# RSA: RECURSIVE SPARSE ATTENTION WITH HIERARCHICAL DEEP–SHALLOW MEMORY AND SPARSE ACTIVATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Due to their linear complexity, linear sequence models excel in long-context modeling but underperform on tasks requiring long-range reasoning and retrieval compared to standard Transformers. Recent efforts using gating, delta rules, or multi-state models still mainly focus on short-range information, limiting effectiveness in complex long-sequence tasks due to insufficient information utilization and limited effective memory capacity. To address this, we propose a shallow-deep memory architecture in which multiple memory states are structurally interconnected: shallow memory stores coarse-grained representations and deeper memory stores residual information. The resulting hierarchy has a multiplicative nominal joint-code space, while a correlation-guided sparse readout retrieves relevant information from the stored states. Co-designing memory storage and retrieval, we introduce Recursive Sparse Attention (RSA). Experiments from 360M to 14B parameters show competitive results with strong linear and Transformer baselines on the evaluated common-sense, retrieval, and long-context benchmarks. We further report controlled continued-pretraining and supervised fine-tuning comparisons, from-scratch long-context retrieval, and diagnostics of redundancy across memory levels.

## 1 INTRODUCTION

Transformer Vaswani et al. (2017) has become the dominant architecture for generative models, powering large language models Touvron et al. (2023); Jiang et al. (2023); DeepSeek-AI et al. (2024); Dubey et al. (2024); Yang et al. (2025a); Brown et al. (2020); Raffel et al. (2020); Zhang et al. (2022). Its block-wise matrix operations Dao (2023b;a) fully utilize modern GPUs, and combined with autoregressive modeling, have significantly advanced AI development. However, standard Attention requires storing the entire sequence, leading to two critical inefficiencies in long-sequence tasks: quadratic complexity during training and unbounded KV Cache growth during inference. Moreover, at large sequence lengths, fully-connected standard Attention introduces more noise than sparse connections, which under limited data and training resources can yield suboptimal performance Tay et al. (2022). To address these issues, numerous efficient Attention variants have emerged, including sparse Attention Ge et al. (2024); Xiao et al. (2024b); Child et al. (2019); Beltagy et al. (2020), linear Attention Katharopoulos et al. (2020b); Qin et al. (2024b); Yang et al. (2024); Katharopoulos et al. (2020a); Choromanski et al. (2021), and State Space Models (SSMs) Dao & Gu (2024); Gu et al. (2022); Gu & Dao (2023).

**Sparse Attention** methods select a subset of the sequence for Attention computation. We categorize them into static and dynamic routing approaches. **Static methods**—e.g., Swin Transformer Liu et al. (2021), BigBird Zaheer et al. (2020), and Attention-sink Xiao et al. (2024b)—preselect fixed positions such as local windows, shifted windows, global tokens, or random positions. While these improve efficiency, their fixed patterns may discard information outside selected positions, causing unacceptable degradation on certain tasks. **Dynamic routing methods**, such as Routing Transformer Roy et al. (2021), MoBA Lu et al. (2025), and NSA Yuan et al. (2025), retrieve the most relevant key-value pairs on-the-fly. NSA further incorporates a compression branch for global coarse information. Dynamic routing outperforms static approaches because it can recover the static selection pattern when that is optimal. However, its drawback is the extra computational overhead



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

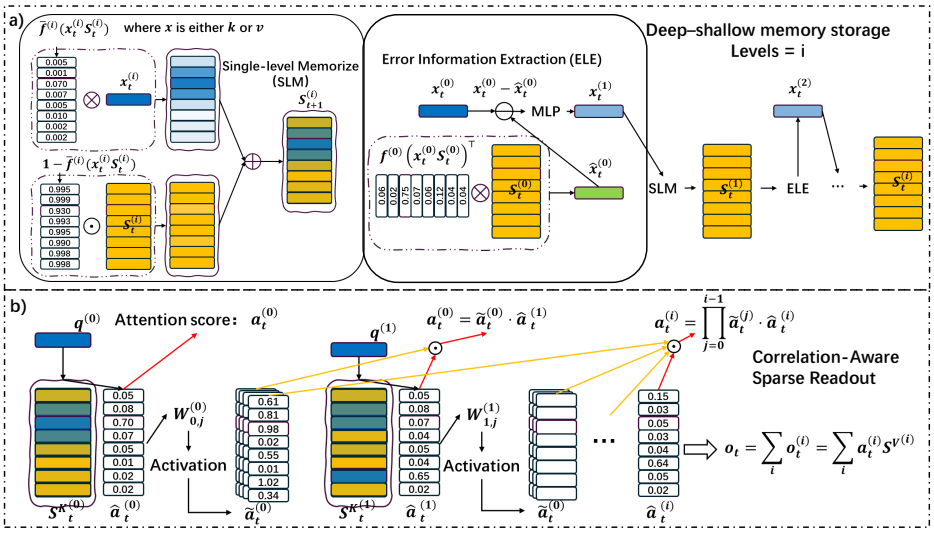


Figure 1: (a) **Deep–shallow memory storage.** At time step $t$, the input is compressed into the shallow state, and the residual error with respect to this state is recursively passed to deeper states. Here, $f^{(i)}$ and $\bar{f}^{(i)}$ are special nonlinear functions; for details, please refer to the text. (b) **Memory readout.** By using shallow attention scores to adaptively sparsify deep attention scores, we can localize relevant information while filtering out noise. $\otimes$: matmul, $\odot$: element wise multiplication.

introduced by the routing process, which trades off with the actual Attention computation and prevents linear complexity. It also requires **storing the full fine-grained KV Cache** Xiao et al. (2024a); Zhang et al. (2024).

**Linear model** methods maintain a fixed-size state and improve performance via various update mechanisms. Early Linear Attention used kernel functions to obtain a fixed KV state Katharopoulos et al. (2020a); Choromanski et al. (2021), but suffered from overly dispersed attention. Subsequent works, such as GLA Yang et al. (2024), introduced gating to sharpen updates, focusing on nearby tokens like sliding-window attention. However, dynamic decay limits their performance on long-sequence and retrieval tasks Qin et al. (2024a). Recently, Delta-rule-based models Schlag et al. (2021); Behrouz et al. (2024); Yang et al. (2025b) remove redundant information during updates, improving state utilization. Multi-state designs like MVA Wang et al. (2025b), MoM Du et al. (2025), and PLA Wang et al. (2025a) further narrow the gap with standard attention, even surpassing it on short sequences. Nevertheless, current linear models still underperform Transformers on complex tasks and lack extrapolation capabilities for retrieval Sun et al. (2024); Luo et al. (2024).

To address these issues, we unify and enhance large-capacity storage mechanisms by proposing a **deep-shallow memory**: shallow memory stores coarse information, while deep memory stores residuals relative to the shallow level. The nominal joint-code space grows multiplicatively across levels, and an MLP refines the residual passed to the next level. We then design a sparse readout mechanism that retrieves relevant stored summaries while suppressing noise: shallow-level relevance guides routing to deeper residual memory, while weak correlations are compressed or removed. Combining these, we introduce **Recursive Sparse Attention (RSA)**. RSA combines fixed-size linear memory with dynamic sparse reading and retains linear sequence-length complexity. Across the tasks reported in our tables, RSA is competitive with or outperforms the evaluated Transformer baselines on common-sense reasoning, LongBench, and retrieval benchmarks, while also improving over strong linear baselines. These task-bounded results provide evidence that RSA remains effective as model size increases, without claiming Transformer-level parity beyond the evaluated settings.

## 2 Related Work and Preliminaries

### 2.1 Standard Attention

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

comparison with linear attention models that update states sequentially, we also express SA in temporal form. Let $q_t \in \mathbb{R}^{1\times d_k}$, $k_t \in \mathbb{R}^{1\times d_k}$, and $v_t \in \mathbb{R}^{1\times d_v}$ denote the query, key, and value at time step $t$. Standard Attention (SA) and the output at time $t$ are computed as:

$$O = \mathrm{SA}(Q,K,V) = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V, \qquad o_t = \sum_{i=1}^{t} \frac{\exp(q_t k_i^\top/\sqrt{d_k})}{\sum_{j=1}^{t}\exp(q_t k_j^\top/\sqrt{d_k})}\, v_i.$$

where $M$ denotes the causal mask, $d_k$ is the normalization dimension, $K_t = k_{1:t} \in \mathbb{R}^{t\times d_k}$ and $V_t = v_{1:t}$ are the complete key and value sequences up to step $t$.

### 2.2 Sparse Attention: Fixed-Pattern vs. Dynamic Routing

Sparse attention reduces complexity by restricting the key positions accessible to each query. Depending on whether the selection depends on input content, sparse attention is categorized into *fixed-pattern* and *dynamically routed* methods.

In general form, sparse attention is written as

$$\tilde{K} = \mathrm{Loc}(K, i_k), \tilde{V} = \mathrm{Loc}(V, i_k), \mathrm{Output} = \mathrm{Softmax}(Q\tilde{K})\tilde{V},$$

where $\mathrm{Loc}(\cdot)$ denotes a selection operator and $i_k$ represents the selected key indices.

In **fixed-pattern sparse attention**, $i_k$ is independent of $Q$ and $K$, determined by predefined rules. Representative examples include: a) Sliding-window attention, where $i_k \in (i - \frac{s}{2}, i + \frac{s}{2})$; b) Global tokens or attention sinks, where $i_k \in (0, s)$; c) Swin Transformer, which alternates between local and shifted window attention. While these methods offer clear computational benefits, their fixed patterns may cause information loss in long-range dependencies or unevenly distributed salient information.

**Dynamically routed sparse attention**, in contrast, determines $i_k$ via content-adaptive mechanisms based on relevance between $Q$ and $K$: $i_k(Q,K)$. This improves expressive power while maintaining sparsity. For example, MoBA and BiFormer Zhu et al. (2023) segment $Q$ and $K$, selecting top-$K$ regions based on segment-level similarity:

$$i_k(Q,K) = \arg\max_{\text{top-}K} \left(\bar{Q}\bar{K}^\top\right),$$

where $\bar{Q}, \bar{K} \in \mathbb{R}^{n/s\times d}$ denote segment-level representations. Routing-based methods typically outperform fixed-pattern sparse attention in flexibility and performance. NSA further introduces a compression branch. An explicit sliding-window branch is not strictly necessary, as dynamic routing can converge to such patterns; in practice, it mainly facilitates efficient Triton Tillet et al. (2019) or CUDA kernel implementations.

### 2.3 Linear and Recursive Attention Mechanisms

Linear attention avoids explicit $QK^\top$ computation by approximating softmax with kernel methods. Its general form is

$$\mathrm{Output} = \mathrm{Sim}(Q,K)V = ((\phi(Q)\phi(K)^\top) \odot M)V,$$

where $\phi(\cdot)$ denotes a feature mapping. Common designs include Linear Attention with $\phi(x) = \mathrm{elu}(x)+1$. Linear attention can be expressed in parallel or recursive form. In the recursive formulation, a fixed-size state is maintained:

$$S_t = S_{t-1} + \phi(k_t)^\top v_t, \quad o_t = \phi(q_t)S_t,$$

enabling constant memory during inference. However, due to the absence of softmax normalization and positional bias, linear attention commonly suffers from attention dilution. To mitigate this, Gated Linear Attention (GLA) introduces input-dependent gating decay:

$$S_t = \mathrm{diag}(g_t)S_{t-1} + k_t^\top v_t,$$

where $g_t$ provides dynamic forgetting, encouraging focus on nearby tokens and improving long-sequence stability. However, this design introduces a structural gap from softmax attention, limiting weight structure inheritance.

Another line compresses keys and values online while retaining softmax at the output stage. For instance, Gated Slot Attention (GSA) recursively updates compressed key–value representations:

$$K_t = \mathrm{diag}(g_t)K_{t-1} + (1-g_t)^\top k_t, V_t = \mathrm{diag}(g_t)V_{t-1} + (1-g_t)^\top v_t, o_t = \mathrm{Softmax}(q_t K_t^\top)V_t. \quad (1)$$

Such methods reduce complexity while preserving the softmax weight structure, improving convergence. Finally, **Delta Rule** memory mechanisms introduce error-driven correction by querying the previous state before updating:

$$S_t = S_{t-1} + g_t \cdot k_t^\top \left(v_t - k_t S_{t-1}\right),$$

which significantly enhances state capacity and long-term dependency modeling.

# 3 METHOD

Current linear models still show a gap from standard attention on complex few-shot and retrieval tasks, especially at larger scales. We therefore study whether hierarchical state organization and selective readout can narrow this gap, and evaluate RSA from 360M to 14B parameters. Large-scale results are reported explicitly as Transformer-to-RSA conversions rather than as evidence from training entirely from scratch.

**Core Premise.** Standard attention relies on two essential components: (1) *precise, unbounded information storage* (the KV cache grows with sequence length), and (2) *attention-concentrated readout for regression* (softmax enhances relevant scores while suppressing noise). For a linear model to match standard Attention, it must equivalently preserve all key-value information and retrieve it with focus, not diffusion.

**(1) Storage.** We use multi-level residual-connected states to form a joint code whose nominal number of hard-assignment paths grows multiplicatively with the number of levels. This count is a design-space upper bound, not a guarantee that training realizes every path or that a fixed state losslessly stores an arbitrary sequence. We unify existing state-expansion ideas in a *deep-shallow memory* mechanism and provide a constructive factorized case in Appendix D. **(2) Readout.** Selecting the most relevant stored key–value summaries for autoregression filters noise. Thus, we leverage relevance information from shallow states to route and reweight deep states, producing a concentrated multi-level readout.

The following sections detail these two core components: deep-shallow memory storage and a recursively refined, correlation-based sparse readout mechanism.

## 3.1 DEEP–SHALLOW MEMORY

**Intuition.** Shallow memory stores coarse categories for rapid localization, while deeper levels encode residual distinctions not reconstructed by earlier states. The hierarchy therefore progresses from coarse to fine information without requiring each level to duplicate the complete representation. An everyday analogue is hierarchical recognition: a shallow level separates broad species or topics, whereas later levels retain residual cues needed to distinguish individuals, attributes, or relations within that category. Because each deeper input contains what earlier levels failed to reconstruct, the levels are encouraged to specialize rather than duplicate the same features.

Guided by this intuition, we design the deep–shallow memory module using information theory, KV quantization Hooper et al. (2025), and the KV type bound from PLA Wang et al. (2025a). With $b$ bits per coordinate, a $d_k$-dimensional key has at most $2^{bd_k}$ quantized types. A hierarchy with $n$ levels and $m_i$ slots per level has at most $\prod_i m_i$ hard-assignment paths. Thus $\sum_i \log_2 m_i \geq bd_k$ is sufficient for a nominal type-count match. For FP8 and $m_i = 256$, the construction uses $n = d_k$ levels. The practical $4 \times 64$ configuration instead has $64^4 = 2^{24}$ nominal paths and is an efficiency-oriented setting, not a claim of matching the full quantized-key space.

**Proposition 3.1** (Deep–shallow residual memory)**.** *Let a paired representation $z = (k, v)$ admit a factorized decomposition $z = \sum_{i=1}^{n} p_i(c_i)$ in a direct sum $\mathcal{U} = \bigoplus_i \mathcal{U}_i$, where $p_i(c_i) \in \mathcal{U}_i$, $c_i \in \{1, \ldots, m_i\}$, and $P_i$ is the corresponding coordinate projection. Starting from $z_1 = z$, the recursion $\widehat{z}_i = P_i z_i$, $z_{i+1} = z_i - \widehat{z}_i$ gives $\widehat{z}_i = p_i(c_i)$: each shallow level reconstructs its component and passes only the unreconstructed residual to the next, deeper level. Consequently the joint memory index is $(c_1, \ldots, c_n)$, with at most $\prod_i m_i$ codes and equality when every tuple is reachable.*

RSA uses the following learned relaxation of this exact construction:

$$k_t^{(i+1)} = \mathrm{MLP}^{K(i)}\Big(k_t^{(i)} - \mathrm{SA}\Big(k_t^{(i)}, S_t^{K^{(i)}}, S_t^{K^{(i)}}\Big)\Big), \tag{2}$$

with an analogous value residual. At finite inverse temperature, the attention term is a convex reconstruction from stored prototypes and is not generally an orthogonal Euclidean projection. Appendix D.3 proves that, under a unique winning-slot condition, high-temperature Softmax over a key similarity (or negative key distance) converges to hard selection of a paired key–value codeword. This limiting result motivates the recursion but does not by itself guarantee that a learned RSA occupies every nominal code.

Proposition 3.1 gives a common residual view of existing state-expansion mechanisms. MVA Wang et al. (2025b) forms the next representation by subtracting a codebook reconstruction, $K_t^{(i+1)} = K_t^{(i)} - \mathrm{Softmax}(K^{(i)}W_C^\top)K_t^{(i)}$, and retrieves through learned $W_C$ rather than PLA's direct state query. MoM instead uses $K_t^{(i)} = x_t^{(0)}W^{(i)}$, so its levels are parallel projections of the same input rather than recursively peeled residuals. RSA places both within a common template: level $i$ reconstructs the information explained by its state, removes it, and uses an MLP to reshape the remaining information for level $i+1$. Appendix D gives the exact factorized construction, counting proof, and approximation conditions.

Interestingly, Equation 2 shares the residual attention–MLP skeleton of a Transformer block. This suggests a coarse-to-fine interpretation: each memory level performs a classification or quantization at a different granularity; shallow levels extract broad categories, while deeper levels encode increasingly precise distinctions left in the earlier residuals. The readout then aggregates these scales for classification, regression, or autoregressive prediction. Under this view, some tasks can be decomposed into progressively refined decisions, and abilities may appear only after the representation reaches sufficient granularity. This is an interpretive analogy rather than a functional equivalence claim; Appendix C.2 gives further discussion.

**State update.** RSA uses normalized online assignments. Let $g_t^{(i)} = \bar{f}^{(i)}(k_t^{(i)} {S_{t-1}^{K(i)}}^\top)$, $l_t^{(i)} = l_{t-1}^{(i)} + g_t^{(i)}$, and $u_t^{(i)} = g_t^{(i)} \oslash (l_t^{(i)} + \epsilon)$, where $\bar{f}^{(i)} : \mathbb{R}^{m_i} \to \mathbb{R}_+^{m_i}$ maps slot scores to nonnegative assignment weights and is normalized across slots in the standard implementation. The complete key and value update fits on one line:

$$S_t^{K(i)} = \mathrm{Diag}(1 - u_t^{(i)})S_{t-1}^{K(i)} + {u_t^{(i)}}^\top k_t^{(i)}, \quad S_t^{V(i)} = \mathrm{Diag}(1 - u_t^{(i)})S_{t-1}^{V(i)} + {u_t^{(i)}}^\top v_t^{(i)}. \tag{3}$$

This Delta-rule-like averaging is used in our implementation; Eq. 1 is an alternative parameterization and gives similar multilevel results. The accumulator therefore acts as an online cluster count: repeated, correlated keys refine a shared prototype, and the same assignments update its paired value payload. Weak assignments cause proportionally small changes, which reduces interference between unrelated slots.

**Inter-state connectivity.** The relation $R_t$ records adjacent-level assignments; in practice a static learned map gives comparable performance and lower cost. Reusing the assignments already defined above, its dynamic online update is

$$R_t^{(i,i+1)} = \mathrm{Diag}(1 - u_t^{(i)})R_{t-1}^{(i,i+1)} + {u_t^{(i)}}^\top g_t^{(i+1)} \in \mathbb{R}^{m_i \times m_{i+1}}. \tag{4}$$

Thus each row is a running estimate of which deeper slots co-occur with a shallow assignment; no additional undefined assignment function is required.

### 3.2 Correlation-Aware Sparse Readout

Most existing linear sequence models focus on state storage and updates. Delta-rule-based and multi-state designs have become mature, while the readout process and value aggregation have received less attention. We argue this component is equally critical. Inspired by neuroscience and sparse attention theory, we propose a correlation-aware sparse readout mechanism: retrieval is guided by token relevance—highly correlated states are selectively retrieved, while weak correlations are compressed or filtered, retained only as auxiliary signals for regression.

**Intuition.** Dense readout mixes every stored value and can dilute a relevant signal in long, noisy contexts. RSA instead uses the shallow score as a coarse prior, routes it to compatible deep residual states,

Table 1: Common-sense reasoning performance.

| Model | Wiki ppl↓ | LMB ppl↓ | LMB acc↑ | PIQA acc↑ | Hella. acc↑ | Wino. $\text{acc}_n$ ↑ | ARC-e acc↑ | ARC-c acc↑ | SIQA $\text{acc}_n$ ↑ | BoolQ acc↑ | Avg acc↑ | MMLU acc↑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *360M parameters / 15B tokens* | | | | | | | | | | | | |
| GLA | 28.51 | 43.02 | 28.73 | 64.05 | 35.96 | 50.00 | 54.19 | 24.29 | 37.13 | 58.39 | 44.09 | 23.8 |
| Mamba2 | 30.83 | 40.21 | 29.94 | 63.79 | 35.88 | 49.82 | 49.24 | 24.56 | 35.41 | 60.07 | 43.59 | 25.1 |
| DeltaNet | 28.65 | 47.30 | 28.43 | 63.52 | 35.95 | 49.63 | 52.68 | 25.37 | **37.96** | 58.79 | 44.04 | 24.2 |
| Transformer-Qwen | 30.46 | 39.08 | 29.70 | 65.18 | 37.96 | 49.17 | 56.17 | 26.45 | 36.96 | 59.54 | 45.14 | 25.8 |
| Gated DeltaNet | 27.51 | 31.25 | 33.86 | 63.12 | 36.82 | 50.89 | 55.04 | 26.86 | 34.68 | 59.06 | 45.04 | 24.2 |
| RSA-360M-q (Ours) | **25.89** | **28.89** | **34.98** | **67.26** | **39.98** | **52.67** | **56.70** | **28.34** | 35.22 | 59.83 | **46.87** | 25.3 |
| RSA-360M (Ours) | 27.05 | 35.86 | 32.89 | 66.50 | 39.24 | 52.04 | 56.70 | 27.81 | 37.24 | **60.12** | 46.57 | **26.4** |
| *1.3B parameters / 100B tokens* | | | | | | | | | | | | |
| Transformer-Qwen | 17.82 | 14.63 | 43.29 | 70.64 | 53.96 | 56.25 | 71.58 | 36.34 | 40.90 | 60.11 | 54.14 | 26.8 |
| RetNet | 19.08 | 17.27 | 40.52 | 70.07 | 49.16 | 54.14 | 67.34 | 33.78 | 40.78 | 60.39 | 52.02 | 24.6 |
| HGRN2 | 19.10 | 17.69 | 39.54 | 70.45 | 49.53 | 52.80 | 69.40 | 35.32 | 40.63 | 56.66 | 51.79 | 25.2 |
| Mamba2 | 16.56 | **12.56** | 45.66 | 71.87 | 55.67 | 55.24 | 72.47 | 37.88 | 40.20 | 60.13 | 54.89 | 26.2 |
| Gated DeltaNet | 16.40 | 12.72 | 44.92 | 71.86 | 54.12 | 56.64 | 71.50 | 37.82 | 40.78 | 60.01 | 54.70 | 26.2 |
| PLA | – | – | 45.16 | 71.79 | 55.34 | 56.57 | 71.67 | 37.69 | 40.97 | 60.09 | 54.91 | – |
| RSA-1.3B-q (Ours) | **16.24** | **12.06** | **46.82** | **72.96** | 55.64 | **57.72** | **72.54** | **38.68** | 40.82 | 60.89 | **55.76** | 26.1 |
| RSA-1.3B (Ours) | 16.97 | 13.06 | 45.41 | 72.80 | **55.88** | 56.80 | 72.01 | 38.42 | **40.92** | **61.12** | 55.42 | **28.6** |

and suppresses weak correlations. The resulting readout retains coarse context while emphasizing the fine-grained residuals needed for prediction.

**Readout formulation.** Given a query $q_t$ at time step $t$, we first retrieve the shallow (level-0) state:

$$\hat{a}_t^{(0)} = \sigma\left(q_t S_t^{K(0)\top}\right) W^{(0)} \in \mathbb{R}^{1\times m_0}, \sigma(x) = \frac{1}{1+e^{-x}}$$

where $\sigma(\cdot)W^{(0)}$ replaces the conventional softmax projection and empirically yields better performance. To induce sparsity, instead of dense normalization, we apply a relevance-preserving compression:

$$a_t^{(0)} = \text{ReLU}\left(\hat{a}_t^{(0)} - \frac{1}{z_0}\right) + \frac{1}{z_0 m_0} \text{ or } a_t^{(0)} = \text{GeLU}\left(\hat{a}_t^{(0)} - \frac{1}{z_0}\right),$$

where $z_0$ is either a fixed hyperparameter or a learnable scalar clamped to a valid range. The resulting $a_t^{(0)}$ serves as attention scores for shallow states. **This operation preserves highly relevant scores while uniformly compressing irrelevant ones.**

The shallow scores $a_t^{(0)}$ are then used to retrieve deeper states. Under dense connections, the level-$i$ attention scores $a_t^{(i)}$ are computed as

$$\tilde{a}_t^{(i)} = \left(\prod_{j=0}^{i-1} \text{GeLU}\left(a_t^{(j)} R_t^{(j,i)} W_R^{(j,i)}\right)\right), \hat{a}_t^{(i)} = \left(\sigma\left(q_t S_t^{K(i)\top}\right) W^{(i)}\right), a_t^{(i)} = \tilde{a}_t^{(i)} \cdot \hat{a}_t^{(i)},$$

where $R_t^{(i-1,i)}$ denotes inter-level state connectivity and $R_t^{(j,i)} = I(j \le i-1)$. Since the attention scores are gated recursively, only the previous layer's $\tilde{a}_t^{(i)}$ is needed. If we consider using learnable parameters to replace $R_t^{(i-1,i)}$, we can explore parallel control. The use of GeLU arises from approximating the dynamic weighting operation where shallow scores query deep single-layer scores, enabling the learning of nonlinear redistribution of subsequent scores conditioned on previous states.

### 3.3 Recursive Sparse Attention (RSA)

We now describe the full RSA memory update and readout process. For completeness, the full formulation is presented in Appendix C.

**Sequential formulation.** Let $h_t$ denote the input hidden state and define the level-zero projections $q_t = h_t W_Q$, $k_t^{(0)} = h_t W_K$, and $v_t^{(0)} = h_t W_V$. All state tensors are initialized to zero. At level $i$, let $S_t^{K(i)} \in \mathbb{R}^{m_i \times d_k}$ and $S_t^{V(i)} \in \mathbb{R}^{m_i \times d_v}$. Using the same $g_t^{(i)}, l_t^{(i)}, u_t^{(i)}$ defined above, it suffices to write the key-state update:

$$S_t^{K(i)} = \text{Diag}(1 - u_t^{(i)}) S_{t-1}^{K(i)} + {u_t^{(i)}}^\top k_t^{(i)}.$$

The value state is updated identically by replacing $(S^{K(i)}, k^{(i)})$ with $(S^{V(i)}, v^{(i)})$ while retaining the same key-derived $u_t^{(i)}$. To construct the next level, RSA removes the component explained by the current key state and propagates its residual in one line:

$$\alpha_t^{(i)} = \mathrm{softmax}\Big(\tau_i[\mathrm{sim}(k_t^{(i)}, S_{t,j}^{K(i)})]_{j=1}^{m_i}\Big), \qquad k_t^{(i+1)} = \mathrm{MLP}^{K(i)}\Big(k_t^{(i)} - \alpha_t^{(i)} S_t^{K(i)}\Big).$$

The value residual is computed in exactly the same way, using $\alpha_t^{(i)} S_t^{V(i)}$ and $\mathrm{MLP}^{V(i)}$; thus no value-side distance is required. The inter-level relation is updated by Equation 4. At causal readout, $a_t^{(0)}$ is the shallow sparse score defined above, while for $i \geq 1$ the attention score and final output are computed immediately as

$$\widehat{a}_t^{(i)} = \sigma\Big(q_t {S_t^{K(i)}}^{\top}\Big) W^{(i)}, \qquad a_t^{(i)} = \left[\bigodot_{j=0}^{i-1} \mathrm{GeLU}\Big(a_t^{(j)} R_t^{(j,i)} W_R^{(j,i)}\Big)\right] \odot \widehat{a}_t^{(i)},$$

$$o_t = \frac{\sum_{i=0}^{c} a_t^{(i)} S_t^{V(i)}}{\epsilon + \sum_{i=0}^{c} a_t^{(i)} \mathbf{1}_{m_i}}.$$

Here the first factor routes shallow relevance to deeper residual states, and the second is the direct query–key score. The implementation uses the efficient dot product $k_t^{(i)} {S_t^{K(i)}}^{\top}$ for write-time residualization; cosine similarity is equivalent after normalization, while negative squared Euclidean distance is an optional analytical form. The same key-derived weights preserve key–value binding throughout. Appendix D.3 formalizes the corresponding hard-selection limit.

## 4 Experiments

We train autoregressive language models at several scales and compare RSA with linear-attention models (DeltaNet, Gated DeltaNet, and RWKV-7), state-space models (Mamba and Mamba2), and

Table 2: Controlled conversion results obtained by replacing Qwen2.5 attention with RSA. CPT and SFT denote continued pre-training and the subsequent supervised stage; starred rows are reported checkpoints, whereas the remaining controls are evaluated with our pipeline.

| **Model** | **Tokens** | **PIQA** acc↑ | **Hella.** acc↑ | **Wino.** acc↑ | **ARC-e** acc↑ | **ARC-c** acc↑ | **SCIQ** acc↑ | **MMLU** acc↑ | **Avg** acc↑ |
|---|---|---|---|---|---|---|---|---|---|
| RWKV6-World2.1-3B* | 2.5T | 76.4 | 68.4 | 66.3 | 71.2 | 35.6 | 92.2 | 28.3 | 62.63 |
| Llama3.2-3B* | 15.0T | 76.7 | 73.6 | 69.9 | 74.5 | 42.2 | 95.7 | 56.5 | 69.87 |
| RWKV7-World3-2.9B* | 5.6T | 79.7 | 76.4 | 72.8 | 81.0 | 48.0 | 95.0 | 55.0 | 72.56 |
| Qwen2.5-3B | 18.0T | 78.6 | 73.5 | 68.5 | 77.4 | 45.0 | 96.2 | 65.7 | 72.13 |
| Qwen2.5-3B-CPT | 20B | 76.1 | 72.0 | 68.6 | 77.2 | 45.9 | 95.2 | 55.6 | 70.09 |
| Qwen2.5-3B-SFT | +10B | 75.9 | 72.7 | 69.2 | 76.8 | 46.3 | 95.1 | 56.9 | 70.41 |
| GSA-3B-CPT | 20B | 73.4 | 70.0 | 65.1 | 74.7 | 42.0 | 92.2 | 33.5 | 64.41 |
| RSA-3B-CPT (Ours) | **20B** | 78.9 | 72.1 | 68.0 | 78.5 | 46.0 | 95.6 | 47.6 | 69.53 |
| RSA-3B-SFT (Ours) | **+10B** | 78.8 | 73.2 | 69.5 | 78.2 | 46.5 | 96.0 | 50.8 | 70.43 |
| RSA-3B-CPT (Ours) | **90B** | 79.8 | 74.0 | 70.7 | 81.2 | 47.2 | 96.2 | 55.8 | 72.13 |
| RSA-3B-SFT (Ours) | **+10B** | 79.7 | 75.1 | 72.4 | 81.0 | 47.62 | 96.4 | 57.6 | 72.86 |
| RWKV6-7B* | 1.4T | 78.4 | 75.2 | 68.5 | 73.6 | 44.0 | 95.5 | 43.9 | 68.44 |
| Mantha2-7B* | 1.2T | 81.0 | 77.8 | 72.3 | 77.6 | 46.8 | 95.9 | 33.2 | 69.23 |
| Mistral-7B* | >2T | 80.6 | 81.1 | 74.0 | 80.8 | 54.0 | 95.9 | 62.4 | 75.54 |
| SUPRA* | 100B | 79.9 | 77.1 | 70.3 | 76.0 | 45.7 | 95.0 | 34.1 | 68.30 |
| GSA-7B-Mistral | 100B | 78.9 | 77.9 | 72.6 | 76.0 | 46.9 | 95.2 | 38.1 | 69.37 |
| Llama3.1-8B | 15.0T | 80.1 | 78.8 | 73.6 | 81.5 | 53.7 | 96.2 | 65.0 | 75.56 |
| Qwen2.5-7B | 18.0T | 78.8 | 79.0 | 73.1 | 80.4 | 51.4 | 96.6 | 74.2 | 76.21 |
| Qwen2.5-7B-CPT | 10B | 77.5 | 77.9 | 72.3 | 80.0 | 52.4 | 96.2 | 63.5 | 74.26 |
| Qwen2.5-7B-CPT | 30B | 77.1 | 77.4 | 72.1 | 80.3 | 52.1 | 96.3 | 61.8 | 73.89 |
| Qwen2.5-7B-SFT | +10B | 78.9 | 79.1 | 73.6 | 81.2 | 52.3 | 96.5 | 62.4 | 74.99 |
| RSA-7B-CPT (Ours) | **30B** | 79.6 | 79.3 | 72.7 | 83.2 | 53.8 | 96.3 | 59.3 | 74.89 |
| RSA-7B-SFT (Ours) | **+10B** | 80.2 | 80.1 | 73.8 | 83.3 | 53.1 | 96.6 | 62.0 | 75.59 |
| Qwen2.5-14B | 18.0T | 82.3 | 82.9 | 75.3 | 82.4 | 58.9 | 97.5 | 79.7 | 79.86 |
| Qwen2.5-14B-CPT | 20+10B | 82.5 | 81.8 | 75.6 | 82.8 | 59.7 | 97.1 | 67.6 | 78.16 |
| RSA-14B (Ours) | **20+10B** | 84.1 | 83.1 | 75.9 | 84.7 | 61.6 | 97.6 | 65.6 | 78.95 |

Table 3: Passkey retrieval accuracy. All models are fine-tuned with a context length of 8K tokens.

| Model / Length | 1K | 2K | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-3B | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| Qwen2.5-7B | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| GLA-7B-Qwen | 0.9 | 0.7 | 0.5 | 0.2 | 0.0 | 0.1 | 0.1 | 0.0 |
| GSA-7B-Qwen | 0.9 | 0.8 | 0.5 | 0.3 | 0.1 | 0.2 | 0.1 | 0.0 |
| RSA-3B-Qwen (Ours) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| RSA-7B-Qwen (Ours) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

Transformer controls. We distinguish models trained from scratch from models converted from pre-trained Qwen2.5 checkpoints. Continued pre-training uses random samples from FineWeb-Edu, and the controlled SFT stage uses the same additional token budget for each paired Qwen/RSA comparison. Appendix A.1 records the principal architecture and protocol details.

### 4.1 Training Configurations

**Training from Scratch.** The RSA-360M model ($\sim$360M parameters) is trained on 15B tokens. We use AdamW with a warmup of 100 steps, processing 0.5M tokens per step, for a total of 30,000 steps. The best checkpoint is selected from steps 29,600–30,400. The peak learning rate is $4 \times 10^{-4}$ with cosine annealing, and the sequence length is fixed at 2K tokens. **RSA-360M-q.** This variant adds an additional query branch at the readout stage (C.1) attending to both key and value states, improving short-context performance.

The RSA-1.3B model ($\sim$1.3B parameters) is trained on 100B tokens. We use AdamW with weight decay 0.01 and a 200-step warmup. Each step processes 1M tokens for a total of 100,000 steps. The peak learning rate is $4 \times 10^{-4}$ (cosine annealing), with a sequence length of 2K tokens. Both models use the Mistral tokenizer (vocab size 32,000) and a four-level memory hierarchy with state size 64 per level. For comparison, GSA is trained from scratch with total state size 256.

Table 4: Performance on LongBench tasks.

| **Model** | **SQA** | **MQA** | **Sum** | **FS** | **Code** | **Avg.** |
|---|---|---|---|---|---|---|
| HGRN2-1.3B | 7.38 | 6.02 | 6.51 | 15.50 | 40.11 | 15.10 |
| GSA-1.3B | 8.21 | 6.63 | 7.75 | 20.29 | 42.83 | 17.14 |
| Gated DeltaNet-1.3B | 15.01 | 11.42 | 11.02 | 22.76 | 36.82 | 19.80 |
| RSA-1.3B | 16.12 | 11.39 | 10.23 | 26.72 | 43.35 | 21.56 |
| Llama2-7B-4k | 24.9 | 22.6 | 24.7 | 60.0 | 48.1 | 36.06 |
| RSA-7B-Llama2 (Ours) | 24.34 | 23.68 | 26.52 | 65.72 | 54.00 | 38.85 |
| Qwen2.5-7B | 28.60 | 28.71 | 23.85 | 69.51 | 58.60 | 41.85 |
| RSA-7B-Qwen-SFT (Ours) | 27.54 | 26.16 | 24.96 | 70.72 | 60.40 | 41.96 |

**Conversion from pretrained Transformers.** We initialize the non-attention backbone and the $Q/K/V$ projections from Qwen2.5, replace the attention operator with RSA, and initialize only the new state, routing, and inter-level modules from scratch. For 3B, we report matched 20B-CPT+10B-SFT controls and a longer 90B-CPT+10B-SFT run; for 7B we use 30B CPT followed by 10B SFT; for 14B the controlled comparison uses 20B CPT+10B SFT. The matched Qwen controls receive the same stage-wise data and token budgets, separating architectural effects from additional training. LoRA is applied to attention and MLP layers with rank 512 and $\alpha = 2 \times$ rank. Embedding layers and all new RSA parameters are fully unfrozen and each step processes 0.5M tokens. AdamW with a constant learning rate $1 \times 10^{-4}$ is used, and gradient clipping is applied with norm 1.0. Sequence length is set to 8K tokens, retaining the original Qwen tokenizer. RSA employs four memory levels with state size 64; for comparison, GSA uses total state size 256. The additional parameters introduced by RSA account for less than 0.1% of the original number of parameters. Because MMLU is sensitive to the late-stage data mixture, we interpret matched CPT/SFT rows together rather than attributing differences from the original 18T-token checkpoint solely to architecture.

**Observations.** Across all scales, RSA matches or exceeds existing linear models and Transformer baselines on common-sense reasoning benchmarks (Table 1). Linear models such as RWKV-7 and Gated DeltaNet often converge quickly on perplexity and perform well on short-context tasks,

Table 5: End-to-end inference time (ms) measured in the same environment. RSA-4×64 denotes four levels with 64 slots each. NSA is included as a sparse-attention reference.

| Seq Len | FlashAttention | NSA | RSA-4×64 | RSA-4×32 | RSA-4×16 | GSA-512 | GSA-256 | GLA-512 | GLA-256 | GDN-256 |
|---|---|---|---|---|---|---|---|---|---|---|
| 8K | 47.69 | 57.81 | 103.55 | 82.01 | 41.73 | 125.16 | 82.07 | 102.65 | 69.01 | 68.34 |
| 16K | 117.51 | 123.55 | 199.96 | 156.69 | 80.03 | 245.77 | 159.76 | 201.79 | 134.24 | 134.56 |
| 32K | 331.92 | 284.45 | 392.32 | 306.67 | 157.06 | 487.49 | 315.98 | 400.73 | 265.46 | 268.08 |
| 64K | 1057.79 | 761.73 | 780.40 | 608.23 | 320.67 | 977.67 | 633.21 | 801.59 | 532.05 | 521.02 |
| 128K | 3801.54 | 2388.78 | 1568.54 | 1218.08 | 632.03 | 1961.19 | 1278.01 | 1607.77 | 1066.78 | 1074.86 |
| 256K | 14420.34 | 8121.86 | 3137.63 | 2439.89 | 1273.23 | 3922.17 | 2541.24 | 3217.24 | 2130.93 | 2150.65 |

| Model | LMB | PIQA | Hella. | Wino. | ARC-e | ARC-c | SIQA | BoolQ | Avg | MMLU |
|---|---|---|---|---|---|---|---|---|---|---|
| RSA-4 × 32 | 32.45 | 66.52 | 39.04 | 51.68 | 56.80 | 27.85 | 37.23 | 60.09 | 46.45 | 25.61 |
| GSA-256 | 30.82 | 66.40 | 38.10 | 50.56 | 55.56 | 26.19 | 37.20 | 60.01 | 45.61 | 23.07 |
| RSA-4 × 16 | 30.96 | 66.28 | 38.49 | 50.98 | 56.05 | 28.03 | 37.16 | 59.98 | 45.99 | 25.28 |

particularly at smaller parameter scales. However, on long-horizon tasks (e.g., MMLU), these models exhibit a clear performance gap compared to RSA and Transformers.

These results suggest that existing linear attention models may overemphasize short-context optimization, implicitly converging toward sliding-window attention, which limits their long-range reasoning capabilities.

### 4.2 Retrieval Tasks

Table 6 evaluates RSA-360M and RSA-1.3B trained from scratch on retrieval and recall benchmarks.

On Passkey (Table 3), RSA-7B-Qwen trained at 8K retains 100% accuracy at 64K, whereas GLA and GSA degrade beyond their effective state budgets. This converted-model result is distinct from scratch training: Table 14 shows gradual, non-perfect extrapolation for RSA-1.3B trained only at 2K.

### 4.3 Training and Inference Efficiency

Table 5 reports end-to-end time for explicit slot budgets rather than equating $\prod_i m_i$ with physical cache size. RSA-4×16 is competitive with single-state baselines at 32K; FlashAttention and NSA are faster at short contexts but grow more rapidly with length. The crossover depends on state budget and sequence length, establishing an efficiency–quality trade-off rather than physical-capacity equivalence.

### 4.4 Long Sequence Evaluation on LongBench

We further evaluate long-sequence performance on LongBench (Table 4), reporting both the from-scratch 1.3B models and converted Llama2/Qwen2.5 7B models.

### 4.5 Ablation Study

Table 9 shows that removing sparse readout lowers the six-task average by 0.37 points, while removing deep–shallow memory lowers it by 1.80 points; storage is the larger contributor and sparse readout is complementary. Controlled CPT/SFT rows in Table 2 also show that data stage and token budget materially affect MMLU.

## 5 Conclusion

In summary, RSA combines deep–shallow memory states with correlation-based sparse readout. The hierarchy provides a multiplicative nominal joint-code space and a constructive residual route for factorized information, while sparse readout uses shallow relevance to localize deeper residual memory. Across models from 360M to 14B, the experiments show competitive common-sense performance, stronger long-context retrieval than the evaluated linear baselines, and favorable long-sequence scaling. The theoretical result is deliberately limited to finite-codebook hard assignment and nominal capacity; realized code usage and compression fidelity remain empirical properties of the learned model.

## REFERENCES


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

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, et al. Rethinking attention with performers. *International Conference on Learning Representations*, 2021.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023a.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023b.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

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.

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.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uNrFpDPMyo.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations*, 2022.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization, 2025. URL https://arxiv.org/abs/2401.18079.

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.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165, 2020a.

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 2020b. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, October 2021.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Ziyang Luo, Can Li, Jing Gao, et al. The positional information bottleneck in long-context language models. *arXiv preprint arXiv:2410.14614*, 2024.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Miao Sun, Xiang Chen, Weizhu Chen, et al. Headkv: Memory-efficient all-heads matter kv cache compression for long-context inference. *arXiv preprint arXiv:2410.04889*, 2024.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

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, and Guoqi Li. Pla: The optimal path from softmax attention to linear models via kv cache compression. *openreview*, 2025a. URL https://openreview.net/forum?id=L47lYCynd7. Submitted to ICLR 2026.

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*, 2025b. URL https://openreview.net/forum?id=WSUO2uRDPc.

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*, 2025a.

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

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2025b. URL https://arxiv.org/abs/2406.06484.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng

Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Susan Zhang, Stephen Roller, Naman Goyal, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10323–10333, 2023.

## AI USE STATEMENT

All scientific ideas, theoretical arguments, experimental designs, analyses, and conclusions in this paper were developed by the authors. Large language models were used only to assist with organization, language polishing, and improving the clarity of presentation. The authors reviewed and verified all generated or revised text and take full responsibility for the content of the paper.

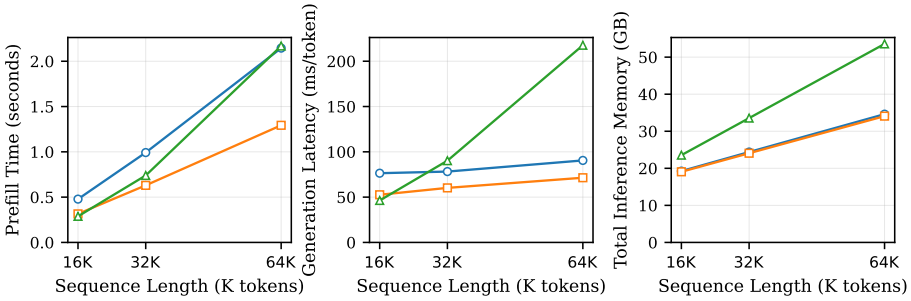


Figure 2: Inference performance of **Mistral-7B** under different attention mechanisms. The green triangle represents FlashAttention2, the blue circle represents RSA-16 $\times$ 4, and the orange square represents GSA-64.

Table 6: Performance of models trained from scratch on standard retrieval and recall benchmarks. Higher scores indicate better performance.

| Model | FDA | SWDE | SQuAD | NQ | TriviaQA | DROP | Avg. |
|---|---|---|---|---|---|---|---|
| **360M Parameters / 15B Tokens** | | | | | | | |
| Transformer++* | 46.14 | 25.87 | 33.22 | 18.94 | 45.97 | 20.03 | 31.70 |
| RetNet* | 5.90 | 9.28 | 22.41 | 6.91 | 40.05 | 18.59 | 17.19 |
| HGRN2* | 11.53 | 17.34 | 24.08 | 12.67 | 43.84 | 17.35 | 21.14 |
| GLA* | 11.26 | 16.78 | 27.85 | 12.77 | 43.90 | 17.68 | 21.71 |
| GSA* | 6.36 | 16.87 | 21.90 | 14.60 | 42.18 | 16.72 | 19.77 |
| Gated DeltaNet* | 20.53 | 23.24 | 28.55 | 14.98 | 44.91 | 16.48 | 24.78 |
| MoM* | 22.98 | 29.90 | 29.69 | 16.60 | 48.82 | 20.99 | 28.16 |
| RSA (Ours) | 32.64 | 28.82 | 31.44 | **18.96** | **49.68** | 20.65 | **30.37** |
| **1.3B Parameters** | | | | | | | |
| Transformer++ | 44.32 | 32.43 | 42.59 | 24.49 | 58.47 | 21.56 | 37.31 |
| RetNet* | 13.62 | 22.59 | 33.46 | 15.43 | 53.79 | 19.79 | 26.45 |
| HGRN2* | 12.35 | 23.24 | 33.19 | 19.10 | 55.27 | 19.65 | 27.13 |
| GLA+* | 27.61 | 30.93 | 35.04 | 22.27 | 56.28 | 19.45 | 31.93 |
| GSA* | 23.25 | 32.80 | 35.57 | 22.96 | 57.05 | 20.65 | 32.05 |
| Gated DeltaNet* | 30.25 | 27.65 | 34.06 | 23.22 | 58.23 | 20.36 | 32.30 |
| MoM* | 41.14 | 34.30 | 37.08 | 24.11 | 58.59 | 21.03 | 36.04 |
| RSA (Ours) | 43.53 | **35.26** | 40.09 | 24.01 | **59.42** | 20.99 | 37.21 |

Table 7: Passkey retrieval accuracy. All models are fine-tuned with a context length of 8K tokens.

| Model / Length | 1K | 2K | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-3B | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| Qwen2.5-7B | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| GLA-7B-Qwen | 0.9 | 0.7 | 0.5 | 0.2 | 0.0 | 0.1 | 0.1 | 0.0 |
| GSA-7B-Qwen | 0.9 | 0.8 | 0.5 | 0.3 | 0.1 | 0.2 | 0.1 | 0.0 |
| RSA-3B-Qwen (Ours) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| RSA-7B-Qwen (Ours) | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

Table 8: Training time and peak memory.

| Model | 32 × 2K | | 8 × 8K | | 2 × 32K | |
|---|---|---|---|---|---|---|
| | Time (it/s) | Memory (MiB) | Time (s) | Memory (MiB) | Time (s) | Memory (MiB) |
| RSA 64 × 4 | 5.56 | 63272 | 5.58 | 63274 | 5.56 | 63271 |
| RSA 32 × 4 | 4.21 | 60536 | 4.23 | 60536 | 4.22 | 60539 |
| GSA 256 | 4.02 | 55266 | 4.03 | 55269 | 4.06 | 55272 |
| GSA 128 | 2.98 | 53830 | 2.98 | 53831 | 3.03 | 53834 |

Table 9: Ablation study of RSA on multiple benchmarks. "Avg." denotes the average score across all listed tasks.

| **Model** | **LMB** | **PIQA** | **Hella.** | **Wino.** | **ARC-e** | **ARC-c** | **Avg.** | **MMLU** |
|---|---|---|---|---|---|---|---|---|
| RSA-360M | 32.89 | 66.50 | 39.24 | 52.04 | 56.70 | 27.81 | 45.86 | 26.4 |
| w/o Sparse read (w/ Intensive reading = PLA) | 31.65 | 66.20 | 38.97 | 51.89 | 55.96 | 28.24 | 45.49 | 26.3 |
| w/o Sparse and Intensive reading = MVA | 30.89 | 65.90 | 38.09 | 51.64 | 56.48 | 27.39 | 45.07 | 25.6 |
| w/o Deep–Shallow Memory (= GSA) | 29.96 | 64.28 | 37.68 | 51.02 | 54.97 | 26.43 | 44.06 | 24.7 |

Table 10: Performance on long-context tasks. MMLU, SQA (Single-hop QA), MQA (Multi-hop QA), Sum (Summarization), FS (Few-shot), Code, and LongBench Avg. RSA-7B models are trained with different token budgets.

| **Model** | **Tokens** | **MMLU** | **SQA** | **MQA** | **Sum** | **FS** | **Code** | **LongBench Avg.** |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-7B | 18.0T | 74.2 | 28.60 | 28.71 | 23.85 | 69.51 | 58.60 | 41.85 |
| Qwen2.5-7B-CPT | 20B | 61.1 | 27.91 | 28.12 | 17.64 | 69.52 | 58.00 | 40.24 |
| RSA-7B-CPT | 20B | 59.1 | 25.25 | 23.68 | 16.96 | 68.96 | 58.00 | 38.57 |
| RSA-7B-CPT | 30B | 60.2 | 26.49 | 24.52 | 17.05 | 69.56 | 59.00 | 39.33 |

# A Additional Controlled and Long-Context Experiments

## A.1 Reproducibility and comparison protocol

Table 11 records the backbone dimensions used in the conversion experiments. Conversion retains embeddings, feed-forward blocks, normalization, tokenizer, and the pretrained $Q/K/V$ projections; RSA state and routing parameters are newly initialized. Within each controlled pair we use the same data stream, context length, token budget, optimizer family, and evaluation code. Unless explicitly marked as a reported checkpoint, all table entries are produced by our evaluation pipeline. We report both the original checkpoints and stage-matched controls because continued pre-training can itself shift downstream scores.

Table 11: Principal backbone configurations used in the conversion experiments.

| Backbone | Layers | $d_{\text{model}}$ | Q heads | KV heads | $d_k$ | $d_{\text{FFN}}$ |
|---|---|---|---|---|---|---|
| Qwen2.5-3B | 36 | 2048 | 16 | 2 | 128 | 11008 |
| Qwen2.5-7B | 28 | 3584 | 28 | 4 | 128 | 18944 |
| Qwen2.5-14B | 48 | 5120 | 40 | 8 | 128 | 13824 |
| Mistral-7B | 32 | 4096 | 32 | 8 | 128 | 14336 |
| Llama2-7B | 32 | 4096 | 32 | 32 | 128 | 11008 |

## A.2 Needle retrieval and extrapolation

Table 12: Needle-in-a-Haystack accuracy for converted 7B models. RSA is trained with an 8K context and evaluated without additional length-specific tuning.

| Model | 2K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|
| Qwen2.5-7B | 100.0 | 100.0 | 100.0 | 100.0 | 99.8 | 46.5 |
| GSA-Qwen-7B | 98.8 | 35.6 | 27.4 | 8.2 | 11.0 | 1.6 |
| PLA-Qwen-7B | 100.0 | 98.8 | 92.6 | 91.2 | 90.2 | 90.0 |
| RSA-Qwen-7B | 100.0 | 99.6 | 98.8 | 99.0 | 97.6 | **95.8** |

Table 13: Single-needle S-NIAH-1 accuracy at longer lengths.

| Model | 2K | 8K | 32K | 128K | 256K |
|---|---|---|---|---|---|
| Qwen2.5-7B | 100.0 | 100.0 | 100.0 | 46.5 | 0.0 |
| GSA-Qwen-7B | 98.8 | 35.6 | 8.2 | 1.6 | 0.6 |
| GDN-Qwen-7B | 97.2 | 31.8 | 4.4 | 0.8 | 1.4 |
| RSA-Qwen-7B | 100.0 | 99.6 | 99.2 | 94.5 | **85.6** |

The converted experiments above benefit from a pretrained backbone. To separate that effect, Table 14 evaluates the 1.3B model trained from scratch only at 2K. Accuracy degrades gradually outside the training length rather than remaining perfect, but useful extrapolation persists through 32K.

Table 14: Passkey accuracy for RSA-1.3B trained from scratch with 2K sequences.

| Length | 2K | 4K | 8K | 16K | 24K | 32K | 64K |
|---|---|---|---|---|---|---|---|
| RSA-1.3B | 1.0 | 1.0 | 1.0 | 0.8 | 0.8 | 0.6 | 0.2 |

Table 15: Protocol sensitivity on the common-sense tasks in Table 1.

| Configuration | Average |
|---|---|
| RetNet | 52.01 |
| HGRN2 | 51.78 |
| Mamba2 | 54.54 |
| Gated DeltaNet | 54.70 |
| RSA-1.3B | 55.42 |
| RSA-1.3B, Llama2 tok., 4K | **55.51** |

### A.3 Protocol sensitivity and long-context reasoning

Changing the tokenizer and sequence length while keeping the 1.3B-scale protocol comparable has little effect on the eight-task average (Table 15), which reduces the likelihood that RSA's gain is an artifact of the default Mistral-tokenizer setup.

Table 16: BABILong accuracy at 128K after task fine-tuning.

| Model | QA1 | QA2 | QA3 | QA4 | QA5 |
|---|---|---|---|---|---|
| ARMT | 100 | 100 | 86 | 100 | 96 |
| Qwen2.5-7B | 100 | 72 | 86 | 96 | 94 |
| RSA-Qwen-7B | 100 | 100 | 98 | 100 | 98 |
| Mistral-7B | 100 | 49 | 36 | 56 | 74 |
| RSA-Mistral-7B | 100 | 100 | 97 | 100 | 95 |

### A.4 Dependence across levels

We estimate total correlation (TC, in bits) and normalized mutual information (NMI) between hard level assignments. Small values support the intended complementary coding, although they neither prove independence nor show that all nominal tuples are occupied. Across 28 layers, mean TC is 0.1162 (range 0.0126–0.1970) and mean NMI is 0.0163 (range 0.0045–0.0464).

Table 17: Layer-wise dependence diagnostics.

| Layer | TC | NMI | Layer | TC | NMI |
|---|---|---|---|---|---|
| 0 | .0771 | .0119 | 14 | .1970 | .0176 |
| 1 | .0236 | .0045 | 15 | .1371 | .0247 |
| 2 | .1550 | .0142 | 16 | .1016 | .0223 |
| 3 | .1621 | .0147 | 17 | .1845 | .0202 |
| 4 | .1542 | .0140 | 18 | .0496 | .0072 |
| 5 | .0670 | .0133 | 19 | .1713 | .0257 |
| 6 | .0126 | .0059 | 20 | .1787 | .0326 |
| 7 | .1964 | .0140 | 21 | .0692 | .0077 |
| 8 | .1134 | .0250 | 22 | .1377 | .0113 |
| 9 | .1564 | .0139 | 23 | .0990 | .0136 |
| 10 | .0347 | .0074 | 24 | .1452 | .0210 |
| 11 | .1896 | .0172 | 25 | .1064 | .0116 |
| 12 | .0523 | .0098 | 26 | .0709 | .0105 |
| 13 | .1583 | .0464 | 27 | .0534 | .0180 |

## B Limitations, Broader Impact, and Future Work

RSA has three important limitations. First, the recursive hierarchy introduces a larger constant factor than simpler linear models at short sequence lengths; the favorable crossover in Table 5 is configuration- and hardware-dependent. Second, $\prod_i m_i$ is a nominal upper bound on joint assignments, not a guarantee that optimization uses every code uniformly or preserves arbitrary token sequences without error. The total-correlation measurements in Appendix A.4 are diagnostics rather than proof of statistical independence. Third, conversion from a pretrained Transformer changes the attention operator while retaining most of the backbone, so performance depends strongly on continued-training data and compute.

Fixed-size recurrent memory offers a useful way to compress long histories, but that compression is necessarily lossy unless the data admit a sufficiently compact representation. RSA's hierarchy and sparse readout improve the available trade-off; they do not imply unbounded or human-like memory. Future work should study learned allocation under distribution shift, calibration of the distance metric and temperature, optimized kernels for the recursive read, and matched-compute evaluations at longer contexts. As with other language models, RSA may reproduce biases or unsafe content from its

training data; deployment requires the same data governance, evaluation, and safeguards used for comparable models.

Table 18: Comparison of NSA, MoBA, and RSA on downstream tasks.

| Model | LMB | PIQA | Hella. | Wino. | ARC-e | ARC-c | SIQA | BoolQ | Avg |
|---|---|---|---|---|---|---|---|---|---|
| NSA-370M | 32.05 | 65.71 | 38.15 | 51.02 | 56.36 | 27.22 | 37.32 | 60.12 | 45.99 |
| MoBA-360M | 32.01 | 66.51 | 39.15 | 51.97 | 56.21 | 26.89 | 36.58 | 60.09 | 46.18 |
| RSA-360M | 32.89 | 66.50 | 39.24 | 52.04 | 56.70 | 27.81 | 37.24 | 60.12 | 46.57 |

## C Parallel Approximation Formulation

The parallel version of our method employs an almost lossless approximation. For ease of understanding, we represent it in operator form as follows:

$$L_t^{(i)} = L_{t-c}^{(i)} + M_L\, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)^{\top}, \tag{5}$$

$${S^{K}}_{t}^{(i)} = {S^{K}}_{t-c}^{(i)} - \Big(\tfrac{1}{L_t^{(i)}}\Big)^{\top} \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)\, {S^{K}}_{t-c}^{(i)} + \tfrac{1}{L_t^{(i)}}\, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)^{\top} K_{t-c:t}^{(i)},$$

$${S^{V}}_{t}^{(i)} = {S^{V}}_{t-c}^{(i)} - \Big(\tfrac{1}{L_t^{(i)}}\Big)^{\top} \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)\, {S^{V}}_{t-c}^{(i)} + \tfrac{1}{L_t^{(i)}}\, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)^{\top} V_{t-c:t}^{(i)}.$$

$$K_{t-c:t}^{(i+1)} = \mathrm{MLP}\Big(K_{t-c:t}^{(i)} - \mathrm{SA}\Big(K_{t-c:t}^{(i)}, {S^{K}}_{t-c}^{(i)}, {S^{K}}_{t-c}^{(i)}\Big)\Big), \tag{6}$$

$$V_{t-c:t}^{(i+1)} = \mathrm{MLP}\Big(V_{t-c:t}^{(i)} - \mathrm{SA}\Big(K_{t-c:t}^{(i)}, {S^{K}}_{t-c}^{(i)}, {S^{K}}_{t-c}^{(i)}\Big)\Big). \tag{7}$$

where $M_L$ is a lower-triangular matrix with all entries in the lower half set to 1.

The local attention outputs are computed as follows:

$$\begin{aligned}\hat{A}_t^{(i)} &= \mathrm{LA}\Big(Q_{t-c:t}^{(i)}, K_{t-c:t}^{(i)}, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big), L_t^{(i)}\Big) \\ &= Q_{t-c:t}^{(i)} \left({S^{K}}_{t-c}^{(i)} - \Big(\tfrac{1}{L_t^{(i)}}\Big)^{\top} \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)\, {S^{K}}_{t-c}^{(i)}\right)^{\top} \\ &\quad + \Big(Q_{t-c:t}^{(i)} K_{t-c:t}^{(i)\top} \odot M\Big) \left(\tfrac{1}{L_t^{(i)}}\, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big)^{\top}\right)^{\top}.\end{aligned} \tag{8}$$

$$A_{t-c:t}^{(i)} = \mathrm{LA}\Big(\hat{A}_t^{(i)}, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big), \bar{f}^{(i+1)}\Big(K_{t-c:t}^{(i+1)} {S^{K}}_{t-c}^{(i+1)\top}\Big), L_t^{(i)}\Big), \tag{9}$$

$$A_{t-c:t}^{(i)} = \sigma\Big(A_{t-c:t}^{(i)}\Big)\, W^{(i)}, \tag{10}$$

$$\tilde{A}_{t-c:t}^{(i)} = \prod_{j=0}^{i} \Big(w_{a^i}\, \mathrm{ReLU}\Big(A_t^{(i)} - \tfrac{1}{z_i}\Big) + \tfrac{1}{z_i m_i}\Big)\, W_r^{(i)}, \tag{11}$$

$$O_{t-c:t}^{(i)} = \mathrm{LA}\Big(\tilde{A}_{t-c:t}^{(i)}, \bar{f}^{(i)}\Big(K_{t-c:t}^{(i)} {S^{K}}_{t-c}^{(i)\top}\Big), V_{t-c:t}^{(i)}, L_t^{(i)}\Big). \tag{12}$$

The operator LA can be replaced with GLA to incorporate data-dependent dynamic decay. The approximation primarily arises in the retrieval of ${S^{K}}_{t-c}^{(i)}$ using $K_{t-c:t}^{(i)}$, where we optionally include the information from the current window; however, the performance gain is minimal, and omitting this computation slightly improves efficiency.

### C.1 Incorporating Short-Sequence Reading and RSA-q

Incorporating the short-sequence read branch from MVA improves performance on short-sequence tasks, but may slightly degrade long-sequence performance. This trade-off is common among linear models: adding short-sequence enhancement mechanisms, such as gating, improves convergence by mimicking sliding-window Transformers, but limits the ability to handle tasks that require information beyond the window. The extrapolation ability of these linear models is similar to SWA, where the model focuses primarily on windowed information, making it relatively insensitive to information outside the window, thus allowing long sequences to maintain near-original perplexity.

For RSA-q, the computation requires only an additional linear projection for $q$ when reading $V$, leaving all other computations identical:

$$o_t = \sum_{i=0}^{c} \frac{a_t^{(i)}}{\sum_{i=0}^{c} a_t^{(i)}} S^{V}{}_t^{(i)} + \frac{q_t W_{qr}^{(i)}}{16} S^{V}{}_t^{(i)}, \quad W_{qr}^{(i)} \in \mathbb{R}^{d_q \times m_i}. \tag{13}$$

The additional $q$-read branch can also be applied before the sparse activation of $a_t^{(i)}$. On the 340M model, this has minimal effect on performance, hence we adopt the above training strategy.

### C.2 Coarse-to-fine interpretability perspective

Equation 2 is structurally related to a Transformer block and can be viewed as progressively refining a representation. Earlier levels extract coarse categories, later levels encode residual distinctions, and the prediction head aggregates features across granularities for classification or regression. This perspective is an interpretation of the architecture rather than a claim about the internal semantics learned by every layer; a direct causal analysis is left to future work.

## D Nominal Capacity, Constructive Factorization, and Softmax Retrieval

### D.1 Tight counting bound and a factorized construction

Let level $i$ expose a discrete index set $\mathcal{C}_i = \{1, \dots, m_i\}$ and let the joint code be $c(x) = (c_1(x), \dots, c_n(x))$. Because the codomain is the Cartesian product $\mathcal{C}_1 \times \cdots \times \mathcal{C}_n$, any such encoder has at most $\prod_i m_i$ distinct joint codes. This is a counting upper bound and does not require a probabilistic independence assumption.

The bound is tight as a design-space statement. Define a factored input domain $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ with $|\mathcal{X}_i| = m_i$, choose a bijection $b_i : \mathcal{X}_i \to \mathcal{C}_i$, and set

$$c_i(x_1, \dots, x_n) = b_i(x_i). \tag{14}$$

Then $c : \mathcal{X} \to \prod_i \mathcal{C}_i$ is a bijection, so every joint tuple is realized exactly once. The construction also gives a precise structural notion of independence: varying factor $x_i$ changes only $c_i$. It does *not* claim statistical independence under an arbitrary data distribution, nor does it imply that a trained RSA model occupies every code.

### D.2 Recursive residual construction

The preceding factorization can be implemented by residual peeling. Suppose the representation decomposes as a direct sum $z = z_1 \oplus \cdots \oplus z_n$, with each $z_i$ taking one of $m_i$ prototypes, and let $\Pi_i$ return the prototype associated with factor $i$. Starting with $r_1 = z$, define

$$c_i = \mathrm{index}(\Pi_i r_i), \qquad r_{i+1} = r_i - \Pi_i r_i. \tag{15}$$

Induction gives $r_i = z_i \oplus \cdots \oplus z_n$; hence the recursion recovers the full tuple $(c_1, \dots, c_n)$ and realizes $\prod_i m_i$ combinations. RSA is a differentiable relaxation of this construction: learned attention replaces the exact selector, and an MLP re-embeds the residual before the next level. Approximation errors may accumulate, so this construction establishes achievability for an ideal factorized encoder rather than a universal guarantee for learned parameters.

### D.3 High-temperature Softmax as key-routed finite-codebook projection

At level $i$, consider the paired codebook $\mathcal{C}^{(i)} = \{(S_j^{K(i)}, S_j^{V(i)})\}_{j=1}^{m_i}$. RSA selects a slot from the current key and stored keys alone. Define the routing score

$$s_j(k) = \mathrm{sim}(k, S_j^{K(i)}), \tag{16}$$

where sim may be a dot product, cosine similarity, or negative squared Euclidean distance. The corresponding paired reconstruction is

$$\alpha_j(\tau) = \frac{\exp[\tau s_j(k)]}{\sum_{r=1}^{m_i} \exp[\tau s_r(k)]}, \qquad (\widehat{k}, \widehat{v}) = \sum_j \alpha_j(\tau)(S_j^{K(i)}, S_j^{V(i)}). \tag{17}$$

This accounts for all three roles without using a value-side distance: the current key is the routing representation, stored keys define the cells, and the paired $S_j^{V(i)}$ is the payload reconstructed with the same weight. Both current and stored keys already lie in the learned key space produced by the model's key projection, so no additional routing maps are required. Dot product is the efficient implementation; cosine is obtained by normalization, and negative squared distance gives the same ordering for normalized vectors because $-\|x - y\|_2^2 = 2x^\top y - 2$. Thus the Euclidean form is sufficient for analysis but is not required by RSA.

**Proposition D.1** (Hard key-selection limit)**.** *Assume $j^* = \arg\max_j s_j$ is unique and let $\Delta = \min_{j \neq j^*}(s_{j^*} - s_j) > 0$. Then*

$$1 - \alpha_{j^*}(\tau) \leq (m_i - 1)e^{-\tau\Delta}, \tag{18}$$

*and therefore $(\widehat{k}, \widehat{v}) \to (S_{j^*}^{K(i)}, S_{j^*}^{V(i)})$ as $\tau \to \infty$.*

*Proof.* For $j \neq j^*$, $\exp[\tau(s_j - s_{j^*})] \leq e^{-\tau\Delta}$. Dividing numerator and denominator of $\alpha_{j^*}$ by $e^{\tau s_{j^*}}$ yields

$$1 - \alpha_{j^*} = \frac{\sum_{j \neq j^*} e^{\tau(s_j - s_{j^*})}}{1 + \sum_{j \neq j^*} e^{\tau(s_j - s_{j^*})}} \leq (m_i - 1)e^{-\tau\Delta}. \tag{19}$$

The reconstruction limit follows because the codebook is finite. If minima tie, the limit is a convex mixture over the tied codewords rather than a unique selection. □

This result corrects a common overstatement: finite-temperature Softmax returns a convex combination and need not be the Euclidean projection onto the convex hull or the orthogonal projection onto the prototypes' span. Only the hard-assignment limit ($\tau \to \infty$) selects the unique best codeword under the chosen similarity; with negative distance, this is the nearest codeword. Consequently the residual is not generally orthogonal.

### D.4 Q/K/V-aware recursion and empirical independence

Combining Equations 16–17 with the recursive construction gives

$$k^{(i+1)} = \mathrm{MLP}^{K(i)}(k^{(i)} - \widehat{k}^{(i)}), \qquad v^{(i+1)} = \mathrm{MLP}^{V(i)}(v^{(i)} - \widehat{v}^{(i)}). \tag{20}$$

Thus later levels are trained on what earlier paired codewords fail to reconstruct. This provides a constructive mechanism for complementary joint codes, but learned code usage remains an empirical question. Table 17 reports low inter-level NMI together with nonzero TC, supporting partial complementarity without asserting exact statistical independence.

For four levels with $m_i = 64$, the nominal joint-code count is $64^4 = 2^{24}$. This number should be compared with discrete code types, not with a physical cache of $2^{24}$ independently retrievable tokens. More generally, $b$-bit coordinates admit at most $2^{bd_k}$ quantized key types; the hierarchy can match such a type space only when $\sum_i \log_2 m_i \geq bd_k$ and the encoder actually realizes those combinations.

### D.5 Formal hierarchical representation system

We now make explicit the system that was only sketched in the main text. Let $\mathcal{Z}$ be the space of key–value representations presented to a memory layer. Level $i$ has a finite index set $\mathcal{C}_i = \{1, \dots, m_i\}$, a paired codebook $\mathcal{S}_i = \{(S_j^{K(i)}, S_j^{V(i)})\}_{j=1}^{m_i}$, a selector $c_i : \mathcal{Z}_i \to \mathcal{C}_i$, and a residual map $F_i : \mathcal{Z}_i \to \mathcal{Z}_{i+1}$. Starting from $z_1 = (k, v)$, the hierarchical encoder is

$$c_i = c_i(z_i), \qquad z_{i+1} = F_i(z_i, c_i), \qquad C(z_1) = (c_1, \dots, c_n). \tag{21}$$

For a domain $\mathcal{D} \subseteq \mathcal{Z}$, define the realized support $\mathcal{A}_C(\mathcal{D}) = \{C(z) : z \in \mathcal{D}\}$ and the realized cardinality $N_C(\mathcal{D}) = |\mathcal{A}_C(\mathcal{D})|$. This separates three notions that were conflated in the initial formulation: the nominal code space $\prod_i \mathcal{C}_i$, the subset actually reached by the encoder, and the probability assigned to that subset by data.

**Theorem D.2** (Cardinality, tightness, and full support)**.** *For every hierarchical encoder of the form in Equation 21,*

$$N_C(\mathcal{D}) \le \prod_{i=1}^{n} m_i. \tag{22}$$

*Equality holds if and only if $C|_{\mathcal{D}}$ is surjective onto $\mathcal{C}_1 \times \cdots \times \mathcal{C}_n$. Such surjectivity is achievable by the factorized residual construction below, but it does not require statistical independence of the random variables $C_1, \dots, C_n$.*

*Proof.* The image of $C$ is a subset of a Cartesian product of size $\prod_i m_i$, which proves the upper bound. Equality of finite cardinalities holds exactly when the image is the whole product. To see why independence is unnecessary, place any strictly positive, non-product probability mass function on all joint tuples. The support still has size $\prod_i m_i$, although the coordinates are statistically dependent. □

An equivalent support condition is *conditional coverage*: for every reachable prefix $(c_1, \dots, c_{i-1})$ and every $a \in \mathcal{C}_i$, at least one input realizes the extended prefix $(c_1, \dots, c_{i-1}, a)$. Recursing over $i$ yields every complete tuple. This property is structural and is weaker than probabilistic independence.

### D.6 Exact residual construction without an orthogonality assumption

The direct-sum construction can be stated without assuming that the factor subspaces are orthogonal. Let $\mathcal{U} = \mathcal{U}_1 \oplus \cdots \oplus \mathcal{U}_n$ be a direct sum, let $p_i : \mathcal{C}_i \to \mathcal{U}_i$ be injective prototype maps, and let $P_i : \mathcal{U} \to \mathcal{U}_i$ be the associated (possibly oblique) coordinate projections satisfying $P_i p_j(c) = 0$ for $j \neq i$ and $P_i p_i(c) = p_i(c)$.

**Proposition D.3** (Exact residual peeling)**.** *For $z_1 = \sum_{i=1}^{n} p_i(c_i)$, define*

$$\widehat{p}_i = P_i z_i, \qquad z_{i+1} = z_i - \widehat{p}_i. \tag{23}$$

*Then $\widehat{p}_i = p_i(c_i)$ and $z_i = \sum_{j=i}^{n} p_j(c_j)$ for every $i$. Consequently the recursion recovers every tuple in $\prod_i \mathcal{C}_i$ and attains the upper bound in Equation 22.*

*Proof.* The claim holds for $z_1$ by definition. If $z_i = \sum_{j=i}^{n} p_j(c_j)$, the coordinate identities give $P_i z_i = p_i(c_i)$. Subtracting it leaves $z_{i+1} = \sum_{j=i+1}^{n} p_j(c_j)$. Induction completes the proof. □

This construction supplies the precise meaning of "independent factors" used by the capacity argument: each coordinate can be changed while the other coordinates remain fixed. Orthogonality is one sufficient implementation, but complementary subspaces and oblique projections are enough. Learned RSA does not assume that a real representation has this exact decomposition; it approximates the peeling operation and tests complementarity empirically.

### D.7 FINITE-TEMPERATURE RECONSTRUCTION AND KEY–VALUE BINDING

The hard-limit proposition above also gives a finite-$\tau$ reconstruction guarantee. Let $D_K = \max_j \|S_j^{K(i)} - S_{j^*}^{K(i)}\|_2$ and $D_V = \max_j \|S_j^{V(i)} - S_{j^*}^{V(i)}\|_2$. Under the unique-gap assumption of Proposition D.1,

$$\|\widehat{k} - S_{j^*}^{K(i)}\|_2 \le D_K(m_i - 1)e^{-\tau\Delta}, \tag{24}$$

$$\|\widehat{v} - S_{j^*}^{V(i)}\|_2 \le D_V(m_i - 1)e^{-\tau\Delta}. \tag{25}$$

Indeed, each error is a convex sum of differences from the winning codeword, and the total non-winning mass is at most $(m_i - 1)e^{-\tau\Delta}$. Because the same $\alpha_j$ reconstructs both components, the value remains bound to the selected key. This is why RSA must account for $Q$, $K$, and $V$: $Q$ supplies the routing representation, $K$ determines the metric partition, and the paired $V$ is the payload whose binding error is controlled by Equation 25.

During both residual construction and causal reading, slot selection is key-routed: the selector is computed from the current routing vector and stored keys, after which the paired stored values are returned. No $v$–$S^V$ distance is needed or used in the reported implementation. This both avoids a more expensive joint distance and prevents target-value information from leaking into the read rule.

### D.8 APPROXIMATION ERROR THROUGH THE RECURSIVE HIERARCHY

The initial proof asserted exact residual removal for learned attention. A correct approximation statement requires explicit stability assumptions. Let the ideal recursion be $z_{i+1} = F_i(z_i - P_i(z_i))$ and the learned recursion be $\widetilde{z}_{i+1} = \widetilde{F}_i(\widetilde{z}_i - \widetilde{P}_i(\widetilde{z}_i))$. Assume on the reachable compact domains that: (i) $F_i$ is $L_i$-Lipschitz; (ii) $\|\widetilde{F}_i(x) - F_i(x)\| \le \eta_i$; and (iii)

$$\|\widetilde{P}_i(x) - P_i(y)\| \le H_i\|x - y\| + \delta_i. \tag{26}$$

Here $\delta_i$ contains finite-temperature codebook error, and may be bounded by Equation 25 when the assignment gap is uniform.

**Proposition D.4** (Recursive error propagation)**.** *Let $e_i = \|\widetilde{z}_i - z_i\|$ and $e_1 = 0$. Under the assumptions above,*

$$e_{i+1} \le a_i e_i + b_i, qquada_i = L_i(1 + H_i), \quad b_i = L_i\delta_i + \eta_i, \tag{27}$$

*and hence*

$$e_r \le \sum_{i=1}^{r-1} b_i \prod_{s=i+1}^{r-1} a_s. \tag{28}$$

*Proof.* Insert and subtract $F_i(\widetilde{z}_i - P_i(z_i))$, apply the approximation bound for $\widetilde{F}_i$, then use the Lipschitz property of $F_i$ and Equation 26. This yields $e_{i+1} \le \eta_i + L_i[(1 + H_i)e_i + \delta_i]$. Unrolling the scalar recurrence proves Equation 28. □

The bound identifies the actual requirement for deep hierarchies: small selector error is insufficient if the inter-level maps amplify residual perturbations. Controlling $L_i$, increasing the assignment gaps $\Delta_i$, or adding contractive normalization limits accumulation.

### D.9 WHEN THE SOFT RECURSION PRESERVES THE HARD JOINT CODE

Let $s_{ij}(z)$ be the hard selector score at level $i$, and suppose the winning index $c_i$ has score margin

$$\gamma_i = s_{i,c_i}(z_i) - \max_{j \ne c_i} s_{ij}(z_i) > 0. \tag{29}$$

If all score differences are $G_i$-Lipschitz, any perturbation satisfying $e_i < \gamma_i/(2G_i)$ preserves the winning index. Combining this observation with Equation 28 gives a sufficient condition for the learned soft recursion to produce exactly the same tuple as the ideal factorized encoder at every level.

**Theorem D.5** (Conditional approximation of a factorized encoder)**.** *Consider a compact, margin-separated domain on which the exact construction in Equation 23 has margins $\gamma_i > 0$. Suppose the inter-level target maps are continuous. For every $\varepsilon > 0$, sufficiently large inverse temperatures and sufficiently expressive MLPs can make the RSA residuals satisfy $e_i < \varepsilon$ at all levels. If in addition $\varepsilon < \min_i \gamma_i/(2G_i)$, RSA realizes the same joint assignments and therefore the same $\prod_i m_i$ factorized codes on that domain.*

*Proof.* On a compact set separated from decision boundaries, the minimum routing-score gap is positive. Proposition D.1 therefore makes each Softmax selector uniformly close to its hard codeword as $\tau_i \to \infty$. Standard universal approximation gives an MLP approximation to each continuous inter-level map. Choosing these errors successively small makes the right-hand side of Equation 28 below $\varepsilon$. The margin condition then preserves every hard decision. □

This theorem is deliberately narrower than the original claim that attention and MLPs approximate *any* maximal-capacity system. Uniform approximation across discontinuous decision boundaries is impossible without a margin or a weaker distributional notion of error. The corrected result establishes constructive sufficiency for compact, margin-separated factorized encoders.

### D.10 State updates as paired online estimators

The normalized state update also has a direct statistical interpretation. For a fixed level, let $g_{tj} \geq 0$ be the assignment weight of token $t$ to slot $j$ and $\ell_{tj} = \epsilon + \sum_{s=1}^{t} g_{sj}$. Applying Equation 3 coordinatewise gives

$$S^K_{t,j} = \frac{\epsilon S^K_{0,j} + \sum_{s=1}^{t} g_{sj} k_s}{\ell_{tj}}, \qquad S^V_{t,j} = \frac{\epsilon S^V_{0,j} + \sum_{s=1}^{t} g_{sj} v_s}{\ell_{tj}}. \tag{30}$$

The identity follows by induction from the running-mean update. With one-hot assignments, each slot is the empirical centroid of its assigned keys and values; with soft assignments, it is a weighted centroid. Using the same $g_{tj}$ in both equations preserves the empirical key–value pairing at slot level. The result explains the storage mechanism but does not imply lossless recovery of every token assigned to the same slot.

The relation matrix has a similar interpretation. With hard adjacent assignments, $r_t^{(i,i+1)} = g_t^{(i)\top} g_t^{(i+1)}$ is a one-sample co-occurrence matrix. Its normalized running average estimates cross-level routing frequencies. A row normalization approximates $p(C_{i+1} = b \mid C_i = a)$ and supplies a data-dependent transition for hierarchical readout; the learned static replacement trades this direct interpretation for lower cost.

### D.11 Entropy, dependence, and effective utilization

Let $C = (C_1, \ldots, C_n)$ be the random joint assignment induced by a data distribution. Its entropy satisfies

$$H(C) = \sum_{i=1}^{n} H(C_i) - \mathrm{TC}(C) \leq \sum_{i=1}^{n} \log_2 m_i = \log_2 \prod_{i=1}^{n} m_i, \tag{31}$$

where $\mathrm{TC}(C) = D_{\mathrm{KL}}(p(C) \| \prod_i p(C_i)) \geq 0$ is total correlation. Equality in the entropy bound requires both uniform marginals and zero total correlation; it is much stronger than merely reaching every tuple. The entropy-effective number of codes $N_{\text{eff}} = 2^{H(C)}$ is therefore at most $\prod_i m_i$.

Two complementary diagnostics are useful in practice: the occupancy ratio $\rho_{\text{occ}} = |\operatorname{supp}(C)| / \prod_i m_i$ measures how much of the nominal Cartesian product is reached, while $N_{\text{eff}}$ measures how evenly probability is distributed. Pairwise NMI and estimated TC in Table 17 probe redundancy, but pairwise NMI alone cannot rule out higher-order dependence. Accordingly, the empirical results support complementary coding rather than proving exact independence or full utilization.

### D.12 SCOPE OF THE CAPACITY STATEMENT

The product bound concerns distinguishable *joint slot assignments for one encoded representation*. It is not a theorem that a fixed memory stores an arbitrary token sequence losslessly. If an alphabet has $M$ possible symbols, injectively representing every length-$T$ sequence would require at least $M^T$ discrete memory states; a joint code with $\prod_i m_i < M^T$ cannot do so without assumptions on the data or task. Conversely, continuous state vectors have formally infinite cardinality in exact real arithmetic, so a finite "bit capacity" is meaningful only after specifying precision, noise, or a decoding margin. Our $2^{bd_k}$ comparison adopts a $b$-bit quantization convention, while $\prod_i m_i$ counts routing types. These quantities are comparable as discrete type counts, not as claims of equal physical storage.

### D.13 APPROXIMATION OF ATTENTION SCORES VIA DYNAMIC WEIGHT ADJUSTMENT

We perform a simple dot product between the shallow attention scores and the scores from a single deep layer, and then use a nonlinear function to generate dynamic weights. The purpose of these weights is to dynamically adjust the routing of the shallow attention scores to the deep layer scores.

However, instead of directly multiplying $a'^{(i)}_t = a^{(i-1)}_t R^{(i-1,i)}_t$ and $\hat{a}^{(i)}_t$, we suppress weak correlations via thresholded routing:

$$\hat{a}'^{(i)}_t = \hat{a}^{(i)}_t \left( \mathrm{ReLU}\Big( a'^{(i)\top}_t \hat{a}^{(i)}_t \Big) - s \right),$$

where $s$ is a learnable threshold. Since $a$ is scalar-valued, an equivalent kernel-expanded formulation is used in practice:

$$\hat{a}'^{(i)}_t = \hat{a}^{(i)}_t \mathrm{ReLU}\Big( a'^{(i)}_t - s \Big)^{\top} \mathrm{ReLU}\Big( \hat{a}^{(i)}_t - s \Big) = w_a \, \mathrm{ReLU}\Big( \hat{a}^{(i)}_t - s \Big) \to \mathrm{GeLU}\Big( \hat{a}^{(i)}_t \Big),$$

where $w_a$ is a scalar normalization factor. Finally, we use the Gelu approximation to estimate the final attention scores $\hat{a}'^{(i)}_t$ of the current layer, and reassign it to $\hat{a}^{(i)}_t$. Although higher-dimensional relevance metrics (e.g., $R^{(0,1)\top}_t S^{K(0)}_t$ versus $S^{K(1)}_t$) are more expressive, we adopt this formulation to balance computational efficiency and performance.