p}(n)) f_p(\mathbf{k}_m, \mathbf{p}^{(t)}(m))^\top - f_p(\mathbf{q}_n, \mathbf{p}(n))\mathbf{k}_m^\top\right) \tag{12}$$

where the positional encoding functions are defined as:

$$\mathbf{p}^c(m, base) = \begin{bmatrix}\cos(m\theta_0) & \cos(m\theta_0) & \dots & \cos(m\theta_{d/2-1}) & \cos(m\theta_{d/2-1})\end{bmatrix} \tag{13}$$

$$\mathbf{p}^s(m, base) = \begin{bmatrix}\sin(m\theta_0) & \sin(m\theta_0) & \dots & \sin(m\theta_{d/2-1}) & \sin(m\theta_{d/2-1})\end{bmatrix} \tag{14}$$

$$\begin{aligned} f_p(\mathbf{x}_m, \mathbf{p}(m)) = &\begin{bmatrix}x_0 & x_1 & x_2 & x_3 & \dots & x_{d-1} & x_d\end{bmatrix} \odot \mathbf{p}^c(m) \\ &+ \begin{bmatrix}x_1 & -x_0 & x_3 & -x_2 & \dots & x_d & -x_{d-1}\end{bmatrix} \odot \mathbf{p}^s(m) \end{aligned} \tag{15}$$

The detailed proof is provided in Appendix A.6.

The separation of positional information effectively emulates gating mechanisms used in state-of-the-art architectures. Theoretical results show that this compression is most effective in the **first layer**. In deeper

layers, where token representations are increasingly entangled with positional cues, simple redundancy reduction cannot maintain the vocabulary-size bound. To address this, we introduce a **layer-wise similarity constraint**, where the **expected cosine similarity** between adjacent hidden states serves as a **contractive factor**. This yields a recursive bound that restricts the memory footprint of intermediate layers to a **constant multiple of the vocabulary size**, ensuring a fixed-state regime while preserving expressivity.

3.4 Inter-layer Similarity and State Propagation

Theorem 4 (Necessity of Inter-layer Similarity)**.** *Due to the presence of residual connections, significant similarity exists between inputs and outputs across adjacent layers. For sequences compressed according to Theorem 3, the storage upper bound for each layer can be expressed through inter-layer similarity operations.*

Specifically, the sequence length at layer l can be controlled by the upper bound at layer $l-1$ and their mutual similarity. Conversely, we can also constrain preceding layers based on the compressed final layer sequence length:

$$N_l = \min\left(\frac{N_{l-1}}{\mathbb{E}[Q(Sim(X^{(l)}, X^{(l-1)}))]}, N\right), \qquad N_{l-1} = \min\left(\frac{N_l}{\mathbb{E}[Q(Sim(X_{l-1}, X_l))]}, N\right) \tag{16}$$

where N_0 is bounded by the vocabulary size, N is the original sequence length, the upper bound N_L for the final layer also approaches the vocabulary size due to the vocabulary projection, and the quantization function is defined as:

$$Q(a_{ij}) = \begin{cases} 1 & \text{if } a_{ij} \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Here, $\mathbb{E}$ denotes the expectation operation, which can be interpreted as taking the maximum along the last dimension followed by averaging along the second-to-last dimension. This approach theoretically achieves the same error level as Theorem 3.

The detailed proof is provided in Appendix A.4. Although the analysis is simplified, empirical results confirm its general applicability. For example, when using inputs from selected layers [0,1,2,5,8,11,14,15,17,18,19,22] as subsequent layer inputs, Mistral-7B achieves 100% accuracy on the passkey retrieval task. This effect arises from inter-layer similarity, where residual connections propagate redundancy reduction while preserving information.

Since the similarity lower bound may approach zero—leading to intermediate states close to the original sequence length—we introduce a constant scaling factor $c \in [1, 2)$ for practical control. Specifically, the state size of layer i is set as $(c-1)N_{i-1}$. Empirically: 1. Beyond certain lengths, attention between new queries and stored states becomes sparse (e.g., NSA, MoBA). 2. Many tasks succeed with fixed-size states (e.g., SnapKV, GLA, GSA). 3. Inference typically operates within bounded state spaces. However, for large-vocabulary models (e.g., Qwen2.5, LLaMA3 with ∼128K tokens), even moderate scaling (e.g., $c = 1.5$ in a 32-layer model) leads to $\sim$ 82M states, necessitating further compression. Inspired by MVA's vocabulary decomposition and MoM's functional partitioning, we adopt multi-memory states to approximate Theorem 4 bounds while preserving fixed-size representations.

3.5 Multi-level State Decomposition and Enhanced Reading

Theorem 5 (Necessity of Multi-level State Decomposition and Enhanced Reading Mechanisms)**.** *Given a fixed-size storage space, the number of states that can be stored using a multi-level decomposition approach is $\prod_{i=1}^{m} C_i$, where m is the number of levels and C_i is the size of the vocabulary at level i. The storage mechanism follows:*

$$S_t^{k^{(i)}} = diag\left(1 - \bar{f}^{(i)}(k_t^{(i)})^\top\right) S_{t-1}^{k^{(i)}} + \bar{f}^{(i)}(k_t^{(i)})^\top k_t^{(i)}, \quad k_t^{(i+1)} = k_t^{(i)} - f^{(i)}(k_t^{(i)}) S_t^{k^{(i)}} \tag{18}$$

where f can be any function that amplifies correlations, such as the Softmax function. This storage approach is equivalent to quantization followed by storage, introducing an error. The average relative error decreases exponentially with the number of levels: $\prod_{i=1}^{m} \epsilon_i$, where ϵ_i is the error between the stored vector and its closest counterpart in the i-th level vocabulary. A key trade-off exists: increasing the number of levels reduces storage error but decreases computational efficiency due to the serial computation required between levels.

We emphasize that previous approaches employ overly simplistic reading mechanisms, typically using direct matrix multiplication between queries q and states. This simplicity constitutes a significant factor (besides storage limitations) contributing to the performance gap with Softmax Attention. Our work is the first to clearly identify enhanced reading mechanisms as crucial for improving linear attention and bridging this performance gap. This mechanism implements a hierarchical access pattern through multiple channels; by comparison, the GSA reading mechanism $\mathrm{Softmax}(q_t {S^k}_t){S^v}_t$ represents the simplest form of indirect reading. Our enhanced version replaces the Softmax with a sigmoid activation followed by learned transformations: $(\sigma(q_t {S^k}_t)W_\sigma){S^v}_t, \quad \text{where } \sigma(x) = \frac{1}{1+e^{-x}}$. Further extending to multiple reading channels: $(q_t W_r + \sigma(q_t {S^k}_t)W_\sigma){S^v}_t$. This approach, which is equivalent to MVA's first-order vocabulary case, demonstrates progressive performance improvement (Table 4). With multi-level vocabularies, multiple vocabularies interactions show even greater improvements over single-state approaches, underscoring the importance of balanced enhancement in both storage and reading capabilities.

3.6 Final Linear Model Update Rules

Integrating all five theoretical principles, we present the complete update formulation of the proposed linear model.

1. Initialization

Initial conditions:

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

Initial states:

$${S^k}_0^{(i)} = 0 \in \mathbb{R}^{m\times d}, n_0^{(i)} = 0 \in \mathbb{R}^{1\times m}, E_t^{(0)} = I_m, {S^k}_t^{(i)} = {S^{kv}}_t^{(i)}[..., : d_k], {S^v}_t^{(i)} = {S^{kv}}_t^{(i)}[..., d_v :],$$

2. Memory Assignment and Normalization.

Here, $f^{(i)}(k_t^{(i)})$ denotes the retrieval score of the current input with respect to stored states, and $n_t^{(i)}$ represents the effective accumulation count. During storage, we normalize the contribution of the current input against historical states using $\bar{f}^{(i)}(k_t^{(i)})$. This storage mechanism is primarily derived from Theory 1, while the state size control is governed by Theories 4 and 5.

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

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

3. Positional State Update and Residual Propagation (Theories 2 & 3).

The positional state ${S^p}_t^{(i)}$ decouples and compresses positional information, which is subsequently integrated with the position-free KV state into a unified compressed representation.

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

4. Cross-Level Coupling and Attention Statistics.

Using the upper bound derived in Theory 4, we determine the number of hierarchical levels required by Theory 5 as well as the state size at each level. Hierarchical decomposition is performed by computing the information loss between the current-level input and its projection onto the fixed-capacity stored state; this residual serves as the input of the next level $m_t^{(i+1)}$. After hierarchical partitioning, we compute the inter-level connection state $R_t^{(i+1)}$. During retrieval, correct memory routing is ensured via $a_t^{(i)}$. Multi-channel enhancements can be incorporated, e.g., $q_t W_r$ and $\sigma(q_t {S^k}_t^{(i)}) W_\sigma$. Finally, $c_t^{(i)}$ corresponds to the Taylor-approximation-based information compression component derived in Theory 3, which can be interpreted as an equivalent form of count and linear positional information, and may be replaced by more advanced data-dependent dynamic decay mechanisms.

$$m_t^{(i+1)} = m_t^{(i)} - f^{(i)}(k_t^{(i)}) {S^{kv}}_t^{(i)}. \tag{22}$$

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

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.

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

4 Experiments

In this paper, we explore experiments related to converting LLMs to linear models through weight inheritance, providing experimental support for each of the five theoretical principles presented in our methodology. In the final section, we integrate these principles into Path-optimized Linear Attention (PLA) and demonstrate the effectiveness of our approach through comprehensive experiments.

We use the lm-evaluation-harness Gao et al. (2024) tool and the LongBench dataset for evaluation. For fine-tuning, we utilize LoRA Hu et al. (2021) to achieve efficient parameter updates, significantly reducing computational resources. Detailed configurations are specified in each subsection. For baseline comparisons, we compare against state-of-the-art methods including MVA and GSA, as well as GLA, RetNet, and SUPRA Mercat et al. (2024), which were benchmarked in the GSA paper.

4.1 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 1 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

Table 1: Results for Theory 1 with different similarity thresholds

Method	Finetune 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

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

Method	Finetune 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 3 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.

Table 3: Results for Theory 4 with different layer scaling factors

Method	Finetune 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 A illustrates the evolution of KV cache length across layers as predicted by Theory 4.

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

Table 4 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 PLA approach building upon GSA by adding multi-channel reading and multi-state interaction, equivalent to incorporating vocabulary interaction(VI) into MVA.

Table 4: Results for Theory 5 with different reading mechanisms

Method	Finetune 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 (PLA: MVA + VI)	500M	40.0	39.4	26.2	24.7

4.2 PLA: Integrated Path-optimized Linear Attention

Building upon the experimental validation of individual theoretical principles, we now present the integrated PLA model that combines all five theoretical components into a unified framework. PLA also employs a two-level vocabulary decomposition, similar to GSA and MVA whose state is 128 in size, and we their basis by adding operations such as positional encoding decoupling, read enhancement, and lexicon interaction.

The experimental results demonstrate that PLA achieves state-of-the-art performance while maintaining competitive efficiency. The integrated approach successfully leverages all five theoretical principles to create a robust linear attention mechanism that narrows the performance gap with softmax attention.

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 PLA models.

Table 5: Comprehensive evaluation of PLA 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	+10B	78.3	47.5	78.1	34.4	60.3
PLA (Ours)	7B	+8B	78.5	47.2	78.3	42.1	61.3

Table 6: 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.

Model	ARC-c	ARC-e	HellaSwag	PIQA	WinoGrande	MMLU	AVG
RWKV7-2.9B[1]	48.7	81.0	76.4	79.7	72.8	55.0	68.93
GatedDeltaNet-1.3B[1]	38.4	71.2	55.7	72.2	57.4	32.4	54.55
Mamba-2-2.7B[1]	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-PLA (ft: 10B tokens)	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-PLA (ft: 10B tokens)	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 (ft: 12.5B tokens)	47.6	72.6	74.9	64.5	59.4	42.1	60.18
LLaMA-3-70B-PLA (ft: 10B tokens)	65.2	87.8	79.8	85.6	78.9	58.4	75.95

Table 7: Experimental results on long-context benchmarks, training efficiency comparison and passkey task

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
PLA	22.3	21.2	10.7

Method	Memory / Time
MetaLA	36,317 MiB / 75.08 s/it
GSA	37,619 MiB / 81.67 s/it
MVA w/ VD	40,096 MiB / 105.79 s/it
PLA w/ VD	41,278 MiB / 118.67 s/it

Model/passkey task	256	512	1024	2048	4096	8192
Qwen	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
PLA	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.9**

Table 8: Performance on LongBench tasks.

Model	SQA	MQA	Sum	FS	Code	Avg.
Qwen2.5-7B	28.60	28.71	17.85	69.51	58.60	40.65
PLA-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
PLA-14B-Qwen (Ours)	34.32	31.22	17.02	73.96	63.58	44.02

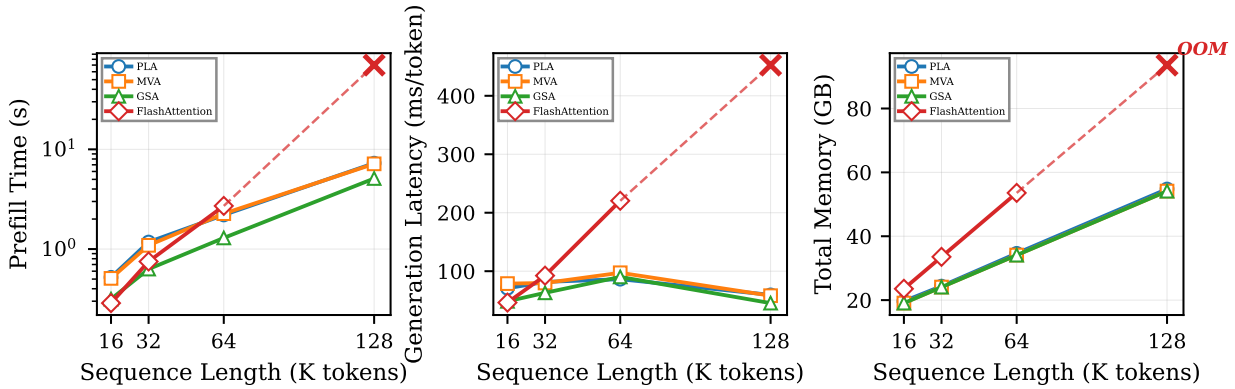

Figure 2: Inference efficiency comparison of PLA, MVA, GSA, and FlashAttention under different sequence lengths. OOM indicates out-of-memory errors.

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.

Common-Sense Reasoning

Table 5 reports the performance of PLA based on Mistral weights on common-sense reasoning benchmarks. Our method achieves superior performance compared to GLA, GSA, and MVA.

To further demonstrate the scalability and generality of PLA, we fine-tune Qwen2.5-3B, Qwen2.5-14B, and LLaMA3-70B models. With only 10B fine-tuning tokens, PLA is able to recover nearly 95% of the original Transformer performance on common-sense reasoning tasks, as shown in Table 6.

Long-Sequence Tasks

Tables 7 and 8 present the performance of PLA models with different parameter scales on LongBench and various long-sequence tasks.

Compared to the original Attention-based models without length extension, PLA surpasses the original models on certain long-sequence benchmarks, demonstrating its ability to improve long-context modeling capacity.

Retrieval Tasks

Table 7 also reports performance on the Passkey retrieval task. With 8K-length fine-tuning, PLA is able to almost perfectly solve the 8K-length retrieval task, a capability that previous linear attention models struggled to achieve.

Efficiency Analysis

Figure 4.2 compares the inference speed of two-level PLA with Transformer and GSA. PLA achieves a speed close to GSA under the same state size, while significantly outperforming GSA in accuracy.

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, PLA 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

Kumar Krishna Agrawal, Long Lian, Longchao Liu, Natalia Harguindeguy, Boyi Li, Alexander Bick, Maggie Chung, Trevor Darrell, and Adam Yala. Atlas: Multi-scale attention improves long context image modeling, 2025. URL https://arxiv.org/abs/2503.12355.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Ruijie Zhu, Yiran Zhong, Yu Qiao, Jibin Wu, Bo Xu, and Guoqi Li. Metala: Unified optimal linear approximation to softmax attention map, 2024. URL https://arxiv.org/abs/2411.10741.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X.

Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories, 2025. URL https://arxiv.org/abs/2502.13685.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

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. URL https://openreview.net/forum?id=FJFVmeXusW.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

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)*, pp. 5961–5971, October 2023.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9099–9117. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hua22a.html.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

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. ISSN 1049-331X. doi: 10.1145/3747588. URL https://doi.org/10.1145/3747588. Just Accepted.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

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 (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

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. URL https://openreview.net/forum?id=8tKjqqMM5z.

Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models, 2024. URL https://arxiv.org/abs/2405.06640.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models, 2024a. URL https://arxiv.org/abs/2401.04658.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion, 2024b. URL https://arxiv.org/abs/2404.07904.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025. URL https://arxiv.org/abs/2407.04620.

Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=WSUO2uRDPc.

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. URL https://openreview.net/forum?id=ou0AsNbXVG.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=bTHFrqhASY.

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 (eds.), *International Conference on Representation Learning*, volume 2024, pp. 21875–21895, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2024a. URL https://arxiv.org/abs/2412.06464.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024b. URL https://arxiv.org/abs/2312.06635.

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

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling, 2024. URL https://arxiv.org/abs/2409.07146.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. 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 (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

A Appendix

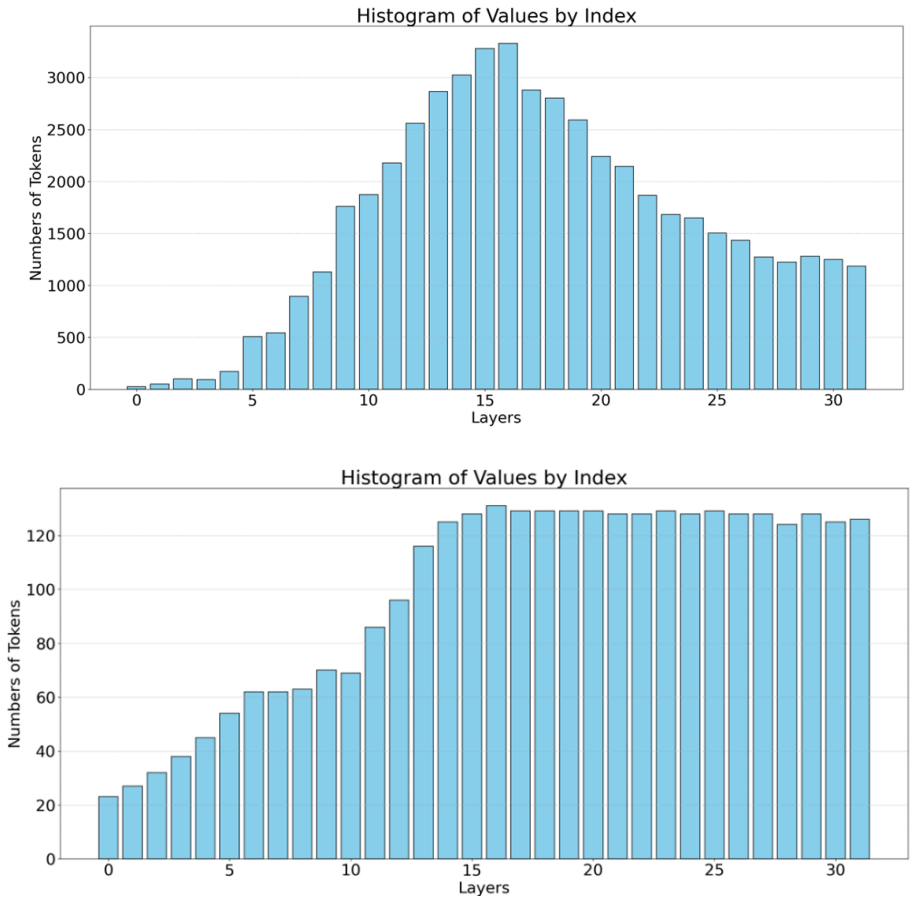

Figure 3: The task uses passkey task. The above figure shows the result of KV cache size per layer obtained by applying Theory 1 for compression and then the following figure shows the result obtained by adding Theory 4 for interlayer similarity, after interlayer similarity the KV cache of the subsequent layers is controlled.

A.1 Proof of Theorem 1

A.1.1 Proof of the storage 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 $\boldsymbol{C} \in \mathbb{R}^{t\times 1}$.

The number of distinct row vectors is bounded as follows: for a vector of dimension $d_k + d_v$, each entry admits 2^b possible values. Since dimensions are independent, the total number of distinct row types is

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

Table 9: 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	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	$O\left(\frac{n}{\text{base}^{2j/d}}\right)$	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

During sequence growth, when a new vector $\boldsymbol{s}_t^{KV}$ is identical to an existing $\boldsymbol{s}_m^{KV}$, we simply increment the counter:

$$\boldsymbol{C}_m \leftarrow \boldsymbol{C}_m + 1.$$

The corresponding attention computation becomes

$$\boldsymbol{o}_t = \exp\big(\boldsymbol{q}_t \boldsymbol{S}^K\big)\,\big(\boldsymbol{S}^V \odot \boldsymbol{C}\big),$$

which is clearly equivalent to

$$\boldsymbol{o}_t = \exp\big(\boldsymbol{q}_t \boldsymbol{K}\big)\,\boldsymbol{V}.$$

A.1.2 Equivalence to the Delta Rule

Consider the unique-filtered sequence as the state $\boldsymbol{S}_t^{(K)}$. 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.

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 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.2 Proof Sketch of Theorem 2

The proof builds upon the storage bound established in Theorem 1. For the first layer input $X^{(1)} \in \mathbb{R}^{t \times d}$, each row vector $x_i^{(1)}$ corresponds to a token embedding from the vocabulary V_T. Since tokenization maps each token to a unique embedding, the number of distinct vectors in $X^{(1)}$ is bounded by $|V_T|$.

After positional encoding $P \in \mathbb{R}^{t \times d}$ is applied, the encoded input becomes:

$$\tilde{X}^{(1)} = f_p(X^{(1)}, P)$$

where f_p represents the positional encoding function (e.g., addition for absolute positional encoding, or rotary multiplication for RoPE).

The key insight is that we can decouple the positional information by storing: 1. The base token embeddings $E \in \mathbb{R}^{V_T \times d}$ (vocabulary embeddings)

2. The positional offsets $\Delta P \in \mathbb{R}^{t \times d}$

3. An index mapping $I_p \in \mathbb{N}^t$ from sequence positions to vocabulary indices

The storage requirement thus becomes:

$$\text{Storage} = \underbrace{V_T \cdot d \cdot b}_{\text{embeddings}} + \underbrace{t \cdot b}_{\text{positional offsets}}$$

For the first layer KV cache, the bound $C = V_T$ emerges because the number of distinct key-value pairs is constrained by the vocabulary size when positional information is properly decoupled. The positional offsets can be compressed using techniques discussed in Theorem 3, while the indices represent the unbounded growth component.

The lossless nature of this compression for the first layer follows from the invertibility of the decoupling operation: given E, ΔP, and I_p, we can perfectly reconstruct $\tilde{X}^{(1)}$.

For subsequent layers, the type diversity increases due to token mixing operations, but decreases toward the final layer due to the vocabulary projection. This creates the characteristic "increase-then-decrease" pattern observed empirically.

The positional decoupling theorem establishes a fundamental connection between the discrete nature of language modeling (through tokenization) and the continuous representations used in transformer layers. This bridges the gap between information-theoretic bounds based on vocabulary size and practical compression algorithms for transformer inference.

Corollary 1. *For models employing subword tokenization with merge operations, the effective vocabulary size V_T^{eff} that bounds the first-layer distinct types may be larger than the nominal vocabulary size, but remains finite and typically grows sublinearly with training data size.*

A.3 Proof of Theorem 3

We begin with the Taylor expansion of the cosine function:

$$\cos\theta = 1 - \frac{\theta^2}{2} + \frac{\theta^4}{4!} + \sum_{i=6,\text{even}}^{\infty} \frac{\theta^i}{i!} \tag{25}$$

The original attention computation can be expressed as:

$$e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}} \sum_{s} e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1)+\sum_{j=0}^{d}\frac{(-1)^{j+1}}{\sqrt{d}} q_{nj}\cdot k_{s,(j+\frac{d}{2})\%d}\cdot(\sin[(n-s)\theta_j])} \tag{26}$$

Let us define the residual term:

$$r(\theta_1,\ldots,\theta_{\frac{d}{2}-1}) = \frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1) + \sum_{j=0}^{d}\frac{(-1)^{j+1}}{\sqrt{d}} q_{nj}\cdot k_{s,(j+\frac{d}{2})\%d}\cdot(\sin[(n-s)\theta_j]) \tag{27}$$

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

$$\cos\theta_j = 1 - \frac{\theta_j^2}{2} + O(\theta_j^4) \tag{28}$$

$$\sin\theta_j = \theta_j + O(\theta_j^3) \tag{29}$$

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 $e^{r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})}$:

$$e^{r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})} = 1 + r(\theta_1,\ldots,\theta_{\frac{d}{2}-1}) + O(r(\theta_1,\ldots,\theta_{\frac{d}{2}-1})^2) \tag{30}$$

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:

$$\begin{aligned} e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}} \sum_{s}\Bigg(1 &+ \frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1) \\ &+ \frac{1}{\sqrt{d}}\sum_{j=0}^{d}(-1)^{j+1} q_{nj}\cdot k_{s,(j+(-1)^j)}\cdot(\sin[(n-s)\theta_j])\Bigg) \end{aligned} \tag{31}$$

Alternatively, we can use the formulation:

$$\begin{aligned} e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} x_{nj}x_{sj}} \sum_{s}\Bigg(1 &+ \frac{1}{\sqrt{d}}\sum_{j=0}^{d} x_{nj}x_{sj}\big(\cos[(n-s)\theta_j]-1\big) \\ &+ \frac{1}{\sqrt{d}}\sum_{j=0}^{d}(-1)^{j+1} x_{nj}x_{s,(j+(-1)^j)}\big(\sin[(n-s)\theta_j]-(n-s)\theta_j\big)\Bigg) \end{aligned} \tag{32}$$

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:

$$e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d} q_{nj}\cdot k_{sj}} \sum_{s}\prod_{p=0}^{d/m}\left(1+\frac{1}{\sqrt{d}}\sum_{j=p\cdot m}^{p\cdot m+m} q_{nj}\cdot k_{sj}\cdot(\cos[(n-s)\theta_j]-1)\right.$$
$$\left.+\frac{1}{\sqrt{d}}\sum_{j=p\cdot m}^{p\cdot m+m}(-1)^{j+1} q_{nj}\cdot k_{s,(j+(-1)^j)}\cdot(\sin[(n-s)\theta_j])\right) \tag{33}$$

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

$$(e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d}(q_{nj}\cos(n\theta_j)-q_{n,(j+\frac{d}{2})\%d}\cdot\sin[n\theta_j])\cdot k_{sj}})$$
$$\cdot\sum_{s} e^{\frac{1}{\sqrt{d}}\sum_{j=0}^{d}(q_{nj}\cos(n\theta_j)-q_{n,(j+\frac{d}{2})\%d}\cdot\sin[n\theta_j])\cdot(k_{sj}(\cos(s\theta_j)-1)-k_{s,(j+\frac{d}{2})\%d}\cdot\sin[s\theta_j])} \tag{34}$$

Applying the same Taylor expansion yields:

$$e^{f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top}\sum_{s}\left(1+f_p(\mathbf{q}_n,\mathbf{p}(n))f_p(\mathbf{k}_s,\mathbf{p}(s))^\top-f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top\right) \tag{35}$$

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

$$e^{f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top}\left(t+f_p(\mathbf{q}_n,\mathbf{p}(n))f_p(\mathbf{k}_s,\mathbf{p}^{(t)}(s))^\top-f_p(\mathbf{q}_n,\mathbf{p}(n))\mathbf{k}_s^\top\right) \tag{36}$$

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

Theorem 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.4 Proof Sketch of Theorem 4

Given the complexity of inter-layer dynamics, we provide a simplified theoretical analysis of hierarchical similarity. For the input at layer l:

$$X^{(l)} = X^{(l-1)} + A(X^{(l-1)}) + F(X^{(l-1)} + A(X^{(l-1)})) \tag{37}$$

where $A(\cdot)$ represents the attention operation and $F(\cdot)$ represents the MLP operation.

Since our primary objective is to control the relationship between state counts across layers based on similarity, and MLP operations generally preserve state diversity, we focus our analysis on the similarity between attention outputs and their inputs. Specifically, we examine:

$$X^{(l)} = X^{(l-1)} + A(X^{(l-1)}) \tag{38}$$

The cosine similarity between $X^{(l)}$ and $X^{(l-1)}$ for position m can be expressed as:

$$s_{mm} = \frac{1}{A}\left(1+\sum_{j=0}^{m}\frac{e^{x_m^{(l-1)}W_{qk}^\top x_j^{(l-1)\top}}\, x_m^{(l-1)}W_{vo}^\top x_j^{(l-1)\top}}{\sum_{j=0}^{m} e^{x_m^{(l-1)}W_{qk}^\top x_j^{(l-1)\top}}}\right) \tag{39}$$

where A is a normalization coefficient. Note that s_{mm} is not necessarily the maximum similarity value. In most cases, when $x_m^{(l-1)} W_{qk}^\top x_j^{(l-1)\top}$ is large (indicating high token relevance), $x_m^{(l-1)} W_{vo}^\top x_j^{(l-1)\top}$ also contributes significantly to the sum, suggesting limited changes in state diversity after attention processing.

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 set the state size for layer i as $(c-1)N_{i-1}$, where $c \in [1, 2)$. This formulation allows complex tasks to utilize larger thinking spaces while maintaining efficiency for simpler tasks. Our experiments show that earlier layers typically require larger c values, while later layers can use smaller values. For layers beyond the midpoint ($L/2$), we set $c = 1$ to optimize storage efficiency.

As shown in Figure A, 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:

$$\text{Storage}(X_{<t}) = \text{Softmax}(CX_{<t}^\top)X_{<t} \tag{40}$$

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 Theorem 4 while maintaining fixed-size representations.

Corollary 2. *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.5 Proof of Theorem 5

Let $X \in \mathbb{R}^{n \times d}$ be a sequence with $n \gg 1$, and let $\{C^{(i)}\}_{i=1}^c$ be a set of vocabulary matrices where each $C^{(i)} \in \mathbb{R}^{m \times d}$ contains m prototype vectors. The multi-level vocabulary decomposition represents each element $x_j \in X$ as:

$$\hat{x}_j = \sum_{i=1}^{c} C^{(i)}_{k_j^{(i)}} \tag{41}$$

where $k_j^{(i)} \in \{1, 2, \ldots, m\}$ is the index selected from the i-th vocabulary for representing x_j.

Then:

1. The maximum number of distinct vectors that can be represented is m^c
2. The approximation error for an optimal decomposition satisfies:

$$\mathbb{E}[\|x_j - \hat{x}_j\|_2^2] \leq \prod_{i=1}^{c} \epsilon_i \tag{42}$$

where ϵ_i is the average quantization error at level i

Proof. **Part 1: Representation Capacity**

The representation capacity follows from combinatorial considerations. For each vector x_j, we select one prototype from each of the c vocabularies. Since each vocabulary contains m prototypes, the total number of possible combinations is:

$$\text{Total combinations} = \underbrace{m \times m \times \cdots \times m}_{c \text{ times}} = m^c \tag{43}$$

Each unique combination of indices $(k_j^{(1)}, k_j^{(2)}, \ldots, k_j^{(c)})$ produces a unique sum:

$$\hat{x}_j = C^{(1)}_{k_j^{(1)}} + C^{(2)}_{k_j^{(2)}} + \cdots + C^{(c)}_{k_j^{(c)}} \tag{44}$$

Assuming linear independence among the vocabulary vectors across different levels, these sums are distinct. Therefore, the maximum number of distinct representable vectors is exactly m^c.

Part 2: Error Analysis

We analyze the error propagation through the multi-level decomposition. Let us define the residual at each level:

$$r_j^{(0)} = x_j \tag{45}$$

$$r_j^{(i)} = r_j^{(i-1)} - C^{(i)}_{k_j^{(i)}} \quad \text{for } i = 1, 2, \ldots, c \tag{46}$$

The final approximation is:

$$\hat{x}_j = \sum_{i=1}^{c} C^{(i)}_{k_j^{(i)}} = x_j - r_j^{(c)} \tag{47}$$

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

Now, consider the optimal index selection at each level. We choose $k_j^{(i)}$ to minimize the residual norm:

$$k_j^{(i)} = \arg\min_{k \in \{1,\ldots,m\}} \|r_j^{(i-1)} - C_k^{(i)}\|_2 \tag{48}$$

Let ϵ_i be the average quantization error at level i:

$$\epsilon_i = \mathbb{E}[\min_{k \in \{1,\ldots,m\}} \|r_j^{(i-1)} - C_k^{(i)}\|_2^2] \tag{49}$$

Assuming the residuals and vocabulary vectors are appropriately normalized, we can bound the error propagation. Using the triangle inequality and the optimality of our index selection:

$$\|r_j^{(i)}\|_2 = \|r_j^{(i-1)} - C^{(i)}_{k_j^{(i)}}\|_2 \tag{50}$$

$$\leq \|r_j^{(i-1)}\|_2 \cdot \min_{k\in\{1,\ldots,m\}} \left\| \frac{r_j^{(i-1)}}{\|r_j^{(i-1)}\|_2} - \frac{C_k^{(i)}}{\|r_j^{(i-1)}\|_2} \right\|_2 \tag{51}$$

For well-designed vocabularies that cover the relevant direction space, the directional error term is bounded. In the worst case, we have:

$$\|r_j^{(i)}\|_2 \leq \|r_j^{(i-1)}\|_2 \cdot \delta_i \tag{52}$$

where δ_i represents the maximum angular error at level i. Applying this recursively:

$$\|r_j^{(c)}\|_2 \leq \|x_j\|_2 \cdot \prod_{i=1}^{c} \delta_i \tag{53}$$

For the mean squared error, under appropriate assumptions about the distribution of residuals and the vocabulary coverage:

$$\mathbb{E}[\|r_j^{(c)}\|_2^2] \leq \mathbb{E}[\|x_j\|_2^2] \cdot \prod_{i=1}^{c} \epsilon_i \tag{54}$$

where $\epsilon_i = \mathbb{E}[\delta_i^2]$ is the expected squared angular error at level i.

The product structure $\prod_{i=1}^{c} \epsilon_i$ demonstrates the exponential error reduction with increasing levels, provided that each $\epsilon_i < 1$. □

Corollary 3 (Trade-off between Representation Capacity and Error)**.** *For a fixed total budget of $B = m \cdot c$ parameters, the optimal balance between m and c that minimizes the approximation error while maximizing representation capacity satisfies:*

$$m^* \approx c^* \approx \sqrt{B} \tag{55}$$

This provides the optimal trade-off point where m^c is maximized subject to the constraint $m \cdot c = B$.

Proof. We maximize the representation capacity m^c subject to the constraint $m \cdot c = B$. Taking logarithms:

$$\log(\text{capacity}) = c \log m = c \log\left(\frac{B}{c}\right) \tag{56}$$

Differentiating with respect to c:

$$\frac{d}{dc}\left[c \log\left(\frac{B}{c}\right)\right] = \log\left(\frac{B}{c}\right) - 1 \tag{57}$$

Setting the derivative to zero gives:

$$\log\left(\frac{B}{c}\right) = 1 \quad \Rightarrow \quad \frac{B}{c} = e \quad \Rightarrow \quad c = \frac{B}{e} \tag{58}$$

Thus, $m = \frac{B}{c} = e$, and the optimal values are $m^* \approx c^* \approx \sqrt{B}$ when considering integer constraints and practical implementation factors. □

The multi-level vocabulary decomposition provides an exponential increase in representation capacity (m^c) compared to a single-level approach (m), while simultaneously achieving exponential error reduction. This theoretical foundation justifies the effectiveness of hierarchical representations in compression applications.

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.

A.6 Enhanced Reading Mechanisms and unified formula

Theorem 6 (Necessity of Enhanced Reading Mechanisms)**.** *We emphasize that previous approaches employ overly simplistic reading mechanisms, typically using direct matrix multiplication between queries q and states. This simplicity constitutes a significant factor (besides storage limitations) contributing to the performance gap with Softmax Attention. Our work is the first to clearly identify enhanced reading mechanisms as crucial for improving linear attention and bridging this performance gap.*

We propose a sophisticated reading approach:

$$R_t^{(i+1)}\left[f(k_t^{(i)}), f^{(i+1)}(k_t^{(i)})\right] = 1 \tag{59}$$

$$d_t^{(i)} = q_t {S^k}_t^{(i)\top} \tag{60}$$

$$e_t^{(i)} = \frac{\exp\left(d_t^{(i)}\right)}{\sum_{j=0}^{d-1}\exp\left(d_{tj}^{(i)}\right)} \tag{61}$$

$$a_t^{(i)} = e_t^{(i)} R_t^{(i)\top} \tag{62}$$

$$b_t^{(i)} = \frac{\exp\left(\sum_i \ln(a_t^{(i)})\right)}{a_t^{(i)}} \tag{63}$$

$$c_t^{(i)} = \left(n_t^{(i)} + q_t {S^p}_t^{(i)\top}\right) \tag{64}$$

$$T_t^{(i)} = R_t^{(i)}\left({S^v}_t^{(i)} \odot e_t^{(i)\top} \odot c_t^{(i)\top}\right) \tag{65}$$

$$o_t = \sum_i o_t^{(i)} = \sum_i \frac{b_t^{(i)}}{b_t^{(i)} \odot e_t^{(i)} \odot c_t^{(i)}} T_t^{(i)} \tag{66}$$

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:

$$(\sigma(q_t {S^k}_t)W_\sigma){S^v}_t, \quad \text{where } \sigma(x) = \frac{1}{1+e^{-x}} \tag{67}$$

Further extending to multiple reading channels:

$$(q_t W_r + \sigma(q_t {S^k}_t)W_\sigma){S^v}_t \tag{68}$$

This approach equivalent to MVA's first-order vocabulary case demonstrates progressive performance improvement (Table 4). With multi-level vocabularies, similar enhancements using Softmax, perceptron, and multi-channel mechanisms show even greater improvements over single-state approaches, underscoring the importance of balanced enhancement in both storage and reading capabilities.

The update formulas in the main text we have improved on the GSA and MVA by combining the five theories. And the update formulas are as follows if we start from the five theories only and do not use the current case of the excellent mechanism(e.g. gating) instead, and the following update method can also achieve similar performance. **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:

$$\begin{aligned} f^{(i)}(k_t^{(i)}) &= F(S^{k}{}_{t-1}^{(i)} k_t^{(i)\top})^{\top}, \quad F(x) = \begin{cases} 1 & \text{if } x_i \text{ is maximum} \\ 0 & \text{otherwise} \end{cases} \text{ or } \mathrm{Softmax}(x) \\ n_t^{(i)} &= n_{t-1}^{(i)} + f^{(i)}(k_t^{(i)}), \quad \bar{f}^{(i)}(k_t^{(i)}) = \frac{f^{(i)}(k_t^{(i)})}{\max(n_t^{(i)}, 1)} \end{aligned} \tag{69}$$

$$\begin{aligned} S^{k}{}_t^{(i)} &= \mathrm{diag}\left(1 - \bar{f}^{(i)}(k_t^{(i)})^{\top}\right) S^{k}{}_{t-1}^{(i)} + \bar{f}^{(i)}(k_t^{(i)})^{\top} k_t^{(i)} \\ S^{v}{}_t^{(i)} &= \mathrm{diag}\left(1 - \bar{f}^{(i)}(k_t^{(i)})^{\top}\right) S^{v}{}_{t-1}^{(i)} + \bar{f}^{(i)}(k_t^{(i)})^{\top} v_t^{(i)} \end{aligned} \quad \text{(Theory 1)} \tag{70}$$

$$\begin{aligned} S^{p}{}_t^{(i)} &= S^{p}{}_{t-1}^{(i)} + f^{(i)}(k_t^{(i)})^{\top}(k_{pt} - k_t^{(0)}) \\ k_t^{(i+1)} &= k_t^{(i)} - f^{(i)}(k_t^{(i)}) S^{k}{}_t^{(i)} \\ v_t^{(i+1)} &= v_t^{(i)} - f^{(i)}(k_t^{(i)}) S^{v}{}_t^{(i)} \end{aligned} \quad \text{(Theories 2 \& 3)} \tag{71}$$

$$R_t^{(i+1)}\left[f(k_t^{(i)}), f^{(i+1)}(k_t^{(i)})\right] = 1 \quad \text{(Theory 5)} \tag{72}$$

$$\begin{aligned} d_t^{(i)} &= q_t S^{k}{}_t^{(i)\top}, \quad e_t^{(i)} = \mathrm{softmax}(d_t^{(i)}) \\ a_t^{(i)} &= e_t^{(i)} R_t^{(i)\top}, \quad b_t^{(i)} = \frac{\exp\left(\sum_i \ln(a_t^{(i)})\right)}{a_t^{(i)}} \end{aligned} \quad \text{(Theories 4 \& 5)} \tag{73}$$

$$\begin{aligned} c_t^{(i)} &= n_t^{(i)} + q_t S^{p}{}_t^{(i)\top} \\ T_t^{(i)} &= R_t^{(i)}\left(S^{v}{}_t^{(i)} \odot e_t^{(i)\top} \odot c_t^{(i)\top}\right) \\ o_t &= \sum_i \frac{b_t^{(i)}}{b_t^{(i)} \odot e_t^{(i)} \odot c_t^{(i)}} T_t^{(i)} \end{aligned} \quad \text{(Theories 4 \& 5)} \tag{74}$$

Chunk-wise Parallel Form

For minibatch processing:

$$R_t^{(i+1)} = R_{t-1}^{(i+1)} + \left(f^{(i)}(K^{(i)})\right)^\top \left(f^{(i+1)}(K^{(i+1)})\right) \tag{75}$$

$$R_t^{(i+1)} = R_{t-1}^{(i+1)} + (Y - R_{t-1}^{(i+1)}) \odot \left(f^{(i)}(K^{(i)})\right)^\top \left(f^{(i+1)}(K^{(i+1)})\right) \tag{76}$$

$$N^{(i)} = \text{CumSum}\left(f^{(i)}(K^{(i)})\right) \tag{77}$$

$$\bar{f}^{(i)}(K^{(i)}) = \frac{f^{(i)}(K^{(i)})}{N^{(i)}} \tag{78}$$

$$D^{(i)} = \text{GLA}^\top\left(Q, K^{(i)}, \bar{f}^{(i)}(K^{(i)}), 1 - \bar{f}^{(i)}(K^{(i)})\right) \tag{79}$$

$$E^{(i)} = \text{softmax}(D^{(i)}) \tag{80}$$

$$A^{(i)} = E^{(i)}(R^{(i)})^\top \tag{81}$$

$$C^{(i)} = \text{GLA}^\top\left(Q, f_p(K^{(i)}, P) - K^{(i)}, \bar{f}^{(i)}(K^{(i)}), 1 - \bar{f}^{(i)}(K^{(i)})\right) + N^{(i)} \tag{82}$$

$$T^{(i)} = R^{(i)}\text{GLA}\left(C^{(i)} \odot E^{(i)}, \bar{f}^{(i)}(K^{(i)}), V^{(i)}, 1 - \bar{f}^{(i)}(K^{(i)})\right) \tag{83}$$

$$O^{(i)} = \frac{B^{(i)}}{B^{(i)} \odot E^{(i)} \odot C^{(i)}} \odot T^{(i)} \tag{84}$$

$$O = \sum_i O^{(i)} \tag{85}$$

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

1. **Error Analysis**: The exponential error reduction $\prod_{i=1}^m \epsilon_i$ follows from the chain rule of differentiation applied to the composition of quantization operations at each level. Each level introduces an independent quantization error, and the total error becomes the product of individual errors.

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. □

Theorem 5 and Theorem 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 4. *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.*