On the Recall Scaling Laws in Mamba: A Theoretical and Mechanistic Study via Hashing

Yuval Koren **Assaf Ben-Kish** **Raja Giryes** **Lior Wolf** **Itamar Zimerman**
Blavatnik School of Computer Science and AI, Tel Aviv University

Abstract

Associative Recall (AR) is the cognitive ability to learn and retrieve links between items in memory. In NLP, AR is used as a benchmark for evaluating the in-context memory capacity of architectures such as Mamba, and has been found to strongly correlate with language modeling performance. This paper explores AR from the perspective of mechanistic interpretability, aiming to reverse-engineer the exact internal algorithm used by Mamba to perform recall. Our key insight is that Mamba performs recall by implicitly learning linear hash functions, and we identify the low-level circuit that enables this behavior. Building on these findings and inspired by theoretical tools in similarity-preserving hashing, such as the Johnson–Lindenstrauss lemma, we develop a theoretical framework for analyzing AR, which we term *Recall Scaling Laws*. Given the vocabulary size and the number of facts in context, this framework allows us to (1) predict the embedding and state dimensions required for Mamba to achieve perfect recall, (2) predict recall success probability given the model dimensions, and (3) analyze multi-layer models and multi-head SSM patterns. Empirical results show that our theoretical findings are accurate and predictive, offering insights into how AR capacity scales with vocabulary, state, embedding size, and architecture.

 github.com/yuvalko1/mamba-recall-scaling-laws

1 Introduction

Although Transformers have achieved remarkable success in sequence modeling, they suffer from a key inefficiency when handling long sequences: during decoding, memory complexity grows linearly with sequence length due to key-value caching. Recent architectures such as Mamba [1], RWKV [2], and other linear RNNs [3] address this by utilizing a fixed-size recurrent state, enabling constant memory complexity. However, this fixed-size state is a double-edged sword: from an information perspective, compressing the entire context into a single vector imposes inherent limitations and may lead to information loss [4, 5]. Hence, improving memory utilization in recurrent architectures has become an active area of research [6, 7]. To better understand the memory bottlenecks of sequence models, researchers have proposed synthetic benchmarks such as Associative Recall (AR) [8], Multi-Query Associative Recall (MQAR) [9] and others [10, 11], which are highly associated with language modeling capabilities [9]. These tasks are designed to diagnose a model's recall abilities in controlled environments and have been widely studied across various architectures [12, 13], including Transformers [14], linear RNNs [15, 16], and others [17].

In this paper, we take a step toward a deeper understanding of Mamba's recall abilities by adopting a mechanistic interpretability perspective [18, 19], aiming to reverse-engineer the internal algorithm employed by linear RNNs to solve AR tasks. Through empirical analysis, we identify the circuit-level mechanism that enables Mamba to perform AR and interpret it as implicitly computing a similarity-preserving linear hash function [20–22]. Building on this connection, and drawing on theoretical tools from linear hashing such as the Johnson–Lindenstrauss lemma [23], we develop the *Recall Scaling Laws* theoretical framework, which allows us to analyze how the state size and embedding dimension must scale with the vocabulary size and the number of facts in order to achieve perfect

Correspondence: yuvalkoren4@mail.tau.ac.il

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

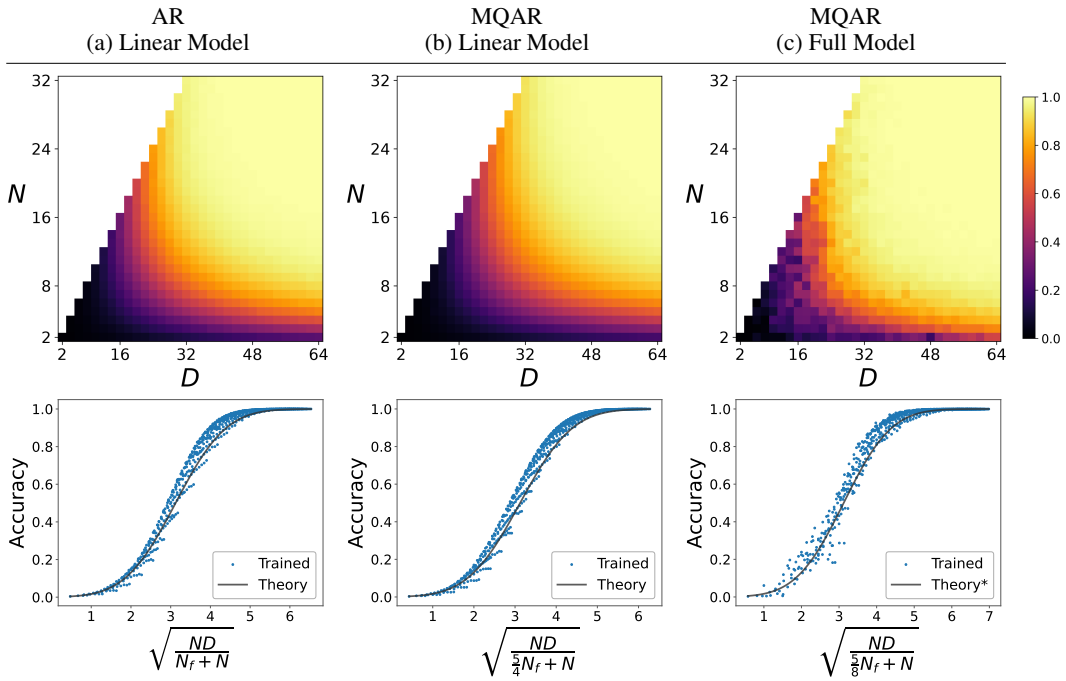


Figure 1: **Mamba recall scaling laws: theory versus empirical results. Top:** Recall accuracy of trained Mamba, represented by pixel color, against model embedding dimension D and state size N. **Bottom:** the same accuracies collapse onto the theoretical curve $p_{\text{recall}}(x) \approx \Phi(x - \sqrt{2 \log V})$ of Thm. 4.2 and Rem. 4.3, with $x = \sqrt{\frac{ND}{aN_f + N}}$. Left to right: (a) linear model on AR, (b) linear model on MQAR, (c) full model, theory fits up to a heuristic constant; see Sec. 5.2. (Here $V = 1024$, $N_f = 16$; see more regimes in Figs. 6 and 7, Appendix.)

recall. Empirical analysis shows that these scaling laws are relatively accurate and reflect the model's behavior in practice.

Our main contributions encompass the following aspects: (1) We study the associative recall capabilities of Mamba through the lens of mechanistic interpretability. By progressively removing components from the model, we isolate the circuit-level mechanism responsible for performing recall. (2) Motivated by this finding, we interpret the learned weights as implicitly computing similarity-preserving linear hash functions, and the entire circuit as exhibiting hash-like behavior. This interpretation allows us to leverage the JL lemma to better understand the theoretical recall capacity of Mamba. (3) We propose a theoretical framework, which we term *Recall Scaling Laws*, that provides an upper bound on the state size, context length and embedding size required to perform near-perfect recall for a given vocabulary size and number of facts, as well as a prediction of recall probability. (4) We extend our framework to multi-head and multi-layer Mamba models. (5) We provide extensive empirical validation of the scaling laws.

Paper roadmap. Sec. 3 suggests theoretical recall circuits (Sec. 3.1) and validates them in trained models via mechanistic-interpretability experiments (Sec. 3.2); Sec. 4 develops the corresponding scaling laws, which are validated empirically in Sec. 5. Given recall tasks of N_f facts over a vocabulary of size V, the requirements on Mamba embedding dimension D and state dimension N form the following ladder of guarantees:

1. Exact recall on every context is achievable by a (non-feasible) non-compressive model with dimensions $N{=}D{=}V$ (worst-case; Thm. 3.1).
2. Exact recall is similarly achievable by a compressive, hash-based model with smaller dimensions, satisfying $\sqrt{ND} = O(N_f \log V)$ (worst-case; Lem. 4.1).
3. High-probability recall is achievable with even smaller model dimensions, satisfying $ND = O(N_f \log V)$ (mean-case; Thm. 3.2 qualitatively, Thm. 4.2 and Rem. 4.3 quantitatively).

The last level is essentially optimal, since a matching lower bound exists as well:

4. High-probability recall *requires* model dimensions satisfying $ND = \Omega(N_f \log V)$ (mean-case; Rem. 4.4 Mamba-specific, Lem. 4.5 general information-theoretic).

In short, the required state memory is linear in the number of facts and logarithmic in vocabulary size.

2 Background & Related Work

Associative Recall. The AR task was first introduced by Ba et al. [8], and has proven to be an effective tool for evaluating recall and retrieval capabilities. More recently, AR has gained popularity due to its strong correlation with language modeling performance [9]. A growing body of work has leveraged AR to guide architectural choices [17, 15, 7, 24], improve initialization strategies [16], analyze optimization dynamics [25], and provide both theoretical and empirical insights into the limitations of recurrent LLMs [4, 5, 26]. The work most closely related to ours is by Bick et al. [13], who use mechanistic interpretability to analyze in-context retrieval through a gather-and-aggregate mechanism. However, it does not study AR or provide theoretical analysis. Le Corre et al. [27] probe a simplified Mamba on small-vocabulary MQAR, identifying write, read and forget behaviours, but do not address capacity. Jelassi et al. [28] investigate the copying capabilities of Transformers and SSMs, offering theoretical insights into these mechanisms. While relevant, their work does not directly examine AR and does not employ mechanistic interpretability to understand what models learn in practice. Nichani et al. [29] bound in-weight fact storage in Transformers, linear in parameter count; we characterize in-context storage in a fixed-size state. Finally, Huang et al. [30] explore the recall capabilities of Mamba by leveraging the JL lemma to derive theoretical bounds, but do not adopt a mechanistic interpretability perspective nor investigate the mechanisms learned in practice. In contrast, our analysis culminates in tighter recall scaling laws showing how both state size and embedding size must scale with vocabulary, and extends to multi-layer and multi-head architectures. (See detailed comparison in App. J.)

Multi-Query Associative Recall (MQAR). MQAR by Arora et al. [9] is an extension of the classical AR. It is constructed by concatenating a factual context section with a multiple queries section to form a single prompt. The vocabulary $\mathcal{V}$ is partitioned into two disjoint subsets: the key vocabulary $\mathcal{V}_k$ and the value vocabulary $\mathcal{V}_v$ (see Table 1, Appendix). The vocabulary size is $V = |\mathcal{V}|$, which is assumed to be even so that half of the tokens correspond to keys, and the other half correspond to values ($V_k = |\mathcal{V}_k| = V_v = |\mathcal{V}_v| = \frac{V}{2}$). Each prompt has a total sequence length L. The context is composed of N_f non-repeating key-value pairs (k_i, v_i), resulting in $2N_f$ tokens. The remaining $L - 2N_f$ tokens form the query section, which includes N_f query tokens q_i, each duplicating a key token that appears in the context. The rest of the query section is filled with padding. Padding tokens may be set to zero or chosen at random, and following the original implementation, we assume they are randomly sampled from the full vocabulary $\mathcal{V}$. For each query q_i, the model's task is to retrieve and output the value v^* corresponding to the fact (k^*, v^*) with a matching key $k^* = q_i$. An example *MQAR* input x and ground-truth output y sequences can be found below, with highlighted key, value, query and padding tokens:

$$\begin{array}{ll} x & \texttt{A 6 B 3 C 7 B 2 5 0 9 C 4 A 8 1} \\ y & \texttt{* * * * * * 3 * * * * 7 * 6 * *} \end{array}$$

where $*$ are *ignored* output tokens (which does not affect the loss). Additionally, the term *AR* is used in this paper to denote a single-query associative recall task, i.e. with $L = 2N_f + 1$.

Mamba. The recently presented *Selective SSM* [1], known as S6, outperforms previous SSMs and various other architectures in NLP [31, 32], vision [33, 34], graph classification [35], and more. S6 incorporates a dynamic input-dependent form of the discrete matrices $\bar{A}$, $\bar{B}$, and C, such that for every time-step the SSM employs a different recurrent rule. This technique differs from the previous state-space layers, which use the same set of matrices for each timestep.
A *Mamba* model of embedding size D and state size N is composed of Λ stacked Mamba blocks. The *Mamba block* combines the S6 state space model (SSM), a Conv1D layer, and other elementwise operators; it borrows elements from Gated MLP and similar architectures. We denote the expanded dimension by $D_{\text{in}} = \text{expand} \cdot D$ (where $\text{expand} \in \mathbb{N}$), and the convolution filter size by D_{conv}.
Given an embedded input sequence $x^e = (x_0^e, \ldots, x_{L-1}^e) \in \mathbb{R}^{D \times L}$, the Mamba block operates as follows (see Tab. 2, 3 for notation details):

$$\hat{x} = \text{SiLU}(\text{Conv1D}(\text{Linear}(x^e))),\ \hat{z} = \text{SiLU}(\text{Linear}(x^e)),\ \hat{y} = \text{SSM}(\hat{x}) \odot \hat{z},\ y^e = \text{Linear}(\hat{y}) \quad (1)$$

where $\odot$ denotes element-wise multiplication, and we omit normalization layers for simplification. For each timestep t, the SSM vectors B_t, $C_t \in \mathbb{R}^N$ and discrete matrices $\bar{A}_t$, $\bar{B}_t \in \mathbb{R}^{D_{\text{in}} \times N}$ are computed as:

$$B_t = S_B\, \hat{x}_t\,,\ C_t = S_C\, \hat{x}_t\,,\ \Delta_t = \text{SoftPlus}(S_\Delta \hat{x}_t)\,,\ \bar{A}_t = \exp(\Delta_t A)\,,\ \bar{B}_t = \Delta_t B_t \quad (2)$$

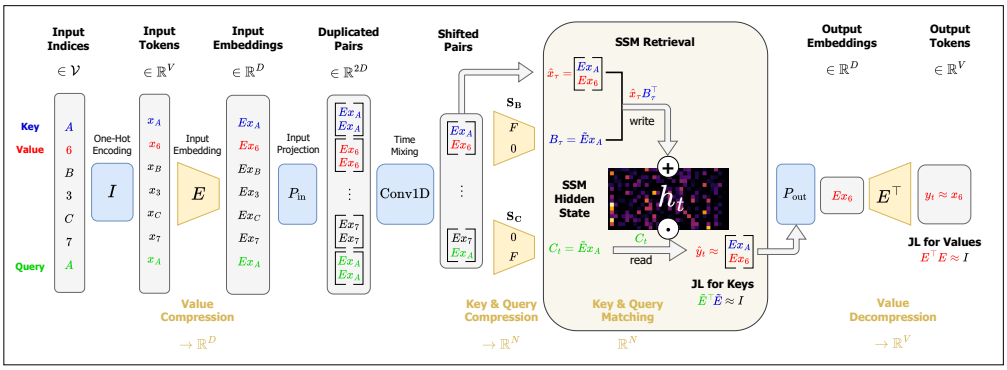


Figure 2: **Minimal recall circuit.** An overview of the Mamba recall circuit in Thm. 3.2. Left to right: (1) Tokens are embedded using $E \in \mathbb{R}^{D\times V}$. (2) P_{in} and Conv1D apply duplication, copy and shift, forming pairs $\binom{Ex_{t-1}}{Ex_t}$. (3) SSM projections S_B, S_C extract keys and queries, then further compress them via $\tilde{E} = FE \in \mathbb{R}^{N\times D}$. (4) Embedding pairs are stored in the SSM using $h_t = \sum_\tau^t \hat{x}_\tau B_\tau^\top$. (5) JL matrix $\tilde{E}$ enables retrieval through key-query matching, $h_t C_t = \sum_\tau^t \hat{x}_\tau (B_\tau^\top C_t) \approx \hat{x}_\tau$. (6) P_{out} selects the value half Ex_τ. (7) JL matrix E enables value reconstruction, $y_t = E^\top E x_\tau \approx x_\tau$.

where S_B, S_C, S_Δ are linear projection matrices, SoftPlus is a smooth approximation of ReLU, and biases are omitted for simplicity. Then, each SSM channel $d = 1, \ldots, D_{\text{in}}$ transforms an input $\hat{x}_t^d \in \mathbb{R}$ into an output $\hat{y}_t^d \in \mathbb{R}$, through updating a recurrent hidden state vector $h_t^d \in \mathbb{R}^N$:

$$h_t^d = \bar{A}_t^d \odot h_{t-1}^d + \hat{x}_t^d \, \bar{B}_t^d \,, \quad \hat{y}_t^d = h_t^d \cdot C_t \tag{3}$$

where $\bar{A}_t^d, \bar{B}_t^d, C_t \in \mathbb{R}^N$, $\odot$ denotes an element-wise product, and $\cdot$ is a dot product. All channels form the hidden state matrix, $h_t \in \mathbb{R}^{D_{\text{in}}\times N}$ (See App. B.3 for details).
The usage of time-variant layers adds to the expressivity of the layer [36], allowing it to adapt to the input, and potentially captures more complex dependencies. While other input-dependent time-variant mechanisms have been proposed in previous works through gated RNNs, S6 also presents an efficient IO-aware implementation, which is parallelized on GPUs via work-efficient parallel scanners [37, 38], making it well-suited for long-context processing [39, 40].

Mechanistic Interpretability. Mechanistic interpretability seeks to reverse-engineer deep learning models so they can be trusted and controlled [18, 19]. The approach works by linking observable behaviors to specific neural components, often by assigning semantic functions to individual neurons or weights and identifying compact "circuits" within the network. Representative examples include the induction-heads [14] and arithmetic mechanisms [41–43], in-context learning via task-vectors [44] and thinking progress via progress-vectors [45]. Our work follows this line of research and aims to uncover the intrinsic mechanisms responsible for recall in Mamba models and linear RNNs.

3 Reverse Engineering Mamba on MQAR

We aim to reverse-engineer the underlying algorithm that enables Mamba to perform MQAR. To do so, in Sec. 3.1 we present a minimal yet efficient Mamba model that successfully solves MQAR. Next, in Sec. 3.2, through a mechanistic interpretability analysis, we verify that our suggested mechanism is indeed the one learned in practice.

3.1 A Mamba Circuit for MQAR

We describe a simplified, purely-linear, single-layer Mamba model, and show theoretically and empirically that it is able to solve MQAR with arbitrarily high probability. A schematic overview of how the minimal model performs recall is presented in Fig. 2.

Minimal model and simplified SSM. In Alg. 1 we present a minimal single-layer Mamba model architecture, where gating, discretization, nonlinearities, biases, normalization layers, and residual connections are all removed. Thus, the simplified-Mamba layer is a linear attention preceded by a Conv1D. Our theorems are stated for this model, which actually fits a broader class of architectures. Dimensions are listed in Tab. 3, Appendix. The model is wrapped with embedding and unembedding layers of a dimension D, which expands to size $D_{\text{in}} = \text{expand} \cdot D$ in the input projection. We further note that without discretization, the SSM recurrent update step described in Eq. 3 becomes rather

Algorithm 1 Simplified Single-Layer Mamba

Input: $x \in \mathbb{R}^{V \times L}$
Output: $y \in \mathbb{R}^{V \times L}$
Parameters: // $D_{\text{in}} = \text{expand} \cdot D$
$E \in \mathbb{R}^{D \times V}$, $P_{\text{in}} \in \mathbb{R}^{D_{\text{in}} \times D}$,
$W_{\text{conv}} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{conv}}}$,
$S_B, S_C \in \mathbb{R}^{N \times D_{\text{in}}}$, $P_{\text{out}} \in \mathbb{R}^{D \times D_{\text{in}}}$
$x_{\text{e}} \leftarrow E\,x$ // input embedding
Mixer:
$x_{\text{in}} \leftarrow P_{\text{in}}\,x_{\text{e}}$
$\hat{x} \leftarrow \text{Conv1D}(x_{\text{in}}, W_{\text{conv}})$
$B \leftarrow S_B\,\hat{x}$
$C \leftarrow S_C\,\hat{x}$
$\hat{y} \leftarrow \text{SSM}(\hat{x}, B, C)$
$y_{\text{out}} \leftarrow P_{\text{out}}\,\hat{y}$
$y \leftarrow E^\top\,y_{\text{out}}$ // output embedding
return y

Algorithm 2 Hidden State Inversion

Input: $h_t \in \mathbb{R}^{2D \times N}$ // $D_{\text{in}} = 2D$
Output: $h_t^{\text{inv}} \in \mathbb{R}^{2V \times V}$
Parameters:
$E \in \mathbb{R}^{D \times V}$, $P_{\text{in}} \in \mathbb{R}^{2D \times D}$,
$W_{\text{conv}} \in \mathbb{R}^{2D \times 2}$ // $D_{\text{conv}} = 2$
$S_B, S_C \in \mathbb{R}^{N \times 2D}$, $P_{\text{out}} \in \mathbb{R}^{D \times 2D}$
Operators:
$\hat{E}_{\text{in}} \leftarrow \left(\text{diag}(W^0_{\text{conv}})\,P_{\text{in}}\,E \;\middle|\; \text{diag}(W^1_{\text{conv}})\,P_{\text{in}}\,E\right)$
$\Pi_{q,\text{in}} \leftarrow S_C\,\hat{E}_{\text{in}}$ // input projection
$\Pi_{v,\text{out}} \leftarrow E^\top P_{\text{out}}$ // output projection
Inversion:
$h_t^{\text{inv}} \leftarrow \Pi_{v,\text{out}}\,h_t\,\Pi_{q,\text{in}}$
Visualization: // $k_n, v_n \in \mathbb{R}^V$: facts key/value
$H_t \leftarrow \sum_{n=1}^{N_f} k_n^\top v_n \in \mathbb{R}^{V \times V}$
$H'_t \leftarrow (0 \mid I_D)\,h_t \in \mathbb{R}^{D \times N}$
$H''_t \leftarrow (0 \mid I_V)\,h_t^{\text{inv}} \in \mathbb{R}^{V \times V}$
return h_t^{inv}

simple. If we additionally set A to identity, the SSM operation can now be written as (see App. B.3 for details):

$$h_t = h_{t-1} + \hat{x}_t\,B_t^\top, \quad \hat{y}_t = h_t\,C_t \tag{4}$$

Mamba recall circuits. We first describe an ideal (dimensionally inefficient) non-compressive circuit that solves MQAR. Next, we present a rather realistic, compressive circuit as a feasible approximation of the ideal one.

Theorem 3.1 (Perfect non-compressive recall circuit). *Given a vocabulary size V, a single-layer simplified Mamba with dimensions $D = N = V$, $\text{expand} = 2$, $D_{\text{conv}} = 2$ can **perfectly solve** an MQAR task (recall probability = 1).*

Proof sketch. (Full proof can be found in App. C.2.) We set the model weights as follows:

$$P_{\text{in}} = \left(\begin{smallmatrix} I_V \\ I_V \end{smallmatrix}\right),\; P_{\text{out}} = \left(0 \mid I_V\right),\; W_{\text{conv}} = \left(\begin{smallmatrix} 1_V & 0_V \\ 0_V & 1_V \end{smallmatrix}\right),\; E = I_V,\; S_B = \left(I_V \mid 0\right),\; S_C = \left(0 \mid I_V\right) \tag{5}$$

where $\mid$ denotes concatenation. P_{in} duplicates the input, $x_t^p = P_{\text{in}}\,E\,x_t = \left(\begin{smallmatrix} x_t \\ x_t \end{smallmatrix}\right)$. The Conv1D performs copy and shift; the SSM input is $\hat{x}_t = \text{Conv1D}\big((x_{t-1}^p, x_t^p), W_{\text{conv}}\big) = \left(\begin{smallmatrix} x_{t-1} \\ x_t \end{smallmatrix}\right)$. Inside the SSM, we have $B_t = S_B\,\hat{x}_t = x_{t-1}$ and $C_t = S_C\,\hat{x}_t = x_t$. After unrolling the expression in Eq. 4 and plugging in the above values, the SSM operation becomes $\hat{y}_t = h_t\,x_t = (\sum_{\tau=0}^{t} \hat{x}_\tau\,B_\tau^\top)\,x_t = \sum_{\tau=0}^{t} \left(\begin{smallmatrix} x_{\tau-1} \\ x_\tau \end{smallmatrix}\right) x_{\tau-1}^\top\,x_t$. Finally, given a query $x_t \equiv q_t \in \mathcal{V}_k$, the output is:

$$y_t = E^\top\,P_{\text{out}}\,\hat{y}_t = \sum_{\tau=0}^{t} x_\tau\,\langle x_{\tau-1}, q_t \rangle \tag{6}$$

Since input vectors are one-hot (orthonormal) and keys are unique, $\langle x_{\tau-1}, q_t \rangle = \delta_{x_{\tau-1},\,q_t}$, so the sum collapses to $y_t = v^*$. □

The circuit described in Theorem 3.1 is impractical because it relies on large state and model dimensions (increases linearly with V), which are unrealistic in practice. Thus, the following theorem presents a more efficient circuit, based on an approximate solution rather than a perfect one.

Theorem 3.2 (Efficient compressive recall circuit). *Given an MQAR task with vocabulary size V, N_f facts and context length $L = 4N_f$, a single-layer simplified Mamba with dimensions satisfying $ND = O(N_f \log V)$, $\text{expand} = 2$, $D_{\text{conv}} = 2$ can solve the task with high probability. (See Lem. 4.1 and Thm. 4.2 for the exact quantitative laws.)*

Proof sketch. (See App. C.3 for full proof.) We adapt the circuit from Thm. 3.1 to enable compression. Given matrices $E \in \mathbb{R}^{D \times V}$ and $F \in \mathbb{R}^{N \times D}$, we now choose:

$$P_{\text{in}} = \left(\begin{smallmatrix} I_D \\ I_D \end{smallmatrix}\right),\; P_{\text{out}} = \left(0 \mid I_D\right),\; W_{\text{conv}} = \left(\begin{smallmatrix} 1_D & 0_D \\ 0_D & 1_D \end{smallmatrix}\right),\; S_B = \left(F \mid 0\right),\; S_C = \left(0 \mid F\right) \tag{7}$$

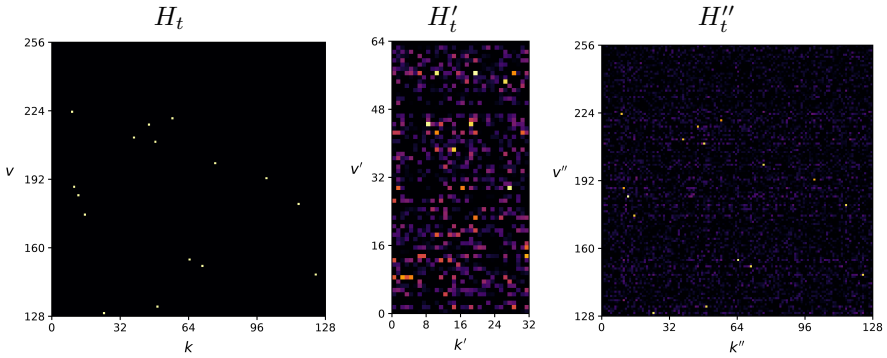


Figure 3: **Interpreting the hidden state.** The context key-value information H_t (left) is compressed by the model into a hidden state H'_t (middle), which lacks clear interpretability. As described in Alg. 2, projection onto vocabulary space (right) reveals an interpretable pattern: $H''_t \approx H_t$, indicating that a compression-decompression scheme is learned.

These weights compress both SSM input $\hat{x}_t = \binom{E\,x_{t-1}}{E\,x_t}$ and operators $B_t = \tilde{E}\,x_{t-1}$, $C_t = \tilde{E}\,x_t$, where $\tilde{E} \equiv FE \in \mathbb{R}^{N\times V}$. Similarly to Eq. 6, the overall model output is now:

$$y_t = E^\top P_{\text{out}}\,\hat{y}_t = \sum_{\tau=0}^{t} E^\top (E\,x_\tau)\,(\tilde{E}\,x_{\tau-1})^\top (\tilde{E}\,x_t) \tag{8}$$

Applying ***Johnson–Lindenstrauss lemma*** [23] (Lem. C.1, Appendix) twice, for $D = O\big(\frac{\log V}{\varepsilon_v^2}\big)$, $N = O\big(\frac{\log V}{\varepsilon_k^2}\big)$, we can construct approximately-orthogonal JL matrices E, $\tilde{E}$ (with $\tilde{E} = FE$; see Lem. C.3) such that $(\tilde{E}\,x_{\tau-1})^\top (\tilde{E}\,x_t) \approx \delta_{x_{\tau-1},\,x_t} + O(\varepsilon_k)$ and $E^\top (E\,x_\tau) \approx x_\tau + O(\varepsilon_v)$. Keys are unique, so the matching term contributes $v^* + O(\varepsilon_v + \varepsilon_k)$, while each of the $N_f - 1$ non-matching terms leaves an $O(\varepsilon_v\varepsilon_k)$ residue. For large N_f, these $O(\varepsilon_v\varepsilon_k)$ terms dominate; in the worst case, the residues add up to $O(N_f\,\varepsilon_v\varepsilon_k)$, while in the mean case they cancel down to $O\big(\sqrt{N_f}\,\varepsilon_v\varepsilon_k\big)$. Thus, to solve the task, $\sqrt{ND} = O(N_f \log V)$ is required in the worst case, while $ND = O(N_f \log V)$ suffices with high probability. (See Lem. 4.1 for the exact worst-case conditions, and Thm. 4.2 for the exact high-probability law.) □

3.2 Validating the Circuit: Mechanistic Interpretability Experiments

Equivalent recall circuits. When conducting a mechanistic examination of Mamba solution to MQAR, we are faced with a problem: even if the circuit from Thm. 3.2 is indeed the exact learned one, it is rather unlikely to find these specific weights naturally within a trained model, since the *solution space* is wider. For example, if we suspect that weights E and P_{in} transform x_t to $P_{\text{in}}E\,x_t$ through a specific operation, an equivalent operation would occur with $P'_{\text{in}} = P_{\text{in}}Q$ and $E' = Q^\top E$ for any orthogonal matrix Q. The evidence we seek for the circuit must be *invariant* under these orthogonal linear transformations. Such invariant operators are presented next. Given the simplified model from Alg. 1, without any assumption on the weights, we define the following effective operators,

$$\begin{gathered}\hat{E}_{\text{in}} = \big(\,\text{diag}(W^0_{\text{conv}})\,P_{\text{in}}\,E \;\big|\; \text{diag}(W^1_{\text{conv}})\,P_{\text{in}}\,E\,\big)\,,\quad \Pi_{v,\text{in}} = \hat{E}_{\text{in}}\,,\quad \Pi_{v,\text{out}} = E^\top P_{\text{out}}\,,\\ \Pi_{k,\text{in}} = S_B\,\hat{E}_{\text{in}}\,,\quad \Pi_{q,\text{in}} = S_C\,\hat{E}_{\text{in}}\,,\quad G_{vv} = \Pi_{v,\text{out}}\,\Pi_{v,\text{in}}\,,\quad G_{kq} = \Pi^\top_{k,\text{in}}\,\Pi_{q,\text{in}}\end{gathered} \tag{9}$$

where we denote $W_{\text{conv}} = \big(W^0_{\text{conv}} \mid W^1_{\text{conv}}\big)$. Using these net operators, the model output is:

$$y_t = \sum_{\tau=0}^{t} G_{vv}\,\xi_\tau\,\xi_\tau^\top G_{kq}\,\xi_t \tag{10}$$

where $\xi_t \equiv \binom{x_{t-1}}{x_t}$ are input token pairs. If we modify model weights such that G_{vv}, G_{kq} are unchanged, we construct an *equivalent recall circuit* with the same performance. Therefore, if the solutions learned in practice are equivalent to the one presented in Theorem 3.2, we predict that the model weights will satisfy:

$$G_{vv} \approx \begin{pmatrix}0\\ I_V\end{pmatrix}\,,\quad G_{kq} \approx \begin{pmatrix}0 & 0\\ I_V & 0\end{pmatrix}\,, \tag{11}$$

matching G_{vv}, G_{kq} computed for the designed circuit weights. In Fig. 15 (Appendix) we can see that indeed, for a linear Mamba trained on MQAR (see App. K for implementation details), the solution weights satisfy this structure almost precisely. Specifically, we see that indeed G_{kq} attends only to the query x_t and the stored key $x_{\tau-1}$, while G_{vv} attends only to the stored values x_τ; all other matrix entries are ≈ 0. Yet, note that the nonzero blocks $G_{vv}^{t,\tau}$ and $G_{kq}^{\tau-1,t}$ present more complex patterns than I_V. As discussed in App. F.3, these patterns are responsible for key-value selectivity.

Hidden state as a hash table. Intuitively, for a perfect recall, the SSM hidden state must encode the full context *information*. This fact-pairs information can be thought of as a simple key-value array, where each key entry contains 1 in the corresponding fact value v_τ index, if exists, else 0 anywhere. The ideal, non-realistic, circuit of Thm. 3.1 suggests such storage via an outer product key-value memory (see definitions and dimensions in App. D.2):

$$H_t \equiv P_{\text{out}}\, h_t = \sum_{n=1}^{N_f} v_n k_n^\top, \quad y_t = H_t\, q_t = \sum_{n=1}^{N_f} v_n k_n^\top q_t \tag{12}$$

This is a $V \times V$ table, quite similar to the array presented above, though with values represented as one-hot vectors. A visualization can be found in Fig. 3 (left). H_t is fairly interpretable, as entries are simply 1 where facts exist and 0 elsewhere. Can we observe such interpretable patterns within trained models? Let us consider the compressive SSM update from Eq. 8. It can be rewritten as

$$H'_t = \sum_{n=1}^{N_f} v'_n {k'_n}^\top, \quad y_t = E^\top \sum_{n=1}^{N_f} v'_n {k'_n}^\top q'_t, \tag{13}$$

with compressed tokens $v'_n = E\, v_n$, $k'_n = \tilde{E}\, k_n$, $q'_t = \tilde{E}\, q_t$.
H'_t can be viewed as a ***hash table***: (i) the model uses JL linear projections, which are by definition similarity-preserving hash functions; (ii) the internal representation is not directly interpretable (Fig. 3, middle), while the final output is (right); (iii) efficiency is achieved through approximation, with error bounded with high probability. The interface - hash, store, match, retrieve - closely mirrors a classical hash table (see App. D.3).

Inspecting the hash table. We notice that H'_t stores the full $V \times V$ information in a compressed $D \times N$ table, in a lossy yet decompressable way. To verify this mechanism in practice, we attempt to *decompress* the whole table, thus project it back onto vocabulary space (see [46, 47]). Let us define the decompressed tokens, $v''_n = E^\top E\, v_n \approx v_n$ and $k''_n = \tilde{E}^\top \tilde{E}\, k_n \approx k_n$. Using them, we can re-formulate the model operation as (see detailed definition in App. D.2):

$$H''_t = \sum_{n=1}^{N_f} v''_n {k''_n}^\top, \quad y_t = H''_t\, q_t = \sum_{n=1}^{N_f} v''_n {k''_n}^\top q_t \tag{14}$$

By this interpretation, the model is thought as using the full, sparse $V \times V$ table, which is an approximation of the non-compressive one: $H''_t = E^\top H'_t \tilde{E} \approx H_t$. Unfortunately, due to the non-uniqueness of solutions, as discussed in Sec. 3.2, these actual operators do not necessarily exist as-is (see App. D.4 for details). Instead, one has to project the hidden state through a more general transformation, as described in Alg. 2. This way, we can verify that indeed, as seen in Fig. 3, the learned solution does leverage a hash-like mechanism. The actual SSM hidden state H'_t is a random-looking efficiently compressed storage, which, when projected back to vocabulary space, seem to preserve the information stored in the context, as $H''_t \approx H_t$.

Conv1D as copy and shift. An important claim in Thm. 3.2 is that the Conv1D applies copy and shift operations on the input sequence. In practice, the Conv1D weights cannot be observed directly, as they are mixed with other linear transformations. However, we can find evidence to its operation within G_{vv} and G_{kq}. If the Conv1D performs copy and shift, we expect $G_{vv}\, \xi_\tau \approx x_\tau$ and $G_{kq}^\top\, \xi_\tau \approx \xi_{\tau-1}$. This can be verified given an MQAR sequence x_t, as visualized in Fig. 14 (Appendix). As expected, $x_t\, G_{vv}\, \xi_\tau \approx \delta_{t,\tau}$ and $\xi_t\, G_{kq}^\top\, \xi_\tau \approx \delta_{t,\tau-1}$. Lastly, in Tab. 8 we validate the importance of the shifting operation through an ablation test. We find that a simplified Mamba block containing only a SSM and a Conv1D of length 2 achieves nearly perfect recall, but fails completely once the Conv1D is removed.

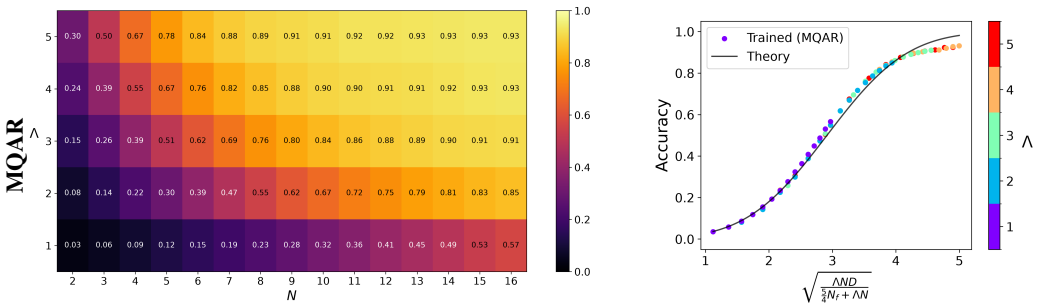


Figure 4: **Multi-layer recall scaling laws.** Recall accuracy depends only on effective state size $N_{\text{eff}} = \Lambda N$. **Left**: N-Λ accuracy grid demonstrating the N-Λ inverse relation. **Right**: In match with Thm. 4.6, accuracy collapses onto a curve of the form $p_{\text{recall}}(x) \approx \Phi(x - b)$. (See full figure in App. A.)

4 Mamba Recall Scaling Laws

The analysis in the Sec. 3 identifies the recall circuit in Mamba and validates it through mechanistic experiments. However, it leaves us with an open question: how do recall capabilities vary as we change the model scale? In this section, we attempt to analytically answer this core question.

4.1 JL-Based Recall Scaling Laws

In Thm. 3.2 we suggest a mechanism that enables Mamba to approximately perform associative recall. The conclusion though, lacks a quantitative aspect. The following Lemma 4.1 addresses the question what model dimensions D, N are required to solve an MQAR task of parameters V, N_f, L. Intuitively, the proof interprets the hidden state as information derived from a collection of approximately orthogonal hashed representations. These representations are constructed through two successive applications of the *JL lemma* (Lem. C.1; also see Lem. C.3). First, keys, queries and values in vocabulary space $\mathbb{R}^V$ are embedded into $\mathbb{R}^D$. Then, keys and queries are further compressed into the state space $\mathbb{R}^N$. Finally, for sufficiently large dimensions D, N, successful retrieval with high probability is guaranteed through a statistical analysis that bounds the error introduced at each compression step.

Lemma 4.1 (JL-based recall scaling laws). *Given a vocabulary size V, a single-layer simplified Mamba with dimensions $D = \frac{4 \log V}{\varepsilon_v^2}$, $N = \frac{4 \log V}{\varepsilon_k^2}$, $\text{expand} = 2$, $D_{\text{conv}} = 2$ can perfectly solve an AR task if model dimensions satisfy $0 < \varepsilon_v < 1$, $0 < \varepsilon_k < 1$ and $\varepsilon_v + \varepsilon_k + 2N_f\, \varepsilon_v\, \varepsilon_k < \frac{1}{2}$. The model can perfectly solve an MQAR task if model dimensions further satisfy $\varepsilon_v + \varepsilon_k + L\, \varepsilon_v\, \varepsilon_k < \frac{1}{2}$. (See proof in App. F.1.)*

4.2 Probabilistic Recall Scaling Laws

The JL-based scaling laws developed in Lemma 4.1 provide a useful bound, quantifying model scales D, N required for perfect recall. Yet, a simple summation of the $\varepsilon_k\, \varepsilon_v$ noise terms might be too strict. In practice, some of these noise terms are of opposite signs, thus cancel each other. This leads us to the following more precise probabilistic scaling laws.

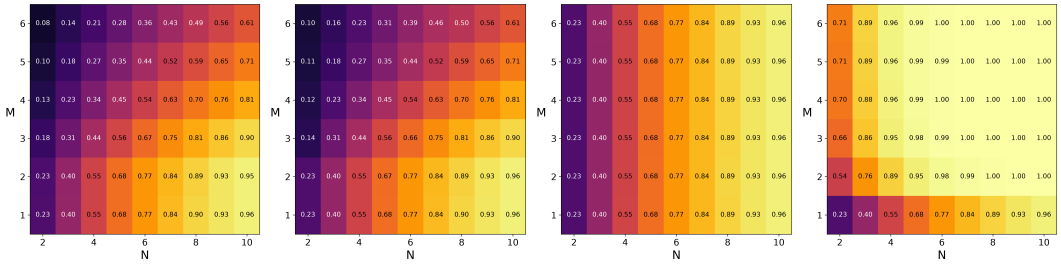
Designed model baseline. Thm. 3.2 describes a recall circuit based on general JL matrices E and $\tilde{E} = FE$. In App. E.1, we suggest a specific weight construction for E, F. The resulting model serves as a sub-optimal baseline solution for associative recall tasks, which is simpler for mathematical analysis. In Thm. F.2 (Appendix), we analyze this baseline model performance on AR and MQAR. Building upon this result, we proceed to analyze a general linear model.

Theorem 4.2 (Trained model recall scaling laws). *A single-layer simplified Mamba can solve AR and MQAR tasks with probabilities*

$$p_{\text{AR}} \approx \Phi\left(\sqrt{\frac{ND}{N_f}} - \sqrt{2 \log V}\right), \quad p_{\text{MQAR}} \approx \frac{1}{L - 2N_f} \sum_{t=2N_f}^{L} \Phi\left(\sqrt{\frac{ND}{\frac{1}{2}N_f + \frac{1}{4}t}} - \sqrt{2 \log V}\right), \quad (15)$$

in the limits of $N_f \gg N, D$ and $V \gg 1$, where Φ is the normal CDF. (See proof in App. F.4.)

Lastly, we note the above result can be further accurated and brought to a unified form.



(a) MQA (b) MKA (c) MVA (d) MHA

Figure 5: **Multi-head recall scaling laws.** Recall accuracy on MQAR for varying per-head state size N, number of heads M, and multi-head SSM pattern. Total channel dimension D is held fixed. Consistent with Thm. 4.7, (c) MVA matches the single-head baseline, while (a) MQA and (b) MKA are weaker due to shared-value compression across heads; only (d) MHA improves recall accuracy by increasing the effective state size.

Remark 4.3 (**Unified recall scaling laws**). A *single-layer simplified Mamba* solves associative recall tasks with probability $p_{\text{recall}} \approx \Phi\left(\sqrt{\frac{ND}{aN_f+N}} - \sqrt{2\log V}\right)$, where $a = 1$ for AR, and $a \approx \frac{5}{4}$ for MQAR. (See exact assumptions and proof in App. F.5.)

Remark 4.4 (**Scaling laws for model dimensions**). To solve AR with probability $\geq 1-\delta$, model dimensions must satisfy $ND = \Omega\big(N_f \log\big(\frac{V}{\delta}\big)\big)$. (See derivation in App. F.6.)

Importantly, Rem. 4.4 is a special case of a broader information-theoretic constraint, which holds for any fixed-state recurrent model, as derived in the following Lemma.

Lemma 4.5 (**Information-theoretic lower bound on state size**). *Consider a recurrent model with a finite state size, where the context is summarized in a hidden state of S scalar entries at b bits of precision each. If such a model solves an AR task with N_f facts (or an MQAR task with N_f facts and context length $L = 4N_f$) over a vocabulary of size V with success probability $> 1-\delta$, for a constant $0 < \delta < 1$, then $bS = \Omega(N_f \log V)$. (See proof in App. F.7.)*

Specifically for Mamba, at fixed precision b, this reads $ND = \Omega(N_f \log V)$ (see Rem. F.8, Appendix), in agreement with Rem. 4.4.

4.3 Multi-Layer and Multi-Head Scaling Laws

Multi-layer models. Given the single-layer circuit and scaling laws, we ask whether these findings extend to deeper architectures.

Theorem 4.6 (**Multi-layer recall scaling laws**). *A simplified multi-layer Mamba with Λ layers can solve associative recall tasks with a probability $p_{\text{recall}} \approx \Phi\left(\sqrt{\frac{\Lambda ND}{aN_f+\Lambda N}} - \sqrt{2\log V}\right)$, where $a = 1$ for AR, and $a \approx \frac{5}{4}$ for MQAR. (See proof in App. G.2.)*

Multi-head models. One of the key architectural innovations introduced in Mamba-2 [48] is the ability to process the input sequence through multiple SSM heads in parallel. Analogously to attention head patterns in Transformers, the authors present several *head patterns* for SSMs (see Tab. 5, Appendix); here we ask which are most beneficial for recall.

Theorem 4.7 (**Multi-head recall scaling laws**). *Given fixed model dimensions D, N, single-layer simplified Mamba-2 models with $M > 1$ heads and multi-head attention patterns MQA, MKA, MVA, or MHA solve recall tasks (AR or MQAR) with probabilities satisfying $p_{\text{MQA}} = p_{\text{MKA}} \leq p_{\text{single}} = p_{\text{MVA}} \leq p_{\text{MHA}}$, where p_{single} is the recall probability for a single-head model ($M = 1$) of similar dimensions D, N, as detailed in Thm. 4.2. (See proof in App. H.1.)*

5 Scaling Laws Experiments

5.1 Single-Layer Model: Accuracy Grids

We train simplified single-layer Mamba (linear and full) with varying D, N across multiple AR and MQAR settings. See further details in App. K. Each grid point is trained with multiple random seeds; we report the *best-of-seeds* accuracy, as our goal is to measure model *capacity* - the existence of a recall solution - rather than the probability of finding it via optimization (see App. K.5).

Comparison with theory. Our raw results are summarized in Fig. 1 (top). Trained linear (a, b) and full (c) models all exhibit the inverse D-N tradeoff predicted by Rem. 4.4, and the full model closely

matches the linear one, confirming that our simplification preserves the core recall behavior. Fig. 6 (Appendix) adds designed models with fixed weights (Thm. F.2) and the theoretical grid of Thm. 4.2, which closely align with the trained linear model.

5.2 Single-Layer Model: Scaling Curves

Predicting recall accuracy. Fig. 1 (bottom) shows that in both AR and MQAR settings, the D-N accuracy grids exhibit a gradual transition from failure to near-perfect recall. This transition is well captured by a one-dimensional scaling variable $x = \frac{1}{\sigma} = \sqrt{\frac{ND}{aN_f+N}}$, where σ^2 is the theoretical variance of the model output y_t, and results closely follow $p_{\text{recall}}(x) \approx \Phi(x - \sqrt{2 \log V})$. Comparing columns (a) and (b), MQAR shows a larger variance (larger $a \approx \frac{5}{4}$), due to the increased number of queries, making high recall harder to achieve at the same D, N. Both AR and MQAR follow the same underlying recall scaling law, apparent after appropriate rescaling. We further validate the V scaling law under more realistic vocabulary sizes (up to $V = 50$K) and context lengths (up to $L = 4$K). Our results align with the theoretical scaling laws; see details in Figs. 9 and 10 (Appendix).

Full model scaling behavior. In Fig. 1 (bottom), column (c) shows that a full nonlinear Mamba model also fits Rem. 4.3, with $a_{\text{full}} \approx \frac{1}{2} a_{\text{linear}}$ (determined heuristically), indicating improved performance under a similar scaling law. This suggests both simplified and full models share a hash-like key-value compression bottleneck, with N, D as the primary capacity constraints. We leave a detailed investigation of this hypothesis for future work.

5.3 Multi-Layer Model Experiments

The effect of additional layers. Experiment results are summarized in Fig. 4. Consistent with Thm. 4.6, measured recall accuracy is determined not by N and Λ separately, but by their product $N_{\text{eff}} \equiv \Lambda N$ (an *effective* state size), quantifying the contribution of additional layers.

Scaling curves. Similarly to the single-layer case, accuracy collapses onto a single curve independently of depth Λ, closely following $p_{\text{recall}}(x) \approx \Phi(x - \sqrt{2 \log V})$; note that $x = \sqrt{\frac{\Lambda ND}{aN_f+\Lambda N}}$, provided by Thm. 4.6, exactly explains the observed results.

5.4 Multi-Head Model Experiments

The effect of multi-head SSM patterns. We train *simplified multi-head Mamba-2* models on MQAR, varying N and M while keeping D fixed (Fig. 5). Consistent with Thm. 4.7, multi-head structure improves recall only when it enlarges both key and query dimensions, which occurs only in MHA - at the cost of more trainable parameters. MQA and MKA achieve worse accuracy due to shared-value compression across heads.

Connection to language-modeling performance. Interestingly, the trends in Thm. 4.7 are consistent with the perplexity ablations for full Mamba-2 [48, Sec. 9.4, Table 5]: MQA and MKA both yield worse perplexity than MVA, matching our finding that shared-value compression degrades recall. When controlling for parameter count, MVA is the stronger design, thus chosen in practice. This suggests our analysis may capture the same tradeoffs observed in full Mamba models in real-world NLP tasks. We further evaluate zero-shot AR performance on pretrained Mamba-2 models of 370M to 2.7B parameters, in the spirit of Ben-Kish et al. [5], and find similar trends: accuracy decays with the number of facts at every size, and the largest state retains the most (Fig. 11, Appendix).

6 Conclusions

In this paper, we identify the neural algorithm underlying associative recall in Mamba models. We build on this understanding to develop a theoretical framework that predicts, given model dimensions, the maximum vocabulary size and number of facts over which accurate recall can be achieved in the MQAR task. Our theoretical predictions are shown to closely align with empirical results.

Limitations. Our theory holds for a simplified, linear variant of Mamba; the agreement of full Mamba models with it is empirical evidence rather than a formal equivalence. We treat gating, nonlinearities, and discretization only through the constants in our bounds; a full theoretical account of their role is a direction for future research. Likewise, direct transfer of our scaling laws from synthetic tasks to language modeling remains to be established [5], though zero-shot recall in pretrained Mamba-2 models follows the same qualitative trends (Sec. 5.4, App. A.3). Future work includes extending the analysis to other architectures, such as xLSTM [49], RWKV [2] and DeltaNet [50].

Acknowledgments and Disclosure of Funding

This work was supported by a grant from the Tel Aviv University Center for AI and Data Science (TAD). Additionally, this research was also supported by the Ministry of Innovation, Science & Technology, Israel (1001576154) and the Michael J. Fox Foundation (MJFF-022407). The authors declare no competing interests.

References

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

[2] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. RWKV: Reinventing RNNs for the Transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[3] Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

[4] Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not Transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Assaf Ben-Kish, Itamar Zimerman, M Jehanzeb Mirza, Lior Wolf, James Glass, Leonid Karlinsky, and Raja Giryes. Overflow prevention enhances long-context recurrent LLMs. *arXiv preprint arXiv:2505.07793*, 2025.

[6] Rom N Parnichkun, Neehal Tumma, Armin W Thomas, Alessandro Moro, Qi An, Taiji Suzuki, Atsushi Yamashita, Michael Poli, and Stefano Massaroli. Quantifying memory utilization with effective state-size. *arXiv preprint arXiv:2504.19561*, 2025.

[7] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

[8] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

[9] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] Aryaman Arora, Neil Rathi, Nikil Roashan Selvam, Róbert Csordás, Dan Jurafsky, and Christopher Potts. Mechanistic evaluation of Transformers and state space models. *arXiv preprint arXiv:2505.15105*, 2025.

[11] Wangjie You, Zecheng Tang, Juntao Li, Lili Yao, and Min Zhang. Revealing and mitigating the local pattern shortcuts of Mamba. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12156–12178, 2025.

[12] Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

[13] Aviv Bick, Eric Xing, and Albert Gu. Understanding the skill gap in recurrent language models: The role of the gather-and-aggregate mechanism. *arXiv preprint arXiv:2504.18574*, 2025.

[14] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[15] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[16] Asher Trockman, Hrayr Harutyunyan, J Zico Kolter, Sanjiv Kumar, and Srinadh Bhojanapalli. Mimetic initialization helps state space models learn to recall. *arXiv preprint arXiv:2410.11135*, 2024.

[17] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional

language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[18] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for Transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[19] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

[20] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[21] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[22] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[23] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26:189–206, 1984.

[24] Shahar Lutati, Itamar Zimerman, and Lior Wolf. Focus your attention (with adaptive IIR filters). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12538–12549, 2023.

[25] Destiny Okpekpe and Antonio Orvieto. Revisiting associative recall in modern recurrent models. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025.

[26] Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to optimally memorize the context at test time. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, volume 306 of *Proceedings of Machine Learning Research*, pages 7361–7390, 2026.

[27] Grégoire Le Corre, Ningyuan Huang, and Alberto Bietti. How does Mamba perform associative recall? A mechanistic study. In *NeurIPS 2025 Workshop on Mechanistic Interpretability*, 2025. URL https://openreview.net/forum?id=ur17K8zXcB.

[28] Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

[29] Eshaan Nichani, Jason D. Lee, and Alberto Bietti. Understanding factual recall in Transformers via associative memories. In *The Thirteenth International Conference on Learning Representations*, 2025.

[30] Ningyuan Huang, Miguel Sarabia, Abhinav Moudgil, Pau Rodriguez, Luca Zappella, and Federico Danieli. Understanding input selectivity in Mamba: Impact on approximation power, memorization, and associative recall capacity. *arXiv preprint arXiv:2506.11891*, 2025.

[31] Quentin Anthony, Yury Tokpanov, Paolo Glorioso, and Beren Millidge. BlackMamba: Mixture of experts for state-space models. *arXiv preprint arXiv:2402.01771*, 2024.

[32] Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. MambaByte: Token-free selective state space model. *arXiv preprint arXiv:2401.13660*, 2024.

[33] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

[34] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[35] Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-Mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

[36] Edo Cohen-Karlik, Itamar Zimerman, Liane Galanti, Ido Atad, Amir Globerson, and Lior Wolf. On the expressivity of selective state-space layers: A multivariate polynomial approach. *arXiv preprint arXiv:2502.02209*, 2025.

[37] Guy E Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

[38] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[39] Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. DeciMamba: Exploring the length extrapolation potential of Mamba. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.14528.

[40] Zhifan Ye, Kejing Xia, Yonggan Fu, Xin Dong, Jihoon Hong, Xiangchi Yuan, Shizhe Diao, Jan Kautz, Pavlo Molchanov, and Yingyan Celine Lin. LongMamba: Enhancing Mamba's long context capabilities via training-free receptive field enlargement, 2025. URL https://arxiv.org/abs/2504.16053.

[41] Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=CqViN4dQJk.

[42] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

[43] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, pages 6243–6267. PMLR, 2023.

[44] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, 2023.

[45] Roy Eisenstadt, Itamar Zimerman, and Lior Wolf. Overclocking LLM reasoning: Monitoring and controlling thinking path lengths in LLMs. *arXiv preprint arXiv:2506.07240*, 2025.

[46] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing Transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, 2023.

[47] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, 2023.

[48] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[49] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37: 107547–107603, 2024.

[50] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear Transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[51] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of Mamba models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1516–1534, 2025.

[52] PeaBrane. mamba-tiny, 2024. URL https://github.com/PeaBrane/mamba-tiny. Minimal Mamba implementation.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[54] OpenAI. ChatGPT (GPT-5), 2025. URL https://chat.openai.com/. Large language model developed by OpenAI.

[55] Anthropic. Claude, 2025. URL https://www.anthropic.com/claude. Large language model developed by Anthropic.

Appendix Contents

A Additional Experiments and Figures **15**
A.1 Scaling Laws Experiments . . . 15
A.2 Long Context and Large Vocabulary . . . 17
A.3 Zero-Shot AR in Natural Language Pretrained Mamba . . . 17
A.4 Reverse Engineering Mamba on MQAR . . . 18

B Notation and Dimensions **22**
B.1 Multi-Query Associative Recall (MQAR) . . . 22
B.2 Mamba Model . . . 22
B.3 SSM State Update Rule . . . 22

C Mamba Recall Circuits: Proofs **23**
C.1 Johnson-Lindenstrauss (JL) Lemma . . . 23
C.2 Non-Compressive Recall Circuit . . . 24
C.3 Compressive Recall Circuit . . . 25

D Mechanistic Interpretability of the Recall Circuit: Supplementary Material **28**
D.1 Invariant Operators . . . 28
D.2 SSM Hidden State and Key-Value Tables . . . 28
D.3 Hidden State as a Hash Table . . . 29
D.4 Inspecting the Hash Table . . . 30
D.5 Noisy Attention Maps . . . 30

E Designed Model Baseline **30**
E.1 Designed Model Weights . . . 30

F Mamba Recall Scaling Laws: Proofs **31**
F.1 JL-Based Scaling Laws . . . 31
F.2 Probabilistic Scaling Laws: Designed Model . . . 33
F.3 Probabilistic Scaling Laws: Trained Model . . . 36
F.4 Unified Recall Scaling Laws . . . 36
F.5 Information-Theoretic Lower Bound for Model Dimensions . . . 39

G Multi-Layer Recall Circuits **39**
G.1 Multi-Layer Mamba . . . 39
G.2 Multi-Layer Recall Circuit and Scaling Laws . . . 39

H Multi-Head Recall Circuits and Mamba-2 **42**
H.1 Multi-Head Patterns . . . 42
H.2 Multi-Head Recall Circuits . . . 43
H.3 Notation and Dimensions . . . 46

I Ablation Studies and Generalizations **47**
I.1 Recall Circuit Minimality Ablation . . . 47
I.2 MQAR with Repeated Keys . . . 48

J Similarities and Differences with *Understanding Input Selectivity in Mamba* **48**

K Training and Implementation Details **50**
K.1 Datasets . . . 50
K.2 Models . . . 50
K.3 Training Hyperparameters . . . 50
K.4 Compute and Environment . . . 50
K.5 Seeds Aggregation . . . 51

L Statements and Declarations **52**
L.1 Impact Statement . . . 52
L.2 Reproducibility Statement . . . 52
L.3 Licenses . . . 52
L.4 The Use of Large Language Models (LLMs) . . . 52

A Additional Experiments and Figures

A.1 Scaling Laws Experiments

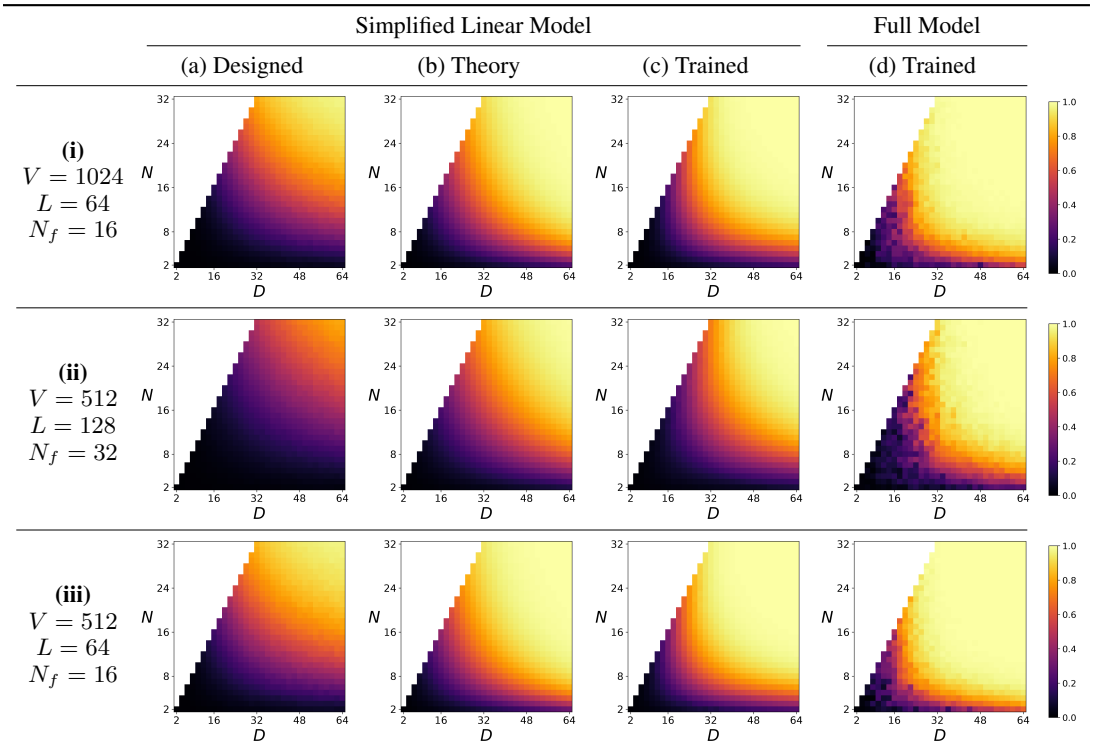


Figure 6: **Mamba recall scaling laws: theory versus empirical results.** Recall accuracy, represented by pixel color, against model embedding dimension D and state size N. Left to right: (a) Simplified linear Mamba model with fixed weights, designed as presented in Thm. F.2. (b) Theoretical scaling law as derived in Thm. 4.2 for a trained linear model. (c) Trained linear model. (d) Trained full model. Each row represents a different regime of MQAR parameters.

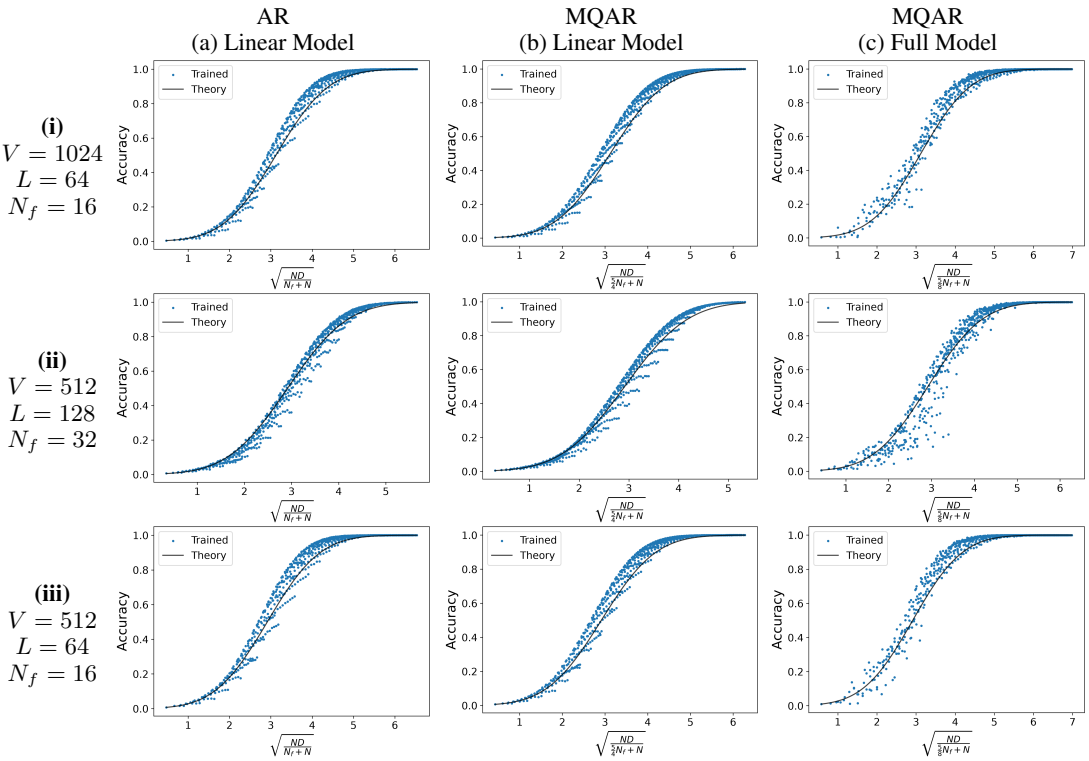


Figure 7: **Single-layer recall scaling curves across regimes.** The scaling curves of Fig. 1 (bottom), for the regimes of Fig. 6 (rows); row (i) repeats Fig. 1 (bottom). The scaling $p_{\text{recall}}(x) \approx \Phi(x - b)$ with $b \approx \sqrt{2 \log V}$ (Rem. 4.3) holds across the different regimes, with $a = 1$, $a = \frac{5}{4}$, and $a = \frac{5}{8}$ (columns, left to right) consistent across regimes.

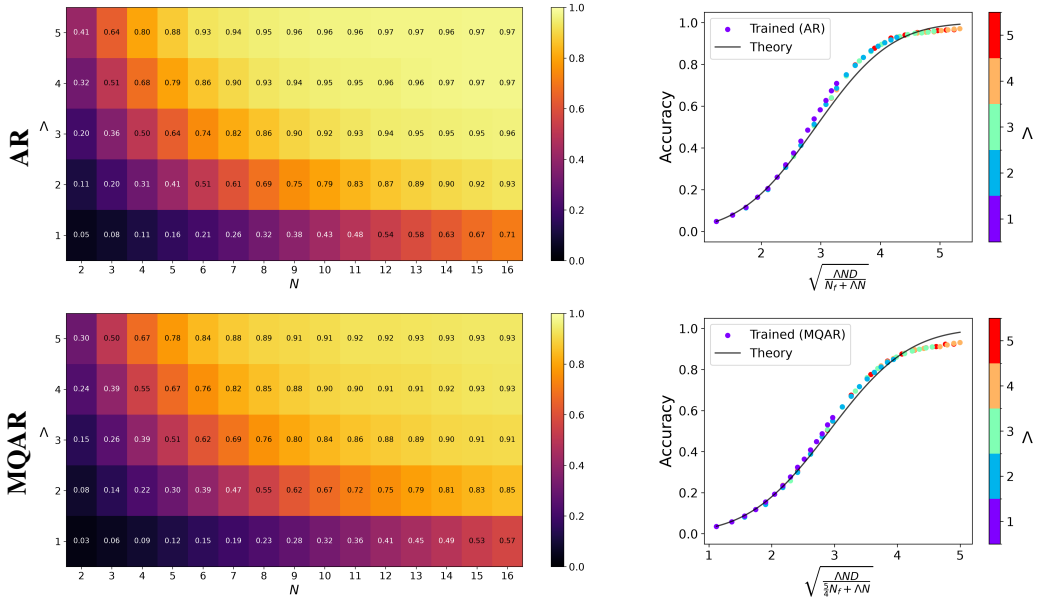


Figure 8: **Multi-layer recall scaling laws.** The recall accuracy of a simplified multi-layer Mamba (up: AR, down: MQAR) is uniquely determined by its effective state size $N_{\text{eff}} = \Lambda N$, rather than by each factor separately. **Left**: N-Λ accuracy grids. Notice that performance is the same for (N, Λ) pairs with the same N_{eff}: for example, $p_{\text{MQAR}} \approx 0.38$ for grid points $(12, 1)$, $(6, 2)$, $(4, 3)$ and $(3, 4)$. **Right**: For both AR and MQAR, we verify that empirical recall accuracy fits our theoretical prediction $p_{\text{recall}}(x) \approx \Phi(x - b)$ from Thm. 4.6, where $x = \frac{1}{\sigma}$ and $b \approx \sqrt{2 \log V}$ as detailed in Thm. 4.6.

A.2 Long Context and Large Vocabulary

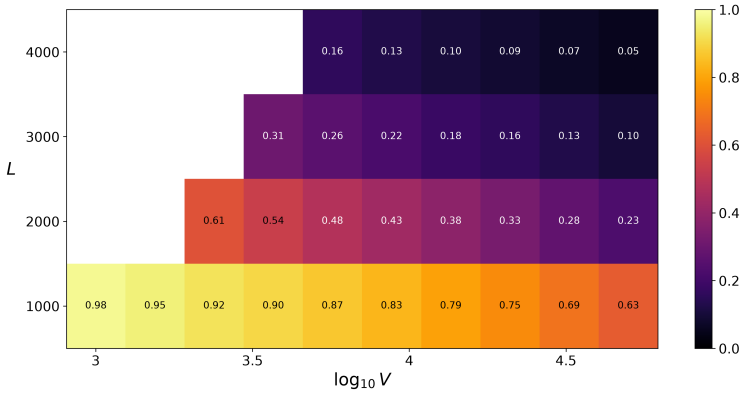


Figure 9: **Larger scale experiments.** MQAR recall accuracy of a trained linear model ($D = 150$, $N = 75$) for vocabulary sizes $V = 1000$ to $V = 50000$ (logarithmically spaced) and context lengths $L = 1000$ to $L = 4000$, with $N_f = L/4$ key-value pairs.

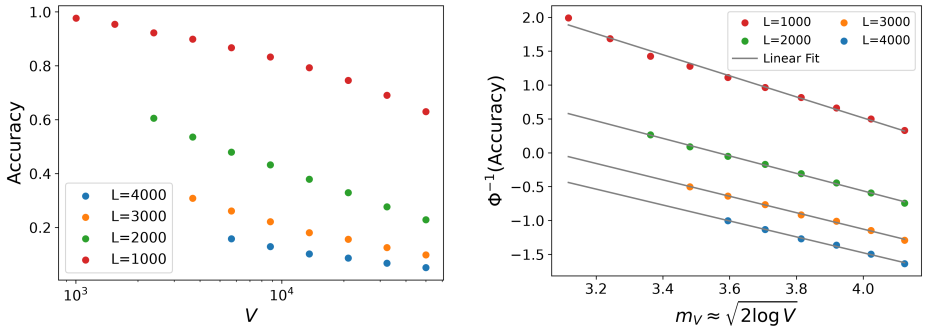


Figure 10: **Larger scale experiments: scaling curves.** The accuracy of Fig. 9 against vocabulary size (left), and its normal-inverse collapse against $m_V \approx \sqrt{2 \log V}$ (see Eq. 108) (right), per context length L. The per-L linear fits (right, gray lines) have slopes 1.18-1.56, against a theoretical slope of 1 (Rem. 4.3). This makes sense: the theoretical scaling is an approximated relation.

A.3 Zero-Shot AR in Natural Language Pretrained Mamba

We evaluate zero-shot AR in natural language on the pretrained Mamba-2 family [48] (370M, 780M, 1.3B and 2.7B parameters), using a simplified version of a protocol from [5]. Each prompt lists N_f random key-value pairs (8-letter keys, 5-digit values) and ends with one of the keys; the greedy continuation is correct if it starts with the matching value. An example with $N_f = 10$, with highlighted key, value and query tokens:

```
Complete the following sequence: bxwgemhg=64304, qcfhfxbi=12277,
ykgtjfjo=64826, gmcamfhj=91471, vgpdjtrp=14082, fvsvvjyl=99729, ejhmfjrp=16832,
whscazat=02116, iqkgfmpc=87547, qqcbobqf=67650, gmcamfhj=
```

with expected output 91471. Accuracy is averaged over 32 queries per context and 30 random contexts per N_f. Fig. 11 shows the results.

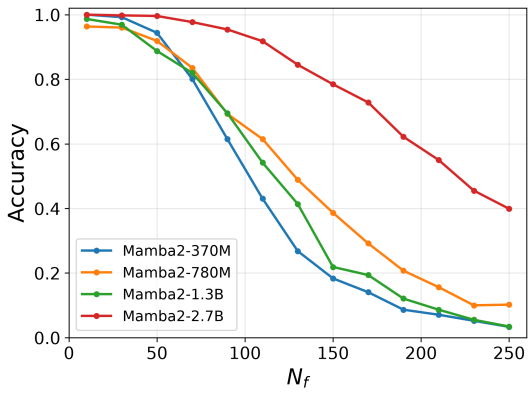


Figure 11: **Zero-shot associative recall in pretrained Mamba-2 models.** Recall accuracy against the number of facts N_f in the prompt. Each point is the mean success over 30 random contexts. The trend, recall degrades with the number of facts, is qualitatively similar to what we prove and measure for the synthetic AR tasks and AR-trained Mamba models. The largest model has the highest accuracy, as expected (while the smaller ones are not consistently ordered by size).

A.4 Reverse Engineering Mamba on MQAR

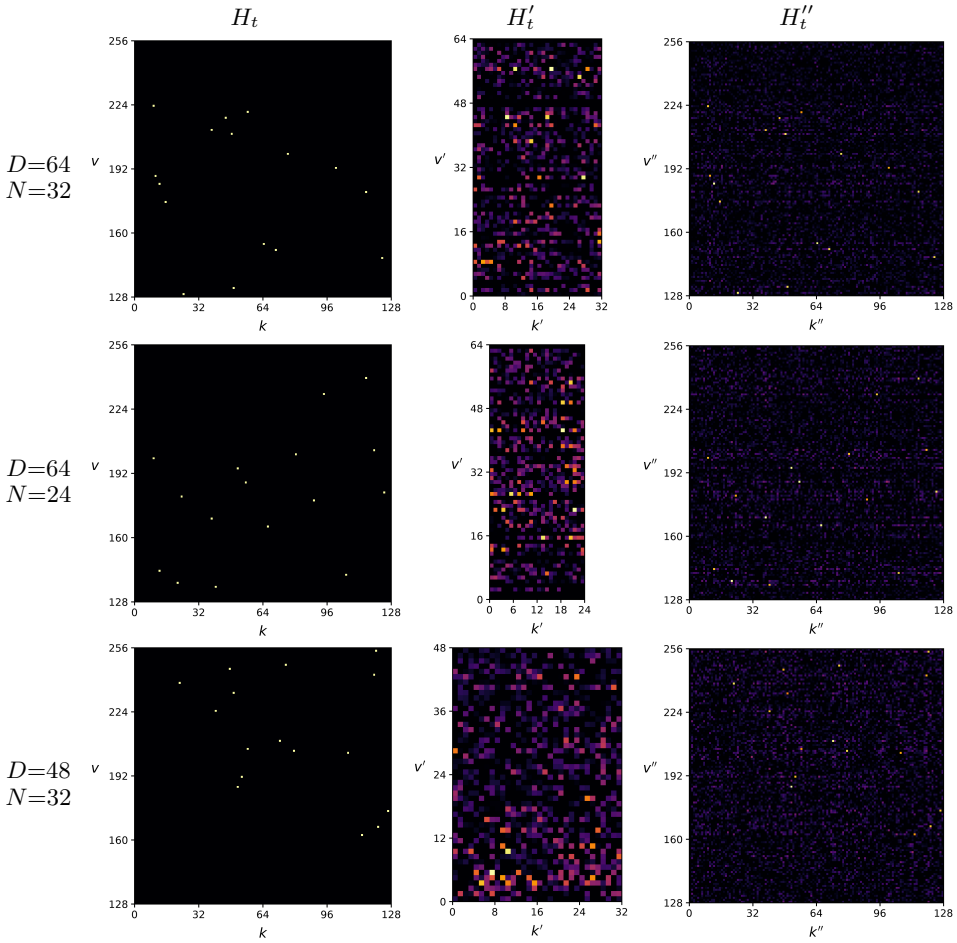


Figure 12: **Interpreting the hidden state.** The context key-value information H_t (left) is compressed by the model into a hidden state H'_t (middle), which lacks clear interpretability. As described in Alg. 2, projection onto vocabulary space (right) reveals an interpretable pattern: $H''_t \approx H_t$, indicating that a compression-decompression scheme is learned. Rows: same analysis repeated for trained models with varying D, N. Note that H and H'' are $V \times V$, while H' is $D \times N$.

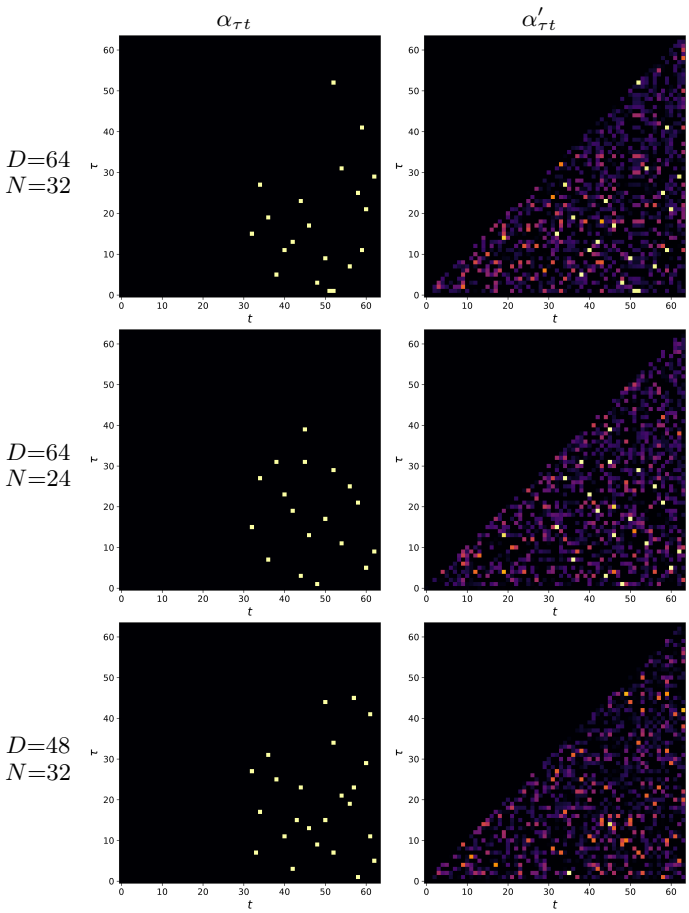


Figure 13: **Implicit attention maps.** The ideal attention map of a theoretical non-compressive model (left) is well approximated by a trained real-world model (right). Rows: same analysis repeated for trained models with varying D, N.

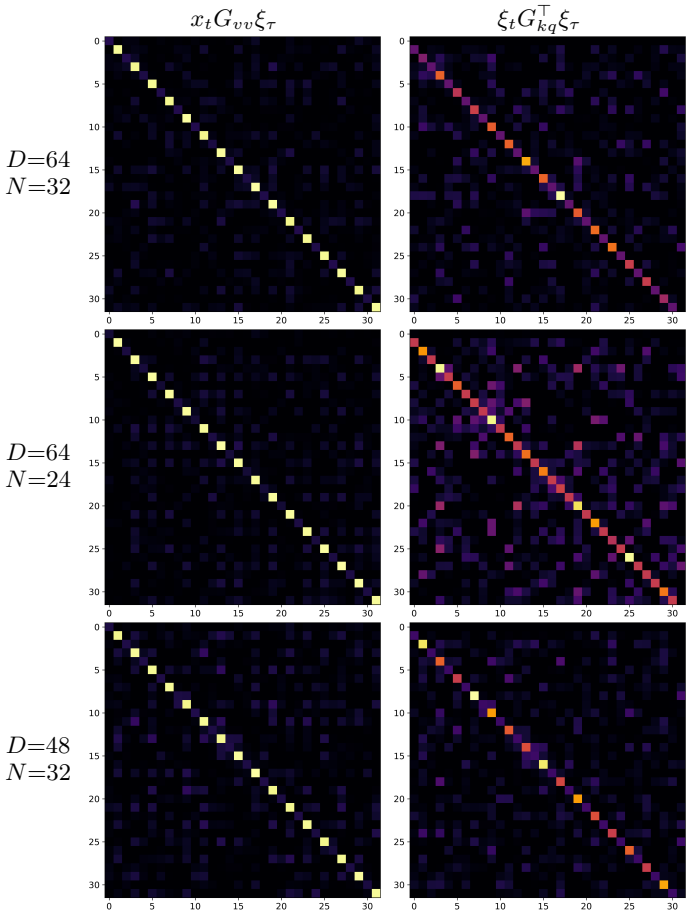


Figure 14: **Conv1D copy & shift.** Values are copied (left), as indicated by the dominant main diagonal. Keys are shifted (right), as indicated by the dominant second diagonal. Rows: same analysis repeated for trained models with varying D, N.

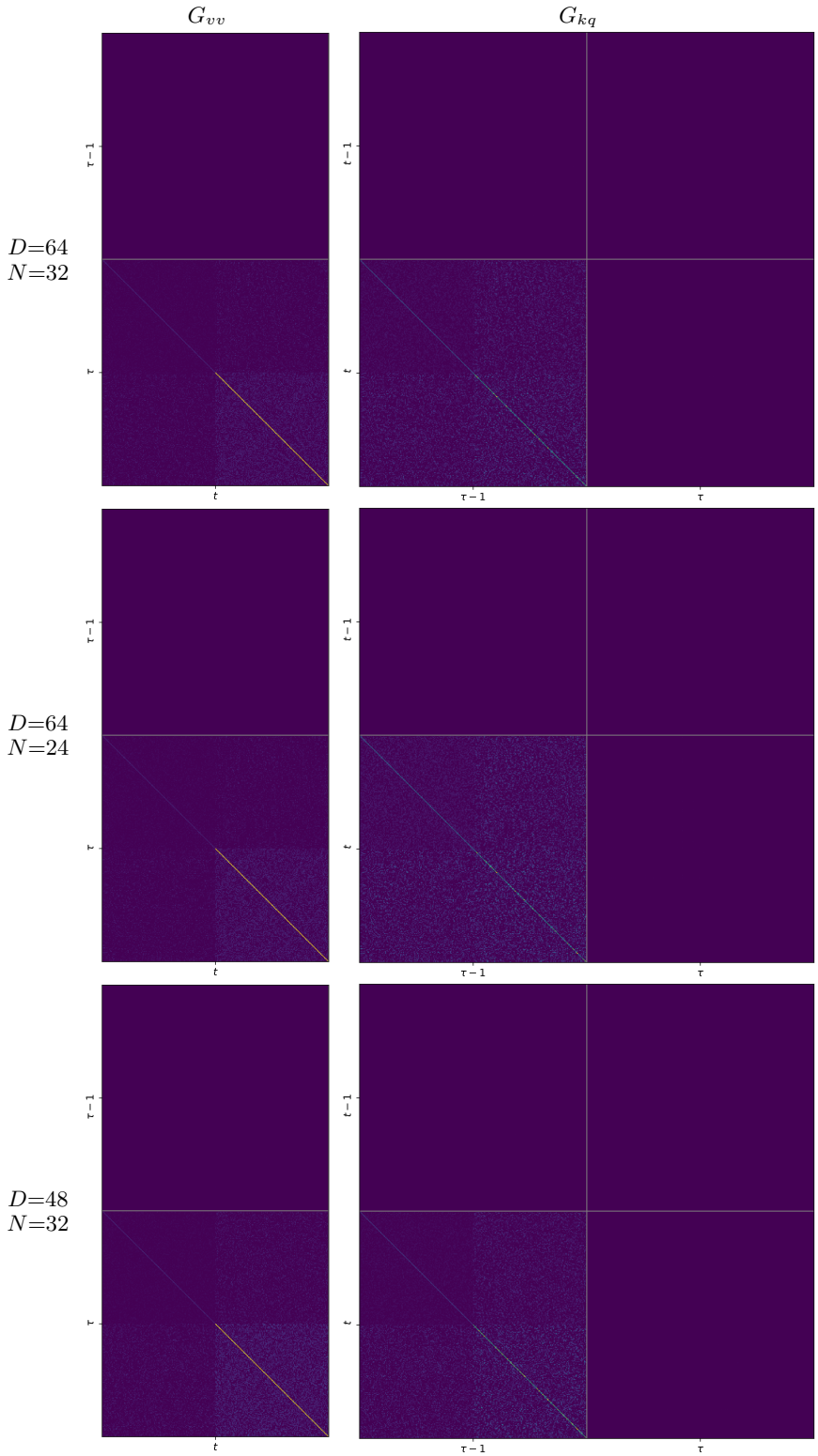


Figure 15: **Invariant operators.** The model's internal mechanism is revealed when applying the appropriate transformations. As predicted by Eq. 11, the core operation compares a query q_t with a key $k_{\tau-1}$, then outputs the corresponding value v_τ. Rows: same analysis repeated for trained models with varying D, N.

B Notation and Dimensions

This appendix provides reference tables for the notation appearing in the paper, as well as tables for tensor dimensions, as used in our algorithms and analysis.

B.1 Multi-Query Associative Recall (MQAR)

Table 1 includes notation for the MQAR benchmark.

Table 1: Notation for MQAR

Symbol	Meaning	Value	Notes
$\mathcal{V}$	Vocabulary	-	
$\mathcal{V}_k$	Key vocabulary	-	
$\mathcal{V}_v$	Value vocabulary	-	
V	Vocabulary size	$\lvert\mathcal{V}\rvert$	
V_k	Key vocabulary size	$\lvert\mathcal{V}_k\rvert$	$V_k = \frac{V}{2}$, unless stated otherwise
V_v	Value vocabulary size	$\lvert\mathcal{V}_v\rvert$	$V_v = \frac{V}{2}$, unless stated otherwise
N_f	Number of facts	-	
L	Total context length	-	$L \geq 4N_f$, following implementation [9]
-	Facts section length	$2N_f$	N_f key-value token pairs
-	Queries section length	$L - 2N_f$	N_f query tokens, $L - 3N_f$ random padding tokens

B.2 Mamba Model

Tables 2, 3 include detailed notation and dimensions for Mamba model and tensors.
For detailed dimensions of hidden state and key-value matrices H_t, H'_t, H''_t used in Sec. 3.2 for mechanistic interpretability, see Appendix D.2.
For Mamba-2 dimensions, see Appendix H.

Table 2: Notation for Mamba

Symbol	Meaning	Value	Notes
D	Embedding size	-	
N	SSM State size	-	
expand	SSM expansion factor	$\frac{D_{\text{in}}}{D}$	
D_{in}	SSM number of channels	$\text{expand} \cdot D$	
D_{conv}	Convolution kernel size	-	
Λ	Number of layers	-	

B.3 SSM State Update Rule

In the following Appendix, we clarify shapes and dimensions for SSM state update.

Full Mamba model. For each timestep t, the SSM vectors B_t, $C_t \in \mathbb{R}^N$ and discrete matrices $\bar{A}_t, \bar{B}_t \in \mathbb{R}^{D_{\text{in}} \times N}$ are computed as:

$$B_t = S_B\,\hat{x}_t\,, \quad C_t = S_C\,\hat{x}_t\,, \quad \Delta_t = \text{SoftPlus}(S_\Delta \hat{x}_t)\,, \quad \bar{A}_t = \exp(\Delta_t A)\,, \quad \bar{B}_t = \Delta_t B_t \tag{16}$$

where S_B, S_C, S_Δ are linear projection matrices, SoftPlus is a smooth approximation of ReLU, and biases are omitted for simplicity.
Then, for any timestep t, each SSM channel $d = 1, \ldots, D_{\text{in}}$ has a scalar input $\hat{x}_t^d \in \mathbb{R}$ and a scalar output $\hat{y}_t^d \in \mathbb{R}$. The output is computed through a recurrent state-update step:

$$\begin{cases} h_t^d = \bar{A}_t^d \odot h_{t-1}^d + \hat{x}_t^d\,\bar{B}_t^d \\ \hat{y}_t^d = h_t^d \cdot C_t \end{cases} \tag{17}$$

where $h_t^d, \bar{A}_t^d, \bar{B}_t^d, C_t \in \mathbb{R}^N$ are all N-vectors, $\odot$ denotes an element-wise product, and $\cdot$ a vector dot-product. (For $\hat{x}_t^d\,\bar{B}_t^d$ we have a scalar multiplying a vector.)
Per-channel, we have a state vector $h_t^d \in \mathbb{R}^N$. In total, all channels together form a state matrix $h_t \in \mathbb{R}^{D_{\text{in}} \times N}$.

Table 3: Dimensions for Mamba

Symbol	Meaning	Dimension	Notes
x	Model input sequence (one-hot)	$\mathbb{R}^{V\times L}$	$x_t \in \mathbb{R}^V$
y	Model output sequence (one-hot)	$\mathbb{R}^{V\times L}$	$y_t \in \mathbb{R}^V$
x^e	Model input sequence (embedded)	$\mathbb{R}^{D\times L}$	$x^e_t \in \mathbb{R}^D$
$\hat{x}$	SSM input sequence	$\mathbb{R}^{D_{\text{in}}\times L}$	$\hat{x}_t \in \mathbb{R}^{D_{\text{in}}}$
$\hat{y}$	SSM output sequence	$\mathbb{R}^{D_{\text{in}}\times L}$	$\hat{y}_t \in \mathbb{R}^{D_{\text{in}}}$
$\hat{z}$	Gate sequence	$\mathbb{R}^{D_{\text{in}}\times L}$	$\hat{z}_t \in \mathbb{R}^{D_{\text{in}}}$
E	Model embedding	$\mathbb{R}^{D\times V}$	
W_{conv}	Convolution kernel	$\mathbb{R}^{D_{\text{in}}\times D_{\text{conv}}}$	
P_{in}	Input projection	$\mathbb{R}^{D_{\text{in}}\times D}$	
P_{out}	Output projection	$\mathbb{R}^{D\times D_{\text{in}}}$	
S_B	B projection	$\mathbb{R}^{N\times D_{\text{in}}}$	
S_C	C projection	$\mathbb{R}^{N\times D_{\text{in}}}$	
A	SSM transition coefficient	$\mathbb{R}^N$	
B	SSM input coefficient	$\mathbb{R}^{N\times L}$	$B_t \in \mathbb{R}^N$
C	SSM readout coefficient	$\mathbb{R}^{N\times L}$	$C_t \in \mathbb{R}^N$
Δ	SSM discretization parameter	$\mathbb{R}^{D_{\text{in}}\times L}$	$\Delta_t \in \mathbb{R}^{D_{\text{in}}}$
$\bar{A}$	SSM Discrete transition coefficient	$\mathbb{R}^{D_{\text{in}}\times N\times L}$	$\bar{A}_t \in \mathbb{R}^{D_{\text{in}}\times N}$
$\bar{B}$	SSM Discrete input coefficient	$\mathbb{R}^{D_{\text{in}}\times N\times L}$	$\bar{B}_t \in \mathbb{R}^{D_{\text{in}}\times N}$
h	SSM hidden state	$\mathbb{R}^{D_{\text{in}}\times N\times L}$	$h_t \in \mathbb{R}^{D_{\text{in}}\times N}$
α	SSM equivalent attention matrix	$\mathbb{R}^{L\times L}$	

Simplified Mamba model. In order to ease Mamba model analysis, we preset a simplified SSM, where discretization is removed, and $\bar{A}$ is ignored (by setting its values to 1). The SSM operation can be now written as:

$$\begin{cases} h^d_t = h^d_{t-1} + \hat{x}^d_t\, B_t \\ \hat{y}^d_t = h^d_t \cdot C_t \end{cases} \tag{18}$$

Note that $\bar{B}^d_t = B_t \in \mathbb{R}^N$ is now shared between channels.
If we consider the channel dimension, we have a vector input and output $\hat{x}_t, \hat{y}_t \in \mathbb{R}^{D_{\text{in}}}$, and a state matrix $h_t \in \mathbb{R}^{D_{\text{in}}\times N}$. Equation 18 can be written in a matrix form as:

$$\begin{cases} h_t = h_{t-1} + \hat{x}_t\, B_t^\top \\ \hat{y}_t = h_t\, C_t \end{cases} \tag{19}$$

where $\hat{x}_t\, B_t^\top \in \mathbb{R}^{D_{\text{in}}\times N}$ is a vector-vector outer product.

C Mamba Recall Circuits: Proofs

C.1 Johnson-Lindenstrauss (JL) Lemma

Lemma C.1 (Johnson-Lindenstrauss lemma, inner-product form [23]). *Let $x_1, \ldots, x_n \in \mathbb{R}^m$ be nonzero vectors and $0 < \varepsilon < 1$. If $d \geq \frac{c\,\log n}{\varepsilon^2}$ for an absolute constant c, then there exists a linear map $E \in \mathbb{R}^{d\times m}$ such that for all i, j:*

$$\left|\langle E\,x_i, E\,x_j\rangle - \langle x_i, x_j\rangle\right| \;\leq\; \varepsilon\,\|x_i\|\,\|x_j\| \tag{20}$$

Proof. We use the probabilistic method: draw E with i.i.d. $\mathcal{N}(0, \frac{1}{d})$ entries, and show the desired map exists since random E satisfies Eq. 20 with nonzero probability.

Norm concentration. For any fixed vector u, the entries of $E\,u$ are i.i.d. $\mathcal{N}(0, \frac{\|u\|^2}{d})$, so $\|E\,u\|^2 \sim \|u\|^2 \cdot \chi^2_d/d$, which concentrates around its mean $\|u\|^2$: a Chernoff bound on the χ^2_d distribution gives

$$\Pr\Big[\,\big|\,\|E\,u\|^2 - \|u\|^2\big| \geq \varepsilon\,\|u\|^2\Big] \;\leq\; 2\,e^{-\left(\varepsilon^2/2 - \varepsilon^3/3\right)d/2} \tag{21}$$

Polarization. Let $u_i = x_i/\|x_i\|$. Suppose the norms of u_i, u_j, $u_i{+}u_j$ and $u_i{-}u_j$ are all preserved within $(1 \pm \varepsilon)$, as in Eq. 21. Then, since $\langle E\,u_i, E\,u_j\rangle = \frac{1}{4}\big(\|E(u_i{+}u_j)\|^2 - \|E(u_i{-}u_j)\|^2\big)$, the

inner product deviates by at most $\big|\langle E\,u_i, E\,u_j\rangle - \langle u_i, u_j\rangle\big| \le \varepsilon \cdot \frac{\|u_i\|^2+\|u_j\|^2}{2} = \varepsilon$. Scaling both sides back by $\|x_i\|\,\|x_j\|$ yields Eq. 20.

Union bound. The above requires at most $2n^2$ norm events to hold simultaneously; by Eq. 21, the probability that any of them fails is at most $4n^2\, e^{-\left(\varepsilon^2/2-\varepsilon^3/3\right)d/2}$, which drops below 1 once $d \ge \frac{c\,\log n}{\varepsilon^2}$ for a suitable absolute constant c. A map satisfying Eq. 20 for all i, j therefore exists. □

Corollary for one-hot vectors. In this paper the lemma is applied to sets of one-hot vectors (and their near-orthonormal embeddings); for the one-hot case, bounding the Gram entries of E directly sharpens the constant to $c = 4$. (See for instance our usage in Lemma 4.1.)

C.2 Non-Compressive Recall Circuit

Theorem C.2 (Perfect non-compressive recall circuit). *Given a vocabulary size V, a single-layer simplified Mamba with dimensions $D = V$, $N = V$, $\text{expand} = 2$, $D_{\text{conv}} = 2$ can **perfectly solve** an MQAR task, i.e. with recall probability = 1.*

Proof. We construct model weights as follows:

Embeddings. Let us choose the trivial identity embedding $\in \mathbb{R}^{V\times V}$:

$$E = I_V \tag{22}$$

The embedded input vectors are then trivially $x_t^e = E\,x_t = x_t$.

Projections. We choose:

$$\begin{aligned} P_{\text{in}} &= \big(I_V \mid I_V\big)^\top \\ P_{\text{out}} &= \big(0 \mid I_v\big) \end{aligned} \tag{23}$$

The input projection simply duplicates the state:

$$x_t^p = P_{\text{in}}\, x_t^e = \begin{pmatrix} x_t \\ x_t \end{pmatrix} \tag{24}$$

The output projection extracts the "current" half of duplicated current-previous vectors, then projects it onto the value vocabulary space $\mathcal{V}_v$.

Conv1D kernel. We choose

$$W_{\text{conv}} = \big(W_p \mid W_c\big) = \left(\begin{smallmatrix} 1_V & 0_V \\ 0_V & 1_V \end{smallmatrix}\right) \tag{25}$$

with 1_V and 0_V all-ones and all-zeros column vectors in $\mathbb{R}^V$, respectively. This implies:

$$\begin{aligned} \text{diag}\big(W_c\big) &= \big(0 \mid I_V\big) \\ \text{diag}\big(W_p\big) &= \big(I_V \mid 0\big) \end{aligned} \tag{26}$$

This kernel applies identity and shift operations, to form a vector:

$$\hat{x}_t = \text{Conv1D}\big(x_t^p,\, W_{\text{conv}}\big) = \begin{pmatrix} x_{t-1}^e \\ x_t^e \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ x_t \end{pmatrix} \tag{27}$$

This stacked vector is the input to the SSM.

SSM projections and embeddings. For the SSM selection matrices, let us choose:

$$\begin{aligned} S_B &= \big(I_V \mid 0\big) \\ S_C &= \big(0 \mid I_V\big) \end{aligned} \tag{28}$$

That is to say, S_B outputs the previous token,

$$B_t = S_B\ \hat{x}_t = x_{t-1} \tag{29}$$

while S_C outputs the current token:

$$C_t = S_C\, \hat{x}_t = x_t \tag{30}$$

As we will soon see, this allows S_B and S_C to function as key and query extractors, respectively.

Overall mechanism. Let us now look at consecutive context token pairs $(x_{\tau-1}, x_\tau)$, not necessarily a key-value pair, and a current token x_t, not necessarily a (query) key token. Let us also denote

the stacked token-pair vectors as $\xi_t \equiv \begin{pmatrix} x_{t-1} \\ x_t \end{pmatrix}$. Note that in this non-compressive scheme described, the SSM input is exactly $\hat{x}_t = \xi_t$ (see (27)).
With the matrix construction defined above, the SSM recurrent update rule from Alg. 1 now becomes:

$$h_t - h_{t-1} = \hat{x}_t \, B_t^\top = \xi_t \, {x_{t-1}}^\top \tag{31}$$

Hence, assuming $h_0 = 0$, we have

$$\begin{aligned} h_t &= \sum_{\tau=0}^{t} \hat{x}_t \, B_t^\top = \sum_{\tau=1}^{t} \xi_\tau \, {x_{\tau-1}}^\top \\ \hat{y}_t &= h_t \, C_t = \sum_{\tau=1}^{t} \xi_\tau \, {x_{\tau-1}}^\top x_t \end{aligned} \tag{32}$$

The overall model output can be now expressed as:

$$y_t = E^\top P_{\text{out}} \, \hat{y}_t = \sum_{\tau=1}^{t} (P_{\text{out}} \, \xi_\tau) \, {x_{\tau-1}}^\top x_t \tag{33}$$

However, we have constructed P_{out} to extract the current token:

$$P_{\text{out}} \, \xi_\tau = \big(0 \mid I_V\big) \, \xi_\tau = x_\tau \tag{34}$$

We then have:

$$y_t = \sum_{\tau=1}^{t} x_\tau \, {x_{\tau-1}}^\top x_t \tag{35}$$

Hence, if we now have a *query* $x_t = q_m \in \mathcal{V}_k$, then

$$y_t = \sum_{n=1}^{N_f} x_\tau \, {x_{\tau-1}}^\top q_m \tag{36}$$

If we know the query corresponds to a *key* fact in the context $q_m = k_{n^*}$ which is a part of the a fact pair (k_{n^*}, v_{n^*}), then finally, since one-how keys and values are orthonormal:

$$y_t = \sum_{n=1}^{N_f} v_n \, {k_n}^\top k_{n^*} = \sum_{n=1}^{N_f} v_n \, \delta_{n,n^*} = v_{n^*} \tag{37}$$

with $\delta_{i,j}$ the Kronecker delta.
This means the model achieves perfect recall: for any query, it exactly retrieves the correct value from the context. □

C.3 Compressive Recall Circuit

Lemma C.3 (Two consecutive JL transformations). *Let $x_1, \ldots, x_V \in \mathbb{R}^V$ be nonzero vectors and $0 < \varepsilon_v < \varepsilon_k < 1$. If $D \geq \frac{c \log V}{\varepsilon_v^2}$ and $N \geq \frac{c \log V}{\varepsilon_k^2}$ for an absolute constant c, then there exist $E \in \mathbb{R}^{D \times V}$ and $F \in \mathbb{R}^{N \times D}$ such that the JL lemma (Eq. 20) holds **simultaneously** for E w.r.t. ε_v and for $\tilde{E} \equiv FE$ w.r.t. ε_k.*

Proof. Set the auxiliary distortions $\delta_v = \min\big(\varepsilon_v, \frac{\varepsilon_k}{3}\big) \geq \frac{\varepsilon_v}{3}$ and $\delta_k = \frac{\varepsilon_k}{3}$; the assumed D, N then satisfy $D \geq \frac{c' \log V}{\delta_v^2}$, $N \geq \frac{c' \log V}{\delta_k^2}$. Apply the JL lemma (Lem. C.1) to $\{x_i\}_{i=1}^V$ to obtain E with distortion $\delta_v \leq \varepsilon_v$. Fix this realization of E, and apply the JL lemma (Lem. C.1) again to the finite set $\{E \, x_i\}_{i=1}^V \subset \mathbb{R}^D$ to obtain F with distortion δ_k. For any i, j, the triangle inequality gives

$$\begin{aligned} \big|\langle \tilde{E}x_i, \tilde{E}x_j \rangle - \langle x_i, x_j \rangle\big| \;&\leq\; \delta_v \, \|x_i\| \|x_j\| + \delta_k \, \|E \, x_i\| \|E \, x_j\| \;\leq \\ &\leq\; \big(\delta_v + \delta_k + \delta_v \delta_k\big) \|x_i\| \|x_j\| \;\leq \\ &\leq\; \varepsilon_k \, \|x_i\| \|x_j\| \end{aligned} \tag{38}$$

using $\|E \, x_i\| \leq \sqrt{1 + \delta_v} \, \|x_i\|$, where the last inequality holds since $\delta_v, \delta_k \leq \frac{\varepsilon_k}{3}$ and $\varepsilon_k < 1$. □

Theorem C.4 (Efficient compressive recall circuit). *Given an MQAR task with vocabulary size V, N_f facts and context length $L = 4N_f$, a single-layer simplified Mamba with dimensions satisfying*

$ND = O(N_f \log V)$, $\text{expand} = 2$, $D_{\text{conv}} = 2$ *can solve the task* ***with high probability****. (See Lem. 4.1 and Thm. 4.2 for the exact quantitative laws.)*

Proof. We apply quite the same mechanism described in Theorem 3.1 , with modifications to enable compression and decompression.

Projectors. As in the non-compressive circuit, let us define similar *previous and current* projectors, now in $\mathbb{R}^{D\times 2D}$ compressed space:

$$\begin{aligned} M_p &= (I_D \quad 0) \\ M_c &= (0 \quad I_D) \end{aligned} \tag{39}$$

Embeddings. Given a general embedding $E \in \mathbb{R}^{D\times V}$, the embedded input vectors are then $x_t^e = E\, x_t \in \mathbb{R}^D$.

Projections. We choose:

$$\begin{aligned} P_{\text{in}} &= M_p^\top + M_c^\top = \big(I_D \mid I_D\big)^\top \\ P_{\text{out}} &= M_c = \big(0 \mid I_D\big) \end{aligned} \tag{40}$$

The input projection simply duplicates the embedded state:

$$x_t^p = P_{\text{in}}\, x_t^e = \begin{pmatrix} E\, x_t \\ E\, x_t \end{pmatrix} \tag{41}$$

The output projection extracts the "current" half of duplicated current-previous vectors.

Conv1D kernel. We choose

$$W_{\text{conv}} = \big(W_p \mid W_c\big) = \left(\begin{smallmatrix} 1_D & \\ 0_D & \end{smallmatrix} \;\middle|\; \begin{smallmatrix} & 0_D \\ & 1_D \end{smallmatrix}\right) \tag{42}$$

such that

$$\begin{aligned} \text{diag}\big(W_c\big) &= \big(0 \mid I_D\big) \\ \text{diag}\big(W_p\big) &= \big(I_D \mid 0\big) \end{aligned} \tag{43}$$

This kernel applies identity and shift operations, to form

$$\hat{x}_t = \text{Conv1D}\big(x_t^p,\, W_{\text{conv}}\big) = \begin{pmatrix} x_{t-1}^e \\ x_t^e \end{pmatrix} = \begin{pmatrix} E\, x_{t-1} \\ E\, x_t \end{pmatrix} \tag{44}$$

SSM projections and embeddings. Let $F \in \mathbb{R}^{N\times D}$ be an embedding matrix. We choose:

$$\begin{aligned} S_B &= F\, M_p = \big(F \mid 0\big) \\ S_C &= F\, M_c = \big(0 \mid F\big) \end{aligned} \tag{45}$$

That is to say, S_B outputs a compressed previous token,

$$B_t = S_B\, \hat{x}_t = F\, M_p\, \hat{x}_t = F\, x_{t-1}^e = F\, E\, x_{t-1} = \tilde{E} x_{t-1} \in \mathbb{R}^N \tag{46}$$

while S_C outputs a compressed current token,

$$C_t = S_C\, \hat{x} = F\, M_c\, \hat{x}_t = F\, x_t^e = F\, E\, x_t = \tilde{E} x_t \in \mathbb{R}^N \tag{47}$$

where we have introduced $\tilde{E} \equiv FE$.

Overall mechanism. Given the context consecutive key-value token pairs $(x_{\tau-1}, x_\tau) \equiv (k_\tau, v_\tau)$, and a query token $x_t \equiv q_t$, we notice the matrix setup defined above describes a simple compression-decompression scheme.

The SSM hidden state and output from 32 are now expressed as:

$$\begin{aligned} h_t &= \sum_{\tau=0}^{t} \hat{x}_t\, B_t^\top = \sum_{(\tau-1)\in\{\tau_k\}}^{t} \begin{pmatrix} E\, x_{\tau-1} \\ E\, x_\tau \end{pmatrix} (\tilde{E}\, x_{\tau-1})^\top \\ \hat{y}_t &= h_t\, C_t = \sum_{(\tau-1)\in\{\tau_k\}}^{t} \begin{pmatrix} E\, x_{\tau-1} \\ E\, x_\tau \end{pmatrix} (\tilde{E}\, x_{\tau-1})^\top (\tilde{E}\, x_t) \end{aligned} \tag{48}$$

If we recall that $P_{\text{out}} = M_c = \big(0 \mid I_D\big)$, the overall layer output is therefore:

$$y_t = E^\top P_{\text{out}}\, \hat{y}_t \sum_{(\tau-1)\in\{\tau_k\}}^{t} E^\top (E\, x_{\tau-1})\, (\tilde{E}\, x_{\tau-1})^\top (\tilde{E}\, x_t) \tag{49}$$

This expression can be seen as simple bilinear function of the original one-hot (non-compressed) tokens:

$$y_t = \sum_{\tau=0}^{t} (E^\top E)\, x_\tau\, {x_{\tau-1}}^\top (\tilde{E}^\top \tilde{E})\; x_t \tag{50}$$

If we further look at the Gram matrices, defined as

$$\begin{aligned} G_E &\equiv E^\top E \in \mathbb{R}^{V\times V} \\ G_{\tilde{E}} &\equiv \tilde{E}^\top \tilde{E} \in \mathbb{R}^{V\times V} \end{aligned} \tag{51}$$

(which, in our case, corresponds to applying successive compression and decompression), the output vector can now be expressed as:

$$y_t = \sum_{\tau=0}^{t} G_E\, x_\tau\, {x_{\tau-1}}^\top\; G_{\tilde{E}}\; x_t \tag{52}$$

This would have reduced to a perfect retrieval, as described in the perfect case, only if we had perfect embeddings, satisfying $G_E = I_V$ and $G_{\tilde{E}} = I_V$, which is, of course, impossible for $N < D < V$. However, for proper, large enough dimensions, we can approximately achieve perfect recall, as we will now state.

Applying JL Lemma. Using *Johnson–Lindenstrauss lemma* [23] (Lem. C.1) twice, for arbitrarily small $0 < \varepsilon_k,\, \varepsilon_v < 1$, if model dimensions are large enough, $D > \frac{c\,\log V}{\varepsilon_v^2}$, $N > \frac{c\,\log V}{\varepsilon_k^2}$ for some constant c, we can construct embedding matrices E, $\tilde{E}$ for which

$$\big|\langle Ex_i, Ex_j\rangle - \langle x_i, x_j\rangle\big| \;=\; \big|\, x_i^\top (E^\top E - I)\, x_j \,\big| \le\; \varepsilon_v\, \|x_i\|\, \|x_j\| \tag{53}$$

and similarly

$$\big|\langle \tilde{E}x_i, \tilde{E}x_j\rangle - \langle x_i, x_j\rangle\big| \;=\; \big|\, x_i^\top (\tilde{E}^\top \tilde{E} - I)\, x_j \,\big| \le\; \varepsilon_k\, \|x_i\|\, \|x_j\| \tag{54}$$

Note that the two applications are not independent, since $\tilde{E}$ must factor as $\tilde{E} = FE$. Using Lem. C.3, there exist matrices E and $\tilde{E} = FE$ satisfying both JL properties simultaneously.
Since our input vectors are all one-hot (satisfying $\|x_i\| = 1$):

$$\big|\langle Ex_i, Ex_j\rangle - \langle x_i, x_j\rangle\big| \;\le\; \varepsilon_v \tag{55}$$

For such embeddings, we can further bound the inner products for one-hot vectors, by:

$$x_i^\top G_E\, x_j = (E\, x_i)^\top (E\, x_j) \in \begin{cases} (1-\varepsilon_v,\, 1+\varepsilon_v) & \text{if } \; i = j \\ (-\varepsilon_v,\, \varepsilon_v) & \text{if } \; i \neq j \end{cases} \tag{56}$$

and similarly:

$$x_i^\top G_{\tilde{E}}\, x_j = (\tilde{E}\, x_i)^\top (\tilde{E}\, x_j) \in \begin{cases} (1-\varepsilon_k,\, 1+\varepsilon_k) & \text{if } \; i = j \\ (-\varepsilon_k,\, \varepsilon_k) & \text{if } \; i \neq j \end{cases} \tag{57}$$

Even without an exact analysis (which is further developed later in Lemma 4.1 and Theorem 4.2), we claim that for small enough ε_v, ε_k, almost-perfect recall is achievable. To see why, let us examine the output vector y_t.

Entry-wise analysis. As before, let us assume the context contains a single correct fact pair with a key m and a value i, both one-hot encoded by token vectors (k_m, v_i).
The vector y_t has V entries in total, divided into:

- 1 *correct* value entry i, corresponds to the correct fact value v_i.
- $N_f - 1$ *wrong* value entries, matching $N_f - 1$ wrong fact values.
- $V_v - N_f$ *empty* value entries, matching values that do not appear in the context.
- V_k *key* entries.

For any vector entry p, using the one-hot basis vector e_p, we can express the entry scalar value using the following bilinear forms:

$$y_t^p = e_p^\top y_t = \sum_{\tau=0}^{t} \left(e_p^\top G_E\, x_\tau\right) \left({x_{\tau-1}}^\top G_{\tilde{E}}\, k_m\right) \tag{58}$$

However, since $E, \tilde{E}$ are JL embeddings, we can state (not quantitatively for now; see Lemma 4.1 and Theorem 4.2):

$$e_p^\top G_E\, x_\tau \;\approx\; \begin{cases} 1 & x_\tau = e_p \\ 0 & x_\tau \neq e_p \end{cases}, \qquad x_{\tau-1}^\top G_{\tilde{E}}\, k_m \;\approx\; \begin{cases} 1 & x_{\tau-1} = k_m \\ 0 & x_{\tau-1} \neq k_m \end{cases} \tag{59}$$

Note that this approximation becomes more exact as the model dimensions grow. Since in MQAR the tokens do not repeat, we have both terms ≈ 1 only if $(x_{\tau-1},\, x_\tau) = (e_p,\, k_m)$. However, this can only be found in the single fact where k_m appears; namely, $(x_{\tau-1},\, x_\tau) = (v_i,\, k_m)$. This occurs only for this single specific entry i, while for all other entries, we have at least one of the terms ≈ 0. Therefore:

$$y_t^p \;\approx\; \begin{cases} 1 & p = i \\ 0 & p \neq i \end{cases} \tag{60}$$

or, more simply, $y_t \approx v_i$.
That is to say, for small enough ε_v, ε_k, or equivalently for large enough D, N, the simplified linear Mamba model with weights as constructed above can achieve almost-perfect recall.
Quantitatively, the matching term contributes $v_i + O(\varepsilon_v + \varepsilon_k)$, while each of the $N_f - 1$ non-matching terms leaves an $O(\varepsilon_v \varepsilon_k)$ residue per entry. For large N_f, these $O(\varepsilon_v \varepsilon_k)$ terms dominate; in the worst case, the residues add up to $O(N_f\, \varepsilon_v \varepsilon_k)$, while in the mean case they cancel down to $O\big(\sqrt{N_f}\, \varepsilon_v \varepsilon_k\big)$. Thus, to solve the task, $\sqrt{ND} = O(N_f \log V)$ is required in the worst case, while $ND = O(N_f \log V)$ suffices with high probability. (See Lem. 4.1 for the exact worst-case conditions, and Thm. 4.2 for the exact high-probability law.)

□

D Mechanistic Interpretability of the Recall Circuit: Supplementary Material

D.1 Invariant Operators

G_{kq} is a bilinear operator, operates on token pairs: $\xi_t = \binom{x_{t-1}}{x_t}$ and $\xi_\tau = \binom{x_{\tau-1}}{x_\tau}$. Therefore, it consists of 4 blocks, corresponding to the 4 combinations of x_t, x_{t-1} parts and $x_\tau, x_{\tau-1}$ parts.
As predicted by theory and observed in Fig. 15, $\xi_\tau^\top\, G_{kq}\, \xi_t$ is nonzero only for the $(\tau - 1, t)$ block. This corresponds to the mechanism of matching between a query $q_t = x_t$ and its corresponding key $k_t = x_{\tau-1}$. Note that x_τ and x_{t-1} are completely ignored by G_{kq}, which makes sense: they are unrelated to the query-key matching procedure.
Similarly, G_{vv} is thought as a matching between token pairs $\xi_\tau = \binom{x_{\tau-1}}{x_\tau}$ and output tokens y_t. Hence, it consists of 2 blocks, corresponding the $x_\tau, x_{\tau-1}$ parts and the y_t part.
With a fit to our theory, Fig. 15, shows that $y_t^\top\, G_{vv}\, \xi_t$ is nonzero only for the (t, τ) block. This corresponds to the mechanism of retrieving $y_t = x_\tau$, which is the correct value given a query-key match $x_t = x_{\tau-1}$. Also note that G_{vv} ignores $x_{\tau-1}$ (the key token): only the value x_τ matters.

D.2 SSM Hidden State and Key-Value Tables

We shortly present and discuss the definitions for H_t, H'_t and H''_t.

Table 4: Dimensions for hidden state and key-value tables

Symbol	Meaning	Dimension	Notes
h_t^{ideal}	Hidden state, non-compressive circuit	$\mathbb{R}^{2V \times V}$	As constructed in Thm. 3.1
h_t^{noisy}	Hidden state, compressive circuit	$\mathbb{R}^{2D \times N}$	As constructed in Thm. 3.2
H_t	Projected hidden state, non-compressive circuit	$\mathbb{R}^{V \times V}$	$= (0 \mid I_V)\, h_t^{\text{ideal}}$
H'_t	Projected hidden state, compressive circuit	$\mathbb{R}^{D \times N}$	$= (0 \mid I_D)\, h_t^{\text{noisy}}$
H''_t	Decompressed hidden state, compressive circuit	$\mathbb{R}^{V \times V}$	$= E^\top H'_t\, \tilde{E}$

For clarity, let us denote the variables from Thm. 3.1 as *ideal*, and the variables from Thm. 3.2 as *noisy*.
For the two circuits, ideal and compressive, we now define:

$$H_t \equiv P_{\text{out}}\, h_t^{\text{ideal}} = \sum_{n=1}^{N_f} v_n k_n^\top \tag{61}$$

$$H'_t \equiv P_{\text{out}}\, h_t^{\text{noisy}} = \sum_{n=1}^{N_f} v'_n {k'_n}^\top \tag{62}$$

$$H''_t \equiv E^\top H'_t \tilde{E} = \sum_{n=1}^{N_f} v''_n {k''_n}^\top \tag{63}$$

where the *compressed* tokens are $v'_n = E\, v_n$, $k'_n = \tilde{E}\, k_n$, $q'_t = \tilde{E}\, q_t$, and the *decompressed* tokens are $v''_n = E^\top E\, v_n \approx v_n$ and $k''_n = \tilde{E}^\top \tilde{E}\, k_n \approx k_n$. (Note that $E \in \mathbb{R}^{D\times V}$ and $\tilde{E} \in \mathbb{R}^{N\times V}$ are JL matrices, thus achieving approximately perfect compression and decompression, given sufficiently large D, N.)
The model output for each of the circuits can be now written as:

$$y_t^{\text{ideal}} = H_t\, q_t = \sum_{n=1}^{N_f} v_n k_n^\top q_t \tag{64}$$

$$y_t^{\text{noisy}} = E^\top H'_t\, q'_t = E^\top \sum_{n=1}^{N_f} v'_n {k'_n}^\top q'_t \tag{65}$$

$$y_t^{\text{noisy}} = H''_t\, q_t = \sum_{n=1}^{N_f} v''_n {k''_n}^\top q_t \tag{66}$$

where the latter two equations are equivalent.
Throughout the manuscript, the *projected hidden states* H_t, H'_t, H''_t are sometimes referred to as *hidden states* as well. The reason is that for MQAR, the *informative* parts of h_t^{ideal} and h_t^{noisy} (which contain all key-value information) entirely lie in the H_t, H'_t halves, respectively.

D.3 Hidden State as a Hash Table

Intuitively, for a perfect recall, the SSM hidden state must encode the full context *information*. This fact-pairs information can be thought of as a simple key-value array, where each key entry contains 1 in the corresponding fact value v_τ index, if exists, else 0 anywhere. The ideal, non-realistic, circuit of Thm. 3.1 suggests such storage via an outer product key-value memory (see definitions and dimensions in App. D.2):

$$H_t \equiv P_{\text{out}}\, h_t = \sum_{n=1}^{N_f} v_n k_n^\top, \quad y_t = H_t\, q_t = \sum_{n=1}^{N_f} v_n k_n^\top q_t \tag{67}$$

This is a $V \times V$ table, quite similar to the array presented above, though with values represented as one-hot vectors. A visualization can be found in Fig. 3 (left). H_t is fairly interpretable, as entries are simply 1 where facts exist and 0 elsewhere. Can we observe such interpretable patterns within trained models? Let us consider the compressive SSM update from Eq. 8. It can be rewritten as

$$H'_t = \sum_{n=1}^{N_f} v'_n {k'_n}^\top, \quad y_t = E^\top \sum_{n=1}^{N_f} v'_n {k'_n}^\top q'_t\,, \tag{68}$$

with compressed tokens $v'_n = E\, v_n$, $k'_n = \tilde{E}\, k_n$, $q'_t = \tilde{E}\, q_t$.
H'_t can be viewed as a ***hash table***, for several reasons: (i) both mechanisms rely on a hash function. In our case, the model uses JL linear projections, which is by definition a similarity-preserving hash function; (ii) in both cases, the internal representation is not directly interpretable (Fig. 3, middle), while the final output is fairly interpretable (Fig. 3, right); (iii) efficiency is achieved by design through approximation, where the error is bounded with high probability due to the

properties of the hash function; (iv) the interface closely follows that of a hash table: First, keys and queries are first hashed via $\mathrm{hash}(x) = \tilde{E}x$. Next, the model *stores* a fact into the table via $H_t = H_{t-1} + \mathrm{encode}(v_n)\,\mathrm{hash}(k_n)^\top$. Finally, when handling a query, the model *matches* $\mathrm{hash}(q_t)$ with $\mathrm{hash}(k_i)$ for all entries i, using inner products. If a key k_n exists such that $\mathrm{hash}(k_n) \approx \mathrm{hash}(q_t)$, the corresponding value is approximately *retrieved*.

D.4 Inspecting the Hash Table

The starting point of our analysis is the *non-rotated* recall circuit, as constructed in Theorem 3.2, where P_{in}, P_{out}, and W_{conv} are carefully designed with identity and zero matrices and vectors. In this setup, S_B and S_C must have a structure $S_B = (F \mid 0)$ and $S_C = (0 \mid F)$ in order to solve MQAR. If we fix P_{in}, P_{out}, and W_{conv} as in the construction, then indeed F can be extracted from learned model weights, allows to compute $\tilde{E}$, then to compute Equation 14 and $H'' = E^\top H' \tilde{E}$. However, in the general *rotated* case, where P_{in}, P_{out}, and W_{conv} are not constrained, the operator F does not exist as-is in any of the model weights: S_B and S_C do not have the simple structure described in Theorem 3.2, since the key and query compression operations F are distributed across a mixture of linear operators S_B, S_C, P_{in}, and W_{conv}. Therefore, we must use the net operations $\Pi_{k,\mathrm{in}}$ and $\Pi_{q,\mathrm{in}}$ (see Equation 9), which correctly combine all these linear transformations. Decompression via $k''_n = \tilde{E}^\top \tilde{E}\, k_n$ is a special case, which only applies for the *non-rotated* circuit; its general *rotation-invariant* form is $k''_n = G_{kq}^\top\, k_n$. Similarly, $H''_t = E^\top H'_t \tilde{E}$ is a special *non-rotated* case; its general invariant form is $H''_t = \Pi_{v,\mathrm{out}}\, H'_t\, \Pi_{q,\mathrm{in}}$, which is derived by replacing $E^\top$ and $\tilde{E}$ with their general equivalent operators from Equation 9.

D.5 Noisy Attention Maps

In addition to the hash-like recurrent view of the recall mechanism, we can alternatively view it through the lens of implicit attention maps. Following Ali et al. [51], Dao and Gu [48], for our simplified model, the attention matrix of the ideal circuit becomes $\alpha_{\tau t} = B_\tau^\top\, C_t = x_{\tau-1}^\top x_t$, such that $y_t = \sum_{\tau=0}^{t} x_\tau\, \alpha_{\tau t}$, equivalently to Eq. 6. Simply put, this formulation implies that given a query $x_t = q_t$, the model attends to previous tokens that satisfy $x_{\tau-1} = q_t$. For the realistic, compressive circuit, we have, based on Eq. 8, $\alpha'_{\tau t} = B_\tau^\top\, C_t = (\tilde{E}\, x_{\tau-1})^\top (\tilde{E}\, x_t) = x_{\tau-1}^\top \tilde{E}^\top \tilde{E}\, x_t \approx x_{\tau-1}^\top x_t$, hence $\alpha'_{\tau t} \approx \alpha_{\tau t}$. (In equivalent circuits, the map is more complicated, $\alpha'_{\tau t} = \xi_\tau^\top G_{kq}\, \xi_t$, yet still $\approx \alpha_{\tau t}$). Indeed, computing these maps for both ideal and trained compressive models, we verify in Fig. 13 that the learned map $\alpha'_{\tau t}$ functions as an approximation to the ideal one $\alpha_{\tau t}$, revealing a circuit-level resemblance between regimes.

E Designed Model Baseline

In this appendix, we describe an example designed model with a specific weight construction. This model serves as a sub-optimal baseline solution for associative recall tasks, which is simpler for mathematical analysis. Later, in App. F.2, we theoretically analyze the model performance on (single-query) AR and MQAR tasks. The designed model and the theoretical results are used in Section 5 and Figure 6, compared to the trained models.

E.1 Designed Model Weights

Design of E and F. Following Theorem 3.2, given general matrices $E \in \mathbb{R}^{D\times V}$ and $F \in \mathbb{R}^{N\times D}$, we choose:

$$P_{\mathrm{in}} = \begin{pmatrix} I_D \\ I_D \end{pmatrix}, \quad P_{\mathrm{out}} = \begin{pmatrix} 0 \mid I_D \end{pmatrix}, \quad W_{\mathrm{conv}} = \begin{pmatrix} 1_D & | & 0_D \\ 0_D & | & 1_D \end{pmatrix}, \quad S_B = \begin{pmatrix} F \mid 0 \end{pmatrix}, \quad S_C = \begin{pmatrix} 0 \mid F \end{pmatrix} \tag{69}$$

We would now like to choose specific E and F matrices.
The embedding matrix E is constructed as follows:

- Sample a random matrix $G \in \mathbb{R}^{D\times V}$ with i.i.d. standard normal entries.
- Rescale each column G_i by its norm $\|G_i\|$, to form a matrix $\hat{G} \in \mathbb{R}^{D\times V}$ with columns of unit norm, $\|\hat{G}_i\| = 1$.
- Set the embedding matrix $E = \hat{G} \in \mathbb{R}^{D\times V}$.

The matrix F is constructed as follows:

- Sample a random matrix $P \in \mathbb{R}^{D\times D}$ with i.i.d. standard normal entries.
- Extract an orthonormal basis $Q \in \mathbb{R}^{D\times D}$ via a QR decomposition of P.

- Select its first N columns, $U = Q[:N] \in \mathbb{R}^{N\times D}$.
- Scale the matrix by a scalar factor $a \in \mathbb{R}$: $F = a\,U \in \mathbb{R}^{N\times D}$.
- We later choose $a = \sqrt{\frac{D}{N}}$.

As in Theorem 3.2, we define $\tilde{E} \equiv FE \in \mathbb{R}^{N\times V}$. We further define the Gram matrices:

$$\begin{aligned} G_E &\equiv E^\top E \in \mathbb{R}^{V\times V} \\ G_{\tilde{E}} &\equiv \tilde{E}^\top \tilde{E} \in \mathbb{R}^{V\times V} \end{aligned} \tag{70}$$

We are interested in the statistics of G_E and $G_{\tilde{E}}$, which we will later use to evaluate the accuracy of the designed model on AR and MQAR. Note that for large V, the entries of G_E and $G_{\tilde{E}}$ obtained from a single realization of E and F exhibit empirical statistics (e.g. mean and variance) that closely match the corresponding ensemble statistics, averaged over different realizations of E and F.

Statistics of $\mathbf{G_E}$. Each column e_i of the embedding matrix $E \in \mathbb{R}^{D\times V}$ is obtained by normalizing an i.i.d. Gaussian vector and is therefore uniformly distributed on the unit sphere S^{D-1}. As a consequence, the Gram matrix $G_E = E^\top E$ has diagonal entries $(G_E)_{ii} = \|e_i\|^2 = 1$ deterministically. For $i \neq j$, the off-diagonal entries $(G_E)_{ij} = \langle e_i, e_j\rangle$ are independent samples from the inner-product distribution of two random unit vectors in $\mathbb{R}^D$, which is known to have a probability density:

$$f_D(t) = \begin{cases} \frac{\Gamma(\frac{D}{2})}{\sqrt{\pi}\,\Gamma(\frac{D-1}{2})}(1-t^2)^{\frac{D-3}{2}} & t \in [-1,1] \\ 0 & \text{otherwise} \end{cases} \tag{71}$$

Alternatively, $(G_E)_{ij} = 2Z - 1$ where $Z \sim \mathrm{Beta}(\frac{D-1}{2}, \frac{D-1}{2})$. This distribution is centered around 0, with:

$$\mathbb{E}(G_E)_{ij} = 0, \qquad \mathrm{Var}(G_E)_{ij} = \frac{1}{D} \tag{72}$$

a variance of $\frac{1}{D}$. For large D, the distribution of $(G_E)_{ij}$ is well approximated by $\mathcal{N}(0, \frac{1}{D})$.

Statistics of $\mathbf{G_{\tilde{E}}}$. The projected embedding matrix is defined as $\tilde{E} = FE = aUE$, where the rows of $U \in \mathbb{R}^{N\times D}$ form an orthonormal basis of a random N-dimensional subspace. Writing $\Pi = U^\top U$ for the associated rank-N orthogonal projector, the Gram matrix of $\tilde{E}$ can be written as $G_{\tilde{E}} = a^2 E^\top \Pi E$. Its diagonal entries satisfy $(G_{\tilde{E}})_{ii} = a^2\,Z$, where $Z = e_i^\top \Pi e_i \sim \mathrm{Beta}(\frac{N}{2}, \frac{D-N}{2})$ distribution. Consequently,

$$\mathbb{E}(G_{\tilde{E}})_{ii} = a^2\frac{N}{D}, \qquad \mathrm{Var}(G_{\tilde{E}})_{ii} = a^4\frac{\frac{N}{D}(1-\frac{N}{D})}{D+2} \approx a^4\frac{N}{D}\left(1-\frac{N}{D}\right)\frac{1}{D} \tag{73}$$

For $i \neq j$, the off-diagonal entries are given by $(G_{\tilde{E}})_{ij} = a^2 e_i^\top \Pi e_j$, which are centered around 0 with variance $a^4\frac{N}{D^2}$, and are well approximated by $\mathcal{N}(0, a^4\frac{N}{D^2})$ in the high-dimensional regime. If we now choose $a = \sqrt{\frac{D}{N}}$, we have:

$$\mathbb{E}(G_{\tilde{E}})_{ii} = 1, \qquad \mathrm{Var}(G_{\tilde{E}})_{ii} \approx \frac{1}{N} - \frac{1}{D} \tag{74}$$

$$\mathbb{E}(G_{\tilde{E}})_{ij} = 0, \qquad \mathrm{Var}(G_{\tilde{E}})_{ij} = \frac{1}{N} \tag{75}$$

F Mamba Recall Scaling Laws: Proofs

F.1 JL-Based Scaling Laws

Lemma F.1 (JL-based recall scaling laws). *Given a vocabulary size V, a single-layer simplified Mamba with dimensions $D = \frac{4\log V}{\varepsilon_v^2}$, $N = \frac{4\log V}{\varepsilon_k^2}$, $\mathrm{expand} = 2$, $D_{\mathrm{conv}} = 2$ can perfectly solve an AR task if model dimensions satisfy $0 < \varepsilon_v < 1$, $0 < \varepsilon_k < 1$ and*

$$\varepsilon_v + \varepsilon_k + 2N_f\,\varepsilon_v\,\varepsilon_k < \frac{1}{2}. \tag{76}$$

The model can perfectly solve an MQAR task if model dimensions further satisfy

$$\varepsilon_v + \varepsilon_k + L\,\varepsilon_v\,\varepsilon_k < \frac{1}{2}. \tag{77}$$

Proof. We use the lower and upper bounds derived from the *JL lemma* (Lem. C.1) to separate the correct value output entry from all other entries.

JL bounds. Assume the model weights are constructed as described in Theorem 3.2. Given a query $x_t = q_t = k_m$, the model output vector y_t has value entries given by: (since v_p are one-hot vectors)

$$y_t^p = v_p^\top y_t = \sum_{\tau=0}^{t} \left(v_p^\top G_E\, x_\tau\right) \left({x_{\tau-1}}^\top G_{\tilde{E}}\; k_m\right) \tag{78}$$

Since $E, \tilde{E}$ are both JL embeddings, we have:

$$v_p^\top G_E\, x_\tau \in \begin{cases} (1-\varepsilon_v,\, 1+\varepsilon_v) & x_\tau = v_p \\ (-\varepsilon_v,\, \varepsilon_v) & x_\tau \neq v_p \end{cases} \tag{79}$$

$$x_{\tau-1}^\top\, G_{\tilde{E}}\, k_m \;\in \begin{cases} (1-\varepsilon_k,\, 1+\varepsilon_k) & x_{\tau-1} = k_m \\ (-\varepsilon_k,\, \varepsilon_k) & x_{\tau-1} \neq k_m \end{cases} \tag{80}$$

For simplicity, let us first examine a single-query retrieval, where a query q_t has a sequence index t. We thus assume the context contains $t-1$ tokens previous to the query.

Entry-wise analysis. We start with the output vector y_t decomposition. For each entry p, we analyze the contribution of each token pair in the context to the sum in Eq. 78. For brevity, we denote $g_\tau = v_p^\top G_E\, x_\tau$ and $\tilde{g}_\tau = x_{\tau-1}^\top\, G_{\tilde{E}}\, k_m$. Given a value entry p, for each token pair $(x_{\tau-1}, x_\tau)$, we say we have a *value match* if $x_\tau = v_p$ (thus $g_\tau \approx 1$) and a *key match* if $x_{\tau-1} = k_m$ (such that $\tilde{g}_\tau \approx 1$). As discussed in F.3, we assume key and value embeddings are orthogonal.
For the ***correct*** entry i, the total t context token pairs are divided into:

- t token pairs:
  - 1 *correct* fact pair: for this pair, both key and value match, $(x_{\tau-1}, x_\tau) = (k_m, v_i)$. Hence, applying JL lower bound, $g_\tau \tilde{g}_\tau > (1-\varepsilon_v)(1-\varepsilon_k)$.
  - $t-1$ other token pairs: since key and values are unique (and k_m, v_i have already appeared in the correct fact), for each of these pairs we have $x_{\tau-1} \neq k_m$ and $x_\tau \neq v_i$. Therefore, using JL bound, $g_\tau \tilde{g}_\tau > -\varepsilon_v \varepsilon_k$.

Applying JL lemma lower bound, the contributions add up to:

$$y_t^i \;>\; (1-\varepsilon_v)(1-\varepsilon_k) \;-(t-1)\varepsilon_v\varepsilon_k \approx\; 1-\varepsilon_v-\varepsilon_k-t\,\varepsilon_v\varepsilon_k \tag{81}$$

For each of the $N_f - 1$ ***wrong*** entries j, we have:

- t token pairs:
  - 1 *correct* fact: for this fact, we have a key match and a value mismatch, since $x_{\tau-1} = k_m$ and $x_\tau = v_i \neq v_j$. Using JL upper bound, this fact contributes $g_\tau \tilde{g}_\tau < \varepsilon_v(1+\varepsilon_k)$.
  - 1 *wrong* fact with $x_\tau = v_j$: we have a key mismatch, since $x_{\tau-1} = k_n \neq k_m$, and a value match, since $x_\tau = v_j$ match entry index j. Hence, applying JL bound, this fact contributes $g_\tau \tilde{g}_\tau < (1+\varepsilon_v)\varepsilon_k$.
  - $t-2$ *mismatching* token pairs with $x_\tau \neq v_j$: as before, we have key mismatch, and now also a value mismatch, since $x_\tau \neq v_j$. These facts contribute $g_\tau \tilde{g}_\tau < \varepsilon_v\varepsilon_k$.

Applying JL lemma upper bound, we have:

$$y_t^j \;<\; \varepsilon_v(1+\varepsilon_k) + (1+\varepsilon_v)\varepsilon_k \;+(t-2)\varepsilon_v\varepsilon_k = \varepsilon_v + \varepsilon_k \;+ t\,\varepsilon_v\varepsilon_k \tag{82}$$

For each of the rest $V - N_f$ ***empty*** entries l, corresponding to tokens y_l which do no appear in any context fact, we simply have:

- t token pairs:
  - 1 *correct* fact: as above, a key match and a value mismatch.
  - $t-1$ *mismatching* pairs: a key mismatch and a value mismatch.

Applying JL lemma upper bound, we have:

$$y_t^l \;<\; \varepsilon_v(1+\varepsilon_k) + (t-1)\,\varepsilon_v\varepsilon_k = \varepsilon_v + t\,\varepsilon_v\varepsilon_k \tag{83}$$

Perfect recall condition. For perfect retrieval, we want the correct-value entry i to be maximal:

$$y_t^i = \max_p \; y_t^p \tag{84}$$

Using our JL-based bounds, $y_t^i > y_t^{p\neq i}$ is achieved when

$$1 - \varepsilon_v - \varepsilon_k - t\varepsilon_v\varepsilon_k > \varepsilon_v + \varepsilon_k \; + t\,\varepsilon_v\varepsilon_k \tag{85}$$

or, simpler, if

$$\varepsilon_v + \varepsilon_k + t\varepsilon_k\varepsilon_v < \frac{1}{2} \tag{86}$$

From JL bound to a scaling law. We can now think of ε_v, ε_k as of new, "natural" dimension variables, replacing the original dimensions as in a coordinate transformation:

$$\begin{aligned} \varepsilon_v &\equiv \sqrt{\frac{c\,\log V}{D}} \\ \varepsilon_k &\equiv \sqrt{\frac{c\,\log V}{N}} \end{aligned} \tag{87}$$

JL lemma generally requires $\varepsilon_v, \varepsilon_k < 1$, or equivalently, $D, N > c\,\log V$. To these requirements we add the specific MQAR condition given in equation 86, which can now be seen as a *scaling law*:

$$\sqrt{\frac{c\,\log V}{D}} + \sqrt{\frac{c\,\log V}{N}} + t\frac{c\,\log V}{\sqrt{ND}} < \frac{1}{2} \tag{88}$$

Choosing $t = 2N_f$ yields the condition $\varepsilon_v < 1$, $\varepsilon_k < 1$ and $\varepsilon_v + \varepsilon_k + 2N_f\,\varepsilon_v\,\varepsilon_k < \frac{1}{2}$ to guarantee perfect recall on AR. For MQAR, if dimensions satisfy $\varepsilon_v + \varepsilon_k + L\,\varepsilon_v\,\varepsilon_k < \frac{1}{2}$, we are guaranteed to perfectly handle all queries, at any $2N_f < t \leq L$.

□

F.2 Probabilistic Scaling Laws: Designed Model

In the following Lemma, we analyze the predicted success probability of the designed model (see App. E.1) solving AR and MQAR tasks.

Theorem F.2 (Designed model scaling laws). *A single-layer simplified Mamba with weights as designed in E.1 solves an AR task with a probability approximated by*

$$p_{\text{AR}} \approx \Phi\left(\sqrt{\frac{ND}{2N_f}} - \sqrt{2\log V}\right), \tag{89}$$

and an MQAR task of $L = 4N_f$ with a probability

$$p_{\text{MQAR}} \approx \frac{1}{L-2N_f}\sum_{t=2N_f}^{L}\Phi\left(\sqrt{\frac{ND}{t}} - \sqrt{2\log V}\right), \tag{90}$$

in the limits of $N_f \gg N, D$ and $V \gg 1$, where Φ is the standard normal CDF.

Proof. Let us denote the input sequence by $x = x_1, x_2, \ldots, x_t = k_1, v_1, k_2, v_2 \ldots, k_{N_f}, v_{N_f}, q$, where the last token is the query $x_t = q = k_m \in \mathcal{V}_k$. Given the model weights as designed in E.1, the model output y_t is:

$$y_t = E^\top \sum_{\tau}^{t} (Ex_\tau)(\tilde{E}x_{\tau-1})^\top(\tilde{E}q) \tag{91}$$

Thus, the i^{th} coordinate of y_t can be written using the one-hot vector $v_i = e_i$:

$$
\begin{aligned}
y_t^i &= v_i^\top y_t = \\
&= v_i^\top E^\top \sum_\tau^t (Ex_\tau)(\tilde{E}x_{\tau-1})^\top (\tilde{E}q) = \\
&= \sum_\tau^t (Ev_i)^\top (Ex_\tau)(\tilde{E}x_{\tau-1})^\top (\tilde{E}k_m) = \\
&= \sum_\tau^t (v_i^\top G_E\, x_\tau)(x_{\tau-1}^\top G_{\tilde{E}}\, k_m) = \\
&= \sum_{n=1}^{N_f} (v_i^\top G_E\, v_n)(k_n^\top G_{\tilde{E}}\, k_m) + \sum_{n=1}^{N_f} (v_i^\top G_E\, k_{n+1})(v_n^\top G_{\tilde{E}}\, k_m)
\end{aligned}
\tag{92}
$$

where in the last equality we split the total $2N_f$ pairs $(x_{\tau-1}, x_\tau)$ into N_f key-value fact pairs (k_n, v_n) and N_f value-key "off-fact" pairs (v_n, k_{n+1}).
Defining random variables S_{ii} and Z_{ij} ("signal" and "zero" terms) for the diagonal and off-diagonal entries of G_E, we have:

$$
x_i^\top G_E\, x_j \;=\; \begin{cases} S_{ii} & i = j \\ Z_{ij} & i \neq j \end{cases}
\tag{93}
$$

Similarly, for $G_{\tilde{E}}$, we define $\tilde{S}_{ii}$ and $\tilde{Z}_{ij}$:

$$
x_i^\top G_{\tilde{E}}\, x_j \;=\; \begin{cases} \tilde{S}_{ii} & i = j \\ \tilde{Z}_{ij} & i \neq j \end{cases}
\tag{94}
$$

Entry-wise analysis. Now let us assume the retrieved fact is (k_m, v_i). We use i for the *correct* value entry, j for *wrong* value entries (associated with "wrong facts" (k_n, v_j) in the context) and l for *empty* entries (all other entries; also see Equations 81, 82, 83). Then, the output entries become:

$$
y_t^i = S_{ii}\, \tilde{S}_{mm} + \sum_p^{2N_f-1} Z_p \tilde{Z}_p
\tag{95}
$$

$$
y_t^j = S_{jj}\, \tilde{Z}_{nm} + Z_{ji}\, \tilde{S}_{mm} + \sum_p^{2N_f-2} Z_p \tilde{Z}_p
\tag{96}
$$

$$
y_t^l = Z_{li}\, \tilde{S}_{mm} + \sum_p^{2N_f-1} Z_p \tilde{Z}_p
\tag{97}
$$

However, for the specific weights of the designed model, $S_{ii}^d = S_{jj}^d = 1$ deterministically, and $\tilde{S}_{mm}^d$ is distributed around 1, such that overall, the mean and variance of the *correct*, *wrong* and *empty* coordinates are, respectively:

$$
\begin{aligned}
\mu_c &= \mathbb{E}(y_t^i) = 1, \qquad & \sigma_c^2 &= \mathrm{Var}(y_t^i) = \tilde{\sigma}_S^2 + (2N_f - 1)\sigma_Z^2\, \tilde{\sigma}_Z^2 \\
\mu_w &= \mathbb{E}(y_t^j) = 0, \qquad & \sigma_w^2 &= \mathrm{Var}(y_t^j) = \tilde{\sigma}_Z^2 + (1 + \tilde{\sigma}_S^2)\, \sigma_Z^2 + (2N_f - 2)\sigma_Z^2\, \tilde{\sigma}_Z^2 \\
\mu_e &= \mathbb{E}(y_t^l) = 0, \qquad & \sigma_e^2 &= \mathrm{Var}(y_t^l) = (1 + \tilde{\sigma}_S^2)\, \sigma_Z^2 + (2N_f - 1)\sigma_Z^2\, \tilde{\sigma}_Z^2
\end{aligned}
\tag{98}
$$

where σ_Z^2, $\tilde{\sigma}_Z^2$, $\tilde{\sigma}_S^2$ are the variances of Z, $\tilde{Z}$ and $\tilde{S}$ respectively. Using Equations 72, 74, 75, we can use $\sigma_Z^2 = \frac{1}{D}$, $\tilde{\sigma}_Z^2 = \frac{1}{N}$, $\tilde{\sigma}_S^2 = \frac{1}{N} - \frac{1}{D}$ to compute entry variances σ_c^2, σ_w^2, σ_e^2 for the designed model output y_t.

Normal approximation. For large enough N_f, using Central Limit Theorem, we have at each entry a summation of many independent random variables. We can thus approximate:

$$
y_t^i \sim \mathcal{N}(1, \sigma_c^2), \qquad y_t^j \sim \mathcal{N}(0, \sigma_w^2), \qquad y_t^l \sim \mathcal{N}(0, \sigma_e^2)
\tag{99}
$$

Note that we have a single correct i entry, $n_w \equiv N_f - 1$ wrong j entries, and $n_e \equiv V - N_f$ empty l entries. The model outputs the correct entry if $i = \arg\max_p y_t^p$. In a probabilistic manner:

$$p_{\text{success}} = \Pr(y_t^i > y_t^p \quad \forall p \neq i) \tag{100}$$

Now let us condition on the realized value of the correct component $y_t^i = x$. Given this value, all other entries are independent. The probability that a single wrong entry satisfies $y_t^j < x$ is $\Phi(\frac{x}{\sigma_w})$, while the probability that a single empty entry satisfies $y_t^\ell < x$ is $\Phi(\frac{x}{\sigma_e})$, where Φ is the standard normal CDF. Therefore, the conditional probability that y_t^i exceeds all other entries is:

$$\Pr(y_t^i > y_t^p \; \forall p \neq i \mid y_t^i = x) = \left[\Phi\left(\frac{x}{\sigma_w}\right)\right]^{n_w} \left[\Phi\left(\frac{x}{\sigma_e}\right)\right]^{n_e} \tag{101}$$

The overall success probability is thus given by:

$$p_{\text{success}} = \int_{-\infty}^{\infty} \frac{1}{\sigma_c}\phi\left(\frac{x-1}{\sigma_c}\right) \left[\Phi\left(\frac{x}{\sigma_w}\right)\right]^{n_w} \left[\Phi\left(\frac{x}{\sigma_e}\right)\right]^{n_e} dx \tag{102}$$

where ϕ is the standard normal PDF.

Large-N_f approximation. In the limit of $N_f \gg N, D$, we claim the variance terms in Eq. 98 are all dominated by the "noise" component $2N_f\,\sigma_Z^2\,\tilde{\sigma}_Z^2$, and are thus approximately equal:

$$\sigma_c^2 \approx \sigma_w^2 \approx \sigma_e^2 \approx 2N_f\sigma_Z^2\,\tilde{\sigma}_Z^2 = \frac{2N_f}{ND} \equiv \sigma^2 \tag{103}$$

Using this approximation, the success probability in Eq. 102 becomes:

$$\begin{aligned} p_{\text{success}} &= \int_{-\infty}^{\infty} \frac{1}{\sigma}\phi\left(\frac{x-1}{\sigma}\right) \left[\Phi\left(\frac{x}{\sigma}\right)\right]^{n_w} \left[\Phi\left(\frac{x}{\sigma}\right)\right]^{n_e} dx = \\ &= \int_{-\infty}^{\infty} \frac{1}{\sigma}\phi\left(\frac{x-1}{\sigma}\right) \left[\Phi\left(\frac{x}{\sigma}\right)\right]^{V-1} dx = \\ &= \int_{-\infty}^{\infty} \phi\left(z - \frac{1}{\sigma}\right) \left[\Phi(z)\right]^{V-1} dz \end{aligned} \tag{104}$$

where we use $n_w + n_e = V - 1$ and $z = \frac{x}{\sigma}$.

Large-V approximation. If we further assume $V \gg 1$, which is typical for the AR task, the expression for the success probability can be rather simplified. Define $a \equiv \frac{1}{\sigma}$ and let $Z \sim \mathcal{N}(0,1)$. Then Eq. (104) can be rewritten as an expectation

$$p_{\text{success}} = \mathbb{E}\Big[\Phi(Z+a)^{V-1}\Big] \tag{105}$$

Equivalently, let $U_1, \ldots, U_{V-1} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$ be independent of Z, and denote their maximum by $M_{V-1} = \max_p U_p$. Conditioning on Z gives

$$\Pr\big(M_{V-1} \leq Z + a \,\big|\, Z\big) = \Phi(Z+a)^{V-1} \tag{106}$$

and therefore

$$p_{\text{success}} = \Pr\big(M_{V-1} \leq Z + a\big) = \Pr\big(Z + a \geq M_{V-1}\big) \tag{107}$$

The maximum of $V-1$ Gaussians concentrates near a typical level of

$$m_V = \sqrt{2\log V}\left(1 - \frac{\log\log V + \log 4\pi}{4\log V} + O\left(\left(\frac{\log\log V}{\log V}\right)^2\right)\right) \approx \sqrt{2\log V} \tag{108}$$

Approximating M_{V-1} by its typical value m_V yields:

$$p_{\text{success}} = \Pr\big(Z \geq M_{V-1} - a\big) \approx \Pr\big(Z \geq m_V - a\big) = \Phi(a - m_V) \tag{109}$$

Thus finally:

$$p_{\text{AR}} = p_{\text{success}} \approx \Phi\left(\frac{1}{\sigma} - m_V\right) = \Phi\left(\sqrt{\frac{ND}{2N_f}} - \sqrt{2\log V}\right) \tag{110}$$

MQAR. Given a query $q_t = k_m$ at $2N_f < t \leq L$, associated with a value v_i, and assuming large N_f, we now have $y_t^p \approx \delta^{ip} + \sum_{(a,b)}^{t-1} \epsilon_v^{pa}\,\epsilon_k^{bm}$, since we sum over all the token pairs up to index t. We thus have $y_t^p \sim \mathcal{N}(\delta^{ip}, \sigma_t^2)$, with $\sigma_t^2 \approx \frac{t}{ND}$. Repeating the $V \gg 1$ approximation, we have a

success probability for a single query given by $p(t) = \Pr\big(y_t^i > \sigma_t\, m_V\big) \approx \Phi(1/\sigma_t - m_V)$. The overall success probability is the average over queries, $p_{MQAR} = \frac{1}{L-2N_f}\sum_{t=2N_f}^{L} p(t)$, which yields Eq. 90.

□

F.3 Probabilistic Scaling Laws: Trained Model

Remark F.3 (Key-value selection mechanism.). Let us consider the following insight. If a realistic, compressive model is able to perfectly ignore non-key-value $(x_{\tau-1}, x_\tau)$ pairs, then its recall abilities may improve. Such non-key-value pairs include, i.e. value-key pairs between context facts, and key-key or value-value *random non-query* token pairs between queries.
Such selectivity is achievable if the learned embeddings E_k and E_v span orthogonal subspaces (which we denote shortly as $E_k \perp E_v$), and similarly for $\tilde{E}_k$ and $\tilde{E}_v$. Let us analyze it case by case, considering a query $x_t = k_m \in \mathcal{V}_k$.

- If $\tilde{E}_k \perp \tilde{E}_v$, then for all (v_i, x_τ) pairs in the context, we have $v_i^\top G_{\tilde{E}}\, k_m = 0$, which means these pairs are not written into the hidden state h_t. This implies all value-key and value-value pairs are ignored.
- Key-key pairs are written into h_t. However, if $E_k \perp E_v$, then for a key-key pair (k_l, k_n), given a value entry p where $e_p \in \mathcal{V}_v$, we have $e_p^\top G_E\, k_n = 0$. This implies that even though written into h_t, this pair has zero contribution when retrieved, thus the "value" k_n would not affect the output y_t.

The overall consequence of this selection mechanism is that only key-value pairs contribute to y_t.
From a mechanistic interpretability perspective, Fig. 15 tells us that, at least empirically, the model indeed learns such orthogonal embeddings, which justifies our conclusion.

Theorem F.4 (Probabilistic recall scaling laws). *A single-layer simplified Mamba can solve an AR task with a probability*

$$p_{\mathrm{AR}} \approx \Phi\left(\sqrt{\frac{ND}{N_f}} - \sqrt{2\log V}\right), \tag{111}$$

and an MQAR task of $L = 4N_f$ with a probability

$$p_{\mathrm{MQAR}} \approx \frac{1}{L-2N_f}\sum_{t=2N_f}^{L} \Phi\left(\sqrt{\frac{ND}{\frac{1}{2}N_f + \frac{1}{4}t}} - \sqrt{2\log V}\right). \tag{112}$$

in the limits of $N_f \gg N, D$ and $V \gg 1$, where Φ is the standard normal CDF.

Proof. (Also see proof for Thm. F.2.) As discussed in Rem. F.3, a simplified linear Mamba model can have a mechanism to ignore non-key-value pairs. More precisely, given a query $q_t = k_m$ we actually have $y_t^p \approx \delta^{ip} + \sum_{(a,b)_{kv}}^{t-1} \epsilon_v^{pa}\, \epsilon_k^{bm}$ where summation include only kv token pairs (x_a, x_b). As in Thm. F.2, we have $y_t^p \sim \mathcal{N}(\delta^{ip}, \sigma_t^2)$. However, now only kv pairs contribute to the accumulated variance. Therefore, $\sigma_t^2 = \frac{N_{kv}(t)}{ND}$, where $N_{kv}(t)$ is the expected number of kv pairs until t. For the AR task, we have a single query at $t = 2N_f + 1$, thus $N_{kv}(t) = N_f$: we have N_f facts (kv pairs) before t and N_f non-fact vk pairs, which have negligible variance. This implies $\sigma_t^2 = \frac{N_f}{ND}$, thus yields Eq. 111. For MQAR with a minimal $L = 4N_f$, the *queries* section is a sequence $q_1, r_1, \ldots, q_{N_f}, r_{N_f}$ where queries $q_t \in \mathcal{V}_k$ are padded with random tokens $r_t \in \mathcal{V}$, including both keys and values. This means that the $2N_f$ token pairs in the queries section are, on average, $\frac{1}{2}$ kk pairs, $\frac{1}{4}$ kv pairs, and $\frac{1}{4}$ vk pairs. Therefore, $N_{kv}(t) = N_f + (t - 2N_f)/4$, from which we obtain Eq. 112. □

F.4 Unified Recall Scaling Laws

Remark F.5 (Unified recall scaling laws). We further consider a higher-order correction and an additional simplification of Thm. F.2 and Thm. F.4. We derive a final simplified but useful form of the recall scaling laws. First, we relax the $N_f \gg N, D$ approximation; second, we use a linear approximation for Φ (which is reasonable for moderate $\sqrt{ND/N_f}$). Under these assumptions, a *single-layer simplified Mamba* solves associative recall tasks with a probability

$$p \approx \Phi\left(\sqrt{\frac{ND}{aN_f + N}} - \sqrt{2\log V}\right), \tag{113}$$

where

$$a \approx \begin{array}{c|cc} & \text{AR} & \text{MQAR} \\ \hline \text{Designed} & 2 & 3 \\ \text{Trained} & 1 & \frac{5}{4} \end{array} \tag{114}$$

Proof. Let us prove the statement case by case.

Designed model, moderate N_f. Recall Thm. F.2. Starting with Eq. 98, the mean and variance of the *correct*, *wrong* and *empty* coordinates of y_t can be approximated as:

$$\begin{aligned} \mu_c &= \mathbb{E}(y_t^i) = 1, \qquad \sigma_c^2 = \mathrm{Var}(y_t^i) \approx \tilde{\sigma}_S^2 + 2N_f\,\sigma_Z^2\,\tilde{\sigma}_Z^2 \\ \mu_w &= \mathbb{E}(y_t^j) = 0, \qquad \sigma_w^2 = \mathrm{Var}(y_t^j) \approx \tilde{\sigma}_Z^2 + \sigma_Z^2 + 2N_f\,\sigma_Z^2\,\tilde{\sigma}_Z^2 \\ \mu_e &= \mathbb{E}(y_t^l) = 0, \qquad \sigma_e^2 = \mathrm{Var}(y_t^l) \approx \sigma_Z^2 + 2N_f\,\sigma_Z^2\,\tilde{\sigma}_Z^2 \end{aligned} \tag{115}$$

where $\sigma_Z^2 = \frac{1}{D}$, $\tilde{\sigma}_Z^2 = \frac{1}{N}$, $\tilde{\sigma}_S^2 = \frac{1}{N} - \frac{1}{D}$. Note that if $N \sim D$ (same order of magnitude), $\tilde{\sigma}_S^2$ becomes a second order term, thus relatively negligible. In this case,

$$\begin{aligned} \sigma_c^2 &= \mathrm{Var}(y_t^i) \approx \frac{2N_f}{ND} \\ \sigma_w^2 &= \mathrm{Var}(y_t^j) \approx \frac{2N_f}{ND} + \frac{1}{D} + \frac{1}{N} \\ \sigma_e^2 &= \mathrm{Var}(y_t^l) \approx \frac{2N_f}{ND} + \frac{1}{D} \end{aligned} \tag{116}$$

Let us assume $1 < N_f \ll V$. Since there is $1 \ll V$ *correct* coordinate, $N_f - 1 \ll V$ *wrong* coordinates and $V - N_f$ *empty* coordinates, it is reasonable to approximate all of them to have a similar variance:

$$\sigma_c^2 \approx \sigma_w^2 \approx \sigma_e^2 \approx \frac{2N_f}{ND} + \frac{1}{D} = \frac{2N_f + N}{ND} \tag{117}$$

The above equation holds for an AR task, where the query is at $t = 2N_f$. More generally, for MQAR, given any single query at $2N_f < t \leq L$, we obtain the variance:

$$\sigma^2(t) \approx \frac{t}{ND} + \frac{1}{D} = \frac{t+N}{ND} \tag{118}$$

Then, we can apply the same derivation from Thm. F.2, which yields the following recall probabilities:

$$p_{\mathrm{AR}} \approx \Phi\left(\sqrt{\frac{ND}{2N_f + N}} - \sqrt{2\log V}\right), \tag{119}$$

$$p_{\mathrm{MQAR}} \approx \frac{1}{L - 2N_f} \sum_{t=2N_f}^{L} \Phi\left(\sqrt{\frac{ND}{t+N}} - \sqrt{2\log V}\right), \tag{120}$$

Linearization of Φ. To obtain a simple closed-form approximation for MQAR, we linearize around $\Phi = \frac{1}{2}$, i.e. around zero argument: $\Phi(x) \approx \frac{1}{2} + \frac{x}{\sqrt{2\pi}}$. Applying this approximation and then re-expressing the result in terms of Φ gives

$$p_{\mathrm{MQAR}} \approx \Phi\left(\frac{1}{L - 2N_f} \sum_{t=2N_f}^{L} \sqrt{\frac{ND}{t+N}} - \sqrt{2\log V}\right). \tag{121}$$

A coarse linearization would replace the mean by its value at the mean index, $\langle t \rangle = 3N_f$, yielding

$$p_{\mathrm{MQAR}} \approx \Phi\left(\sqrt{\frac{ND}{3N_f + N}} - \sqrt{2\log V}\right). \tag{122}$$

A slightly more accurate approximation is obtained by replacing the sum with an integral:

$$\frac{1}{L - 2N_f} \sum_{t=2N_f}^{L} \frac{1}{\sqrt{t+N}} \approx \frac{1}{L - 2N_f} \int_{2N_f}^{L} (t+N)^{-1/2}\,dt = \frac{2(\sqrt{L+N} - \sqrt{2N_f + N})}{L - 2N_f} \tag{123}$$

For the setting $L = 4N_f$, this becomes

$$p_{\text{MQAR}} \approx \Phi\left(\frac{\sqrt{ND}}{N_f}\left(\sqrt{4N_f + N} - \sqrt{2N_f + N}\right) - \sqrt{2\log V}\right). \tag{124}$$

We can rewrite this in the form

$$p_{\text{MQAR}} \approx \Phi\left(\sqrt{\frac{ND}{aN_f + N}} - \sqrt{2\log V}\right), \tag{125}$$

where

$$a = \frac{3 - \frac{N}{N_f} + \sqrt{\left(4 + \frac{N}{N_f}\right)\left(2 + \frac{N}{N_f}\right)}}{2}. \tag{126}$$

This implies $2.91 < a < 3$, so $a \approx 3$ is an accurate approximation.

Trained models. We follow the proof of Thm. F.4, and apply the same moderate-N_f correction as above. We have $\sigma^2(t) = \frac{N_{kv}(t)}{ND} + \frac{1}{D}$, where $N_{kv}(t) = N_f + (t - 2N_f)/4$ is the expected number of kv pairs until t. Thus,

$$\sigma^2(t) = \frac{N_f + \frac{1}{4}(t - 2N_f)}{ND} + \frac{1}{D} = \frac{\frac{1}{2}N_f + \frac{1}{4}t + N}{ND} \tag{127}$$

This allows us to express the success probability for associative recall tasks as:

$$p_{\text{AR}} \approx \Phi\left(\sqrt{\frac{ND}{N_f + N}} - \sqrt{2\log V}\right), \tag{128}$$

$$p_{\text{MQAR}} \approx \frac{1}{L - 2N_f}\sum_{t=2N_f}^{L} \Phi\left(\sqrt{\frac{ND}{\frac{1}{2}N_f + \frac{1}{4}t + N}} - \sqrt{2\log V}\right). \tag{129}$$

For MQAR, linearizing around $\Phi = \frac{1}{2}$ and re-expressing the result gives:

$$p_{\text{MQAR}} \approx \Phi\left(\frac{1}{L - 2N_f}\sum_{t=2N_f}^{L} \sqrt{\frac{ND}{\frac{1}{2}N_f + \frac{1}{4}t + N}} - \sqrt{2\log V}\right) \tag{130}$$

A coarse linearization replaces the mean by its value at the mean index, $\langle t \rangle = 3N_f$, yielding

$$p_{\text{MQAR}} \approx \Phi\left(\sqrt{\frac{ND}{\frac{5}{4}N_f + N}} - \sqrt{2\log V}\right). \tag{131}$$

A slightly more accurate approximation is obtained by replacing the sum with an integral:

$$\frac{1}{L - 2N_f}\sum_{t=2N_f}^{L} \frac{1}{\sqrt{2N_f + t + 4N}} \approx \frac{2\left(\sqrt{L + 2N_f + 4N} - \sqrt{4N_f + 4N}\right)}{L - 2N_f}. \tag{132}$$

For the setting $L = 4N_f$, this gives

$$p_{\text{MQAR}} \approx \Phi\left(\frac{2\sqrt{ND}}{N_f}\left(\sqrt{6N_f + 4N} - \sqrt{4N_f + 4N}\right) - \sqrt{2\log V}\right) = \Phi\left(\sqrt{\frac{ND}{aN_f + N}} - \sqrt{2\log V}\right), \tag{133}$$

where

$$aN_f + N = \frac{\left(\sqrt{6N_f + 4N} + \sqrt{4N_f + 4N}\right)^2}{16} = \frac{5}{4}N_f + N + \frac{1}{4}\sqrt{(3N_f + 2N)(N_f + N)}. \tag{134}$$

Thus, the refined denominator is close to our coarse approximation $\frac{5}{4}N_f + N$, up to an additive term which can be used to derive higher-order corrections.

□

Remark F.6 (**Scaling laws for model dimensions**). Using Eq. 15, the model can solve AR with a probability greater than $1 - \delta$ provided the model dimensions satisfy $\sqrt{ND/N_f} \gtrsim \sqrt{2\log V} + n_\sigma$,

where $n_\sigma = \Phi^{-1}(1-\delta)$. For $\delta \ll 1$, we approximate $n_\sigma^2 \approx 2\log(1/\delta)$, and the dimensions requirement becomes, up to constant factors:

$$ND = \Omega\big(N_f \log\big(\tfrac{V}{\delta}\big)\big) \tag{135}$$

F.5 Information-Theoretic Lower Bound for Model Dimensions

Lemma F.7 (Information-theoretic lower bound on state size). *Consider a recurrent model with a finite state size, where the context is summarized in a hidden state of S scalar entries at b bits of precision each. If such a model solves an AR task with N_f facts (or an MQAR task with N_f facts and context length $L = 4N_f$) over a vocabulary of size V with success probability $> 1-\delta$, for a constant $0 < \delta < 1$, then $bS = \Omega(N_f \log V)$.*

Proof. Following the AR task definition, we note that the N_f facts consist of distinct keys from the key vocabulary $\mathcal{V}_k$ and of values drawn i.i.d. uniformly from the value vocabulary $\mathcal{V}_v$ (which size is $|\mathcal{V}_v| = V_v = V/2$), and that the query is drawn uniformly among the context's N_f keys, independently of the values. The context thus carries $N_f \log_2 V_v$ bits of information about the fact values $v_{1:N_f} \equiv (v_1, \ldots, v_{N_f})$. Consider the recurrent state at the end of the facts section, h_T, where $T = 2N_f$. It is a deterministic function of the context, and from this point on the model's answers depend on the values only through h_T. The state consists of S scalar entries at b bits of precision each, so it can take at most 2^{bS} distinct configurations. Consequently, the state retains at most bS bits of information about the values: $\mathbb{I}(v_{1:N_f}; h_T) \le bS$, where $\mathbb{I}(\cdot\,;\cdot)$ denotes mutual information.
On the other hand, since the query may be any of the context's keys, all values can be recovered from h_T: running the model from h_T separately with any single query $q \in \{x_0, x_2, \ldots, x_{T-2}\}$, and collecting the answers, forms a decoder for $v_{1:N_f}$. By Fano's inequality, any decoder that recovers $v_{1:N_f}$ with error probability smaller than δ must satisfy $\mathbb{H}(v_{1:N_f} \mid h_T) \le 1 + \delta\, N_f \log_2 V_v$, where $\mathbb{H}(\cdot)$ denotes the Shannon entropy. Since $\mathbb{H}(v_{1:N_f}) = N_f \log_2 V_v$, the state must retain $\mathbb{I}(v_{1:N_f}; h_T) = \mathbb{H}(v_{1:N_f}) - \mathbb{H}(v_{1:N_f} \mid h_T) \ge (1-\delta)\, N_f \log_2 V_v - 1$.
Combining the two: $bS \ge (1-\delta)\, N_f \log_2 V_v - 1$, and since $V_v = V/2$, for a constant $0 < \delta < 1$ this gives $bS = \Omega(N_f \log V)$. The same bound holds for MQAR, since $L = 4N_f$. □

Remark F.8 (Information-theoretic lower bound for Mamba dimensions). Notably, Lemma F.7 assumes nothing about the recurrence itself, only that the context is summarized in a fixed-size state. It therefore holds for both the simplified linear model and the *full nonlinear Mamba*, whose hidden state has the same dimensions.

Single-layer Mamba. For a *single-layer simplified Mamba*, the recurrent state is $h_t \in \mathbb{R}^{D_{\text{in}} \times N}$ (where $D_{\text{in}} = \text{expand}{\cdot}D$; in our scope we usually pick $\text{expand} = 2$), i.e. $S = \text{expand}{\cdot}ND = 2ND$ entries. At fixed precision b, Lemma F.7 thus reads:

$$ND = \Omega(N_f \log V) \tag{136}$$

Multi-layer Mamba. For models with Λ layers (both linear and full), the total recurrent state across layers has $S = \text{expand} \cdot \Lambda\, ND$ entries, and the bound becomes:

$$\Lambda\, ND = \Omega(N_f \log V) \tag{137}$$

G Multi-Layer Recall Circuits

G.1 Multi-Layer Mamba

In favor of the analysis of a multi-layer Mamba model, we present the following simplified model, detailed in Alg. 3. As in Alg. 1, we assume gating, discretization, nonlinearities, biases and normalization layers are all removed. Note, however, the insertion of residual connections, as they are important for multi-layer recall, as we soon observe.

G.2 Multi-Layer Recall Circuit and Scaling Laws

Intuition. Let us briefly describe the mechanism. The residual connections between layers allow each Mamba layer l to receive, as an input x^l, a linear combination of the original sequence Ex and the current cumulative recall output $\sum_k^{l-1} y_{AR}^k$ as an input. Each layer then adds its own independent associative recall result $y_{AR}^l = Ex\, \alpha_{AR}^l$, which is then similarly passed through a residual connection to the next layer, such that model output is $y = \sum_k^{\Lambda} y_{AR}^k$.

Algorithm 3 Simplified Multi-Layer Mamba

Require: $x \in \mathbb{R}^{V \times L}$
Ensure: $y \in \mathbb{R}^{V \times L}$
Parameters: $E,\ \{P^l_{\text{in}},\ P^l_{\text{out}},\ W^l_{\text{conv}},\ S^l_B,\ S^l_C \mid l = 1, \dots, \Lambda\}$
$x_{\text{e}} \leftarrow E\, x$
$x^1 \leftarrow x_{\text{e}}$
for $l \leftarrow 1$ **to** Λ **do**
  Mixer:
  $x^l_{\text{in}} \leftarrow P^l_{\text{in}}\, x^l$
  $\hat{x}^l \leftarrow \text{Conv1D}(x^l_{\text{in}}, W^l_{\text{conv}})$
  $B^l,\ C^l \leftarrow S^l_B\, \hat{x}^l,\ S^l_C\, \hat{x}^l$
  $\hat{y}^l \leftarrow \text{SSM}(\hat{x}^l, B^l, C^l)$
  $y^l_{\text{out}} \leftarrow P^l_{\text{out}}\, \hat{y}^l$
  $y^l \leftarrow x^l + y^l_{\text{out}}$
  $x^{l+1} \leftarrow y^l$
end for
$y \leftarrow E^\top\, y^\Lambda$
return y

Theorem G.1 (Multi-layer recall scaling laws). *A simplified multi-layer Mamba with Λ layers can solve recall tasks with the same probability as a single-layer model with the same dimensions and state size $N^{\text{eff}} = \Lambda N$. In particular, the model can solve an AR task with a probability*

$$p_{\text{AR}} \approx \Phi\left(\sqrt{\frac{\Lambda N D}{N_f}} - \sqrt{2 \log V}\right), \tag{138}$$

and an MQAR task of $L = 4N_f$ with a probability

$$p_{\text{MQAR}} \approx \frac{1}{L - 2N_f} \sum_{t=2N_f}^{L} \Phi\left(\sqrt{\frac{\Lambda N D}{\frac{1}{2}N_f + \frac{1}{4}t}} - \sqrt{2 \log V}\right). \tag{139}$$

in the limits of $N_f \gg N, D$ and $V \gg 1$, where Φ is the standard normal CDF. Following Rem. F.5, we can refine and rewrite these results as

$$p_{\text{recall}} \approx \Phi\left(\sqrt{\frac{\Lambda N D}{a N_f + \Lambda N}} - \sqrt{2 \log V}\right), \tag{140}$$

where $a = 1$ for AR, and $a \approx \frac{5}{4}$ for MQAR.

Proof. Let us construct the multi-layer Mamba solution to MQAR step by step.

Step 1: Single-layer Mamba First, we show a single-layer Mamba with residual connection can solve MQAR. We slightly modify the weights construction of Theorem 3.2, such that residual connection does not affect the recall circuit output.

Weights. Given embedding matrices $E \in \mathbb{R}^{D \times V}$ and $F^1 \in \mathbb{R}^{N \times D}$, and given a parameter $a^1 \in \mathbb{R}$, we choose:

$$\begin{gathered} P^1_{\text{in}} = \begin{pmatrix} I_D \\ I_D \end{pmatrix}, \quad P^1_{\text{out}} = \begin{pmatrix} 0 \mid I_D \end{pmatrix}, \\ W^1_{\text{conv}} = \begin{pmatrix} 1_D & | & 0_D \\ 0_D & | & 1_D \end{pmatrix}, \quad S^1_B = \begin{pmatrix} F^1 \mid a^1 F^1 \end{pmatrix}, \quad S^1_C = \begin{pmatrix} 0 \mid F^1 \end{pmatrix} \end{gathered} \tag{141}$$

As before, let us define $\tilde{E}^1 \equiv F^1 E \in \mathbb{R}^{N \times V}$. Inside the SSM, we now have $\hat{x}^1 \in \mathbb{R}^{D_{\text{in}} \times L}$ such that $\hat{x}^1_t = \begin{pmatrix} E\, x_{t-1} \\ E\, x_t \end{pmatrix}$, $C^1_t = \tilde{E}^1\, x_t$, and most importantly $B^1_t = \tilde{E}^1\, x_{t-1} + a^1 \tilde{E}^1\, x_t$.

Attention view. We now leverage *attention* formulation to view SSM operation. Namely, the SSM attention matrix $\alpha^1 \in \mathbb{R}^{L \times L}$ should satisfy $\hat{y}^1_t = \sum_{\tau=0}^{t} \hat{x}^1_\tau\, \alpha^1_{\tau t}$, or in matrix form $\hat{y}^1 = \hat{x}^1 \alpha^1 \in \mathbb{R}^{D_{\text{in}} \times L}$. (Note the right-side matrix multiplication, which operates along time axis rather than dimension axis). Following Ali et al. [51], Dao and Gu [48], the attention matrix for our simplified SSM is:

$$\alpha^1_{\tau t} = {B^1_\tau}^\top\, C^1_t = (\tilde{E}^1\, x_{\tau-1} + a^1 \tilde{E}^1\, x_\tau)^\top (\tilde{E}^1\, x_t) \tag{142}$$

As in Theorem 3.2, we choose E, F^1 such that $\tilde{E}^1 \in \mathbb{R}^{N \times V}$ is a JL embedding matrix w.r.t some small $\varepsilon_k > 0$. Defining $G^1 \equiv (\tilde{E}^1)^\top \tilde{E}^1$ for short, we have:

$$\alpha^1_{\tau t} = x^\top_{\tau-1} G^1\, x_t + a^1\, x^\top_\tau G^1\, x_t \tag{143}$$

Alternatively, minding that for a JL matrix, $G^1 = I_V + O(\varepsilon_k)$,

$$\alpha^1_{\tau t} = x^\top_{\tau-1} G^1 x_t + a^1\, x^\top_\tau x_t + O(\varepsilon_k) \tag{144}$$

which yields the $L \times L$ matrix form

$$\alpha^1 = \alpha^1_{AR} + a^1 I_L + O(\varepsilon_k) \tag{145}$$

where $(\alpha^1_{AR})_{\tau t} = x_{\tau-1}^\top G^1 x_t$ is the compressive-recall attention matrix equivalent of Theorem 3.1 (which performs recall by matching MQAR queries with keys), and $I_L \in \mathbb{R}^{L\times L}$ is the identity matrix in sequence space. We denote α^1_{AR} this way to avoid confusion with the overall SSM attention matrix α^1.

Layer output. Given the input sequence $x^1 = Ex \in \mathbb{R}^{D\times L}$, the SSM output is:

$$\hat{y}^1 = \hat{x}^1\alpha^1 = P^1_{\text{in}} Ex(\alpha^1_{AR} + a^1 I_L) = P^1_{\text{in}} Ex\alpha^1_{AR} + a^1 P^1_{\text{in}} Ex \tag{146}$$

When projected out, since $P^1_{\text{in}} = \left(\begin{smallmatrix} I_D \\ I_D \end{smallmatrix}\right), P^1_{\text{out}} = \left(0 \mid I_D\right)$, we have:

$$y^1_{\text{out}} = P^1_{\text{out}}\, \hat{y}^1 = \hat{x}^1\alpha^1 = Ex\alpha^1_{AR} + a^1 Ex \tag{147}$$

The residual connection now implies:

$$y^1 = x^1 + y^1_{\text{out}} = Ex\alpha^1_{AR} + (a^1 + 1)Ex \tag{148}$$

We are now able to choose $a^1 = -1$, such that

$$y^1 = Ex\alpha^1_{AR} \tag{149}$$

and model output is

$$y = E^\top y^1 = E^\top Ex\alpha^1_{AR} \tag{150}$$

which is, by design, the same solution to MQAR as in Theorem 3.2.

Step 2: Two-layer Mamba Building upon previous step, we now describe a two-layer Mamba solution to MQAR.
Assume a *simplified multi-layer Mamba* with $\Lambda = 2$. We set first layer ($l = 1$) weights as in Step 1, but this time without specifying $a^1 = -1$.

Layer input. Importantly, the second layer ($l = 2$) input is:

$$x^2 = y^1 = x^1 + y^1_{\text{out}} = Ex\alpha^1_{AR} + (a^1 + 1)Ex \tag{151}$$

For clarity, let us denote $y^1_{AR} = Ex\alpha^1_{AR}$ and $b = a^1 + 1$. Layer input now reads:

$$x^2 = y^1_{AR} + bEx \tag{152}$$

Weights. Given a matrix $F^2 \in \mathbb{R}^{N\times D}$ and a parameter $a^2 \in \mathbb{R}$, let us set layer weights as follows:

$$\begin{gathered} P^2_{\text{in}} = \left(\begin{smallmatrix} I_D \\ I_D \end{smallmatrix}\right), \quad P^2_{\text{out}} = \left(0 \mid I_D\right), \\ W^2_{\text{conv}} = \left(\begin{smallmatrix} 1_D & \mid & 0_D \\ 0_D & & 1_D \end{smallmatrix}\right), \quad S^2_B = \frac{1}{b}\left(F^2 \mid a^2 F^2\right), \quad S^2_C = \frac{1}{b}\left(0 \mid F^2\right) \end{gathered} \tag{153}$$

Key and value subspaces. Inside layer 2 SSM, after input projections, we have $\hat{x}^2_t = \left(\begin{smallmatrix} bE\, x_{t-1} + (y^1_{AR})_{t-1} \\ bE\, x_t + (y^1_{AR})_t \end{smallmatrix}\right)$. Importantly, note that $\mathbb{R}^D$ can be decomposed into two orthogonal key and value subspaces, namely $\mathbb{R}^D = U \oplus W$, such that $E = E_k + E_v$ and $(E_k)_i \perp (E_v)_j$ for any i, j. Consequently, we can choose F^2 which acts only on embedded key vectors $Ek_i \in U$ and zeros out embedded value vectors $Ev_j \in W$: for any j, $F^2 Ev_j = 0$.
Notice that, as a result of Theorem 3.2, any $(y^1_{AR})_t$ is approximately a *value* vector, up to a small $O(\varepsilon_k + \varepsilon_v)$ error. (This can be further guaranteed by modifying layer 1 output projection P^1_{out} to zero out the key subspace U.) This implies $F(y^1_{AR})_t \approx 0$, for any t. Then, upon SSM projections, we have, approximately, $C^2_t = \tilde{E}^2\, x_t$ and $B^2_t = \tilde{E}^2\, x_{t-1} + a^2 \tilde{E}^2\, x_t$, without any y^1_{AR} terms, where we have defined $\tilde{E}^2 = F^2 E$.

Attention view. Replicating the attention matrix computation from Step 1, and similarly defining $G^2 \equiv (\tilde{E}^2)^\top \tilde{E}^2$, we now have for layer 2 SSM:

$$\alpha^2 = \alpha^2_{AR} + a^2 I_L + O(\varepsilon_k) \tag{154}$$

where $(\alpha^2_{AR})_{\tau t} = x_{\tau-1}^\top G^2 x_t$ is a compressive-recall attention matrix. Note that $G^2 \neq G^1$ in general, since generally $F^2 \neq F^1$.

Layer output. Following Step 1, we compute the the SSM output:

$$\hat{y}^2_{\text{out}} = P^2_{\text{out}}\,\hat{y}^2 = P^2_{\text{out}}\,\hat{x}^2\alpha^2 = P^2_{\text{out}}\,P^2_{\text{in}}\,x^2(\alpha^2_{AR} + a^2 I_L) \tag{155}$$

But, since projections are trivial:

$$y^2_{\text{out}} = x^2(\alpha^2_{AR} + a^1 I_L) \tag{156}$$

Considering residual connection, the layer output is:

$$y^2 = x^2 + y^2_{\text{out}} = x^2(\alpha^2_{AR} + b^2 I_L) \tag{157}$$

where $b^2 = a^2 + 1$. Recall that $x^2 = y^1_{AR} + b^1 Ex$. Therefore,

$$y^2 = (y^1_{AR} + b^1 Ex)(\alpha^2_{AR} + b^2 I_L) \tag{158}$$

We denote $y^2_{AR} = Ex\alpha^2_{AR}$, since it is a compressive-recall output, similarly to these in Step 1 and in Theorem 3.2. This yields:

$$y^2 = y^1_{AR}\,\alpha^2_{AR} + b^1 y^2_{AR} + b^2 y^1_{AR} + b^1 b^2 Ex \tag{159}$$

Note that we have additional degrees of freedom by choosing the scales of b^1, b^2 and the norms of E_k, E_v, F^1, F^2, w.r.t to some arbitrary scale factor ε. This can be done such that:

$$y^2 = b^1 y^2_{AR} + b^2 y^1_{AR} + O(\varepsilon) \tag{160}$$

This, of course, does not mean the $O(\varepsilon)$ terms are negligible, but rather they have weaker contribution to model output, hence are less probable to change the maximal entry of retrieved value.

Effective state size. Importantly, both y^1_{AR} and y^2_{AR} are linear mixtures of the input sequence Ex, namely, $y^l_{AR} = Ex\alpha^l_{AR}$. Let us absorb the constants b^1, b^2 into $\alpha^2_{AR}, \alpha^1_{AR}$, and omit the error term:

$$y^2 \approx Ex(\alpha^1_{AR} + \alpha^2_{AR}) \equiv Ex\,\alpha^{\text{eff}} \tag{161}$$

Moreover, the layers attention matrices are of a similar form, $(\alpha^l_{AR})_{\tau t} = x^\top_{\tau-1} G^l x_t$. Hence:

$$\alpha^{\text{eff}}_{\tau t} = (\alpha^1_{AR} + \alpha^2_{AR})_{\tau t} = x^\top_{\tau-1}(G^1 + G^2)\,x_t \tag{162}$$

Since $G^l = \tilde{E}^{l\top}\tilde{E}^l = E^\top F^{l\top} F^l E$, we conclude:

$$\alpha^{\text{eff}}_{\tau t} = x^\top_{\tau-1} E^\top (F^{1\top} F^1 + F^{2\top} F^2) E\,x_t \tag{163}$$

This motivates us to define an effective matrix,

$$F^{\text{eff}} \equiv \begin{pmatrix} F^1 \\ F^2 \end{pmatrix} \in \mathbb{R}^{2N\times D} \tag{164}$$

and effective net embedding:

$$\tilde{E}^{\text{eff}} \equiv F^{\text{eff}} E \in \mathbb{R}^{2N\times V} \tag{165}$$

Finally, the model output now becomes:

$$y_t = E^\top y^2_t = \sum_{\tau=0}^{t} E^\top (E\,x_\tau)\,(\tilde{E}^{\text{eff}}\,x_{\tau-1})^\top (\tilde{E}^{\text{eff}}\,x_t) \tag{166}$$

Hence, our multi-layer model with $\Lambda = 2$ is mathematically equivalent to a single-layer model with larger state size, $N^{\text{eff}} = 2N = \Lambda N$.

Step 3: Multi-layer Mamba Statement is trivially proven by induction. We use $\Lambda = 1$ (Step 1) as a *base step*, and a similar claim as done for $\Lambda = 2$ (Step 2) as and *induction step*.
More intuitively: Step 2 shows that a Mamba layer l which receives a linear combination $x^l = \sum_k^{l-1} a^k\, y^k_{AR} + b^l\, Ex$ as an input, can output a similar (independent) associative-recall result $y^l_{AR} = Ex\,\alpha^l_{AR}$, which is then similarly passed through a residual connection to the next layer, as $x^{l+1} = y^l_{AR} + \sum_k^{l-1} a^k\, y^k_{AR} + b^l\, Ex$.
Conclusion is, a simplified multi-layer Mamba model with Λ layers can solve MQAR equivalently to a single-layer model with a larger state size $N^{\text{eff}} = \Lambda N$. □

H Multi-Head Recall Circuits and Mamba-2

H.1 Multi-Head Patterns

Multi-head patterns for SSM. One of the key architectural innovations introduced in Mamba-2 [48], compared to Mamba, is the ability to process the input sequence through multiple SSM heads

in parallel. Equivalently to parameter sharing across attention heads in Transformers, the authors present several ***head patterns*** for multi-head SSM (see Tab. 5). In each head pattern, a different combination of $\hat{x}$, B and C is shared across SSM heads, correspondingly to sharing V, K and Q between attention heads in a Transformer. Specifically, MVA shares B, C between the SSM heads, MKA shares C, MQA shares B, and MHA shares none.

Table 5: Multi-Head Patterns in Mamba-2

	Multi-Head SSM	**Multi-Contract SSM**	**Multi-Expand SSM**	**Multi-Input SSM**
	Multi-Head Attn.	Multi-Query Attn.	Multi-Key Attn.	Multi-Value Attn.
	MHA	*MQA*	*MKA*	*MVA*
$\hat{x}$	(L, M, P)	$(L, 1, P)$	$(L, 1, P)$	(L, M, P)
B	(L, M, N)	$(L, 1, N)$	(L, M, N)	$(L, 1, N)$
C	(L, M, N)	(L, M, N)	$(L, 1, N)$	$(L, 1, N)$

It is assumed that the number of heads M divides model dimension D, such that head dimension is $P = \frac{D}{M}$. Also note that as done in Eq. 4 and Alg. 1, we assume SSM matrix A is an identity operation, hence omit it. Importantly, the head pattern actually *used* in Mamba-2 in practice is MVA. Here, however, we study all patterns, in order to better understand recall capacity in Mamba-2.

Simplified multi-head model. Following these head patterns, we now generalize our single-head simplified model (Alg. 1) to match Mamba-2 multi-head SSM patterns. In Alg. 4, we present a simplified multi-head model, which follows the Mamba-2 design but keeps critical components only. Also notice the different structure of projections and convolutions (compared to Mamba) used to produce $\hat{x}$, B and C from input sequence Ex. See App. H.3 (Table 6 and Table 7) for notation and dimensions details. It should be noted that a *single-layer (single-head) simplified Mamba* is roughly a single-head redundant case of Alg. 4, with $M = 1$, $P = D_{\text{in}}$, and $p = \text{MVA}$.

Algorithm 4 Simplified Single-Layer Mamba-2

Require: $x \quad (L, V)$
Ensure: $y \quad (L, V)$
Parameters: $E,\ P_{\text{in}},\ P_{\text{out}},\ W_{\text{conv}},\ W_x,\ W_y$
Head Pattern: $p \in \{\text{MHA, MQA, MKA, MVA}\}$
$x_{\text{e}} \leftarrow E\,x \quad (L, D)$
Mixer:
$\hat{x} \leftarrow \text{Conv1D}(P_{\text{in}}^{\hat{x}}\, x_{\text{e}}, W_{\text{conv}}^{\hat{x}})$
$B,\ C \leftarrow S_{B,C}\,\text{Conv1D}(P_{\text{in}}^{B,C}\, x_{\text{e}}, W_{\text{conv}}^{B,C})$
$\hat{x}: \quad (L, D)$
$B: \begin{cases} (L, N) & \text{if } p \in \{\text{MQA, MVA}\} \\ (L, N, M) & \text{if } p \in \{\text{MHA, MKA}\} \end{cases}$
$C: \begin{cases} (L, N) & \text{if } p \in \{\text{MKA, MVA}\} \\ (L, N, M) & \text{if } p \in \{\text{MHA, MQA}\} \end{cases}$
$\hat{x}_{\text{heads}} \leftarrow \text{split}(\hat{x}, M) \quad (L, P, M)$
$\hat{x}_{\text{shared}} \leftarrow \text{contract}(\hat{x}_{\text{heads}}, W_x) \quad (L, P)$
for $h \leftarrow 1$ **to** M **do**
  $\hat{x}^h \leftarrow \begin{cases} \hat{x}^h_{\text{heads}} & \text{if } p \in \{\text{MHA, MVA}\} \\ \hat{x}_{\text{shared}} & \text{if } p \in \{\text{MQA, MKA}\} \end{cases} \quad (L, P)$
  $\hat{y}^h \leftarrow \text{SSM}(\hat{x}^h, B^h, C^h) \quad (L, P)$
end for
$\hat{y} \leftarrow \text{aggregate}(\{\hat{y}^h\}, W_y) \quad (L, D)$
$y_{\text{out}} \leftarrow P_{\text{out}}\,\hat{y} \quad (L, D)$
$y \leftarrow E^\top\, y_{\text{out}} \quad (L, V)$
return y

H.2 Multi-Head Recall Circuits

Upon defining the multi-head SSM, one may ask whether it is beneficial in terms of associative recall capabilities, and if so, which head pattern should be chosen. Equipped with the simplified Mamba-2 model, let us now analyze the associative recall capacity of each head pattern.

Theorem H.1 (Multi-head recall scaling laws). *Given fixed model dimensions D, N, single-layer simplified Mamba-2 models with $M > 1$ heads and multi-head attention patterns MQA, MKA, MVA, or MHA solve recall tasks (AR or MQAR) with probabilities satisfying*

$$p_{\text{MQA}} = p_{\text{MKA}} \leq p_{\text{single}} = p_{\text{MVA}} \leq p_{\text{MHA}}\,, \tag{167}$$

where p_{single} is the recall probability for a single-head model ($M = 1$) of similar dimensions D, N, as detailed in Thm. 4.2.

Proof. See the following proofs for lemmas H.2, H.3, H.4 and H.5. □

Lemma H.2 (MVA recall circuit). *A single-layer simplified Mamba-2 model with M heads using MVA pattern and dimensions D, N (with $\text{expand} = 2$, $D_{\text{conv}} = 2$, as in Thm. 3.2) is **equivalent**, in term of associative recall (AR or MQAR tasks), to a similar single-head model of the same dimensions $D' = D$, $N' = N$. Specifically, the model probability of success satisfies:*

$$p_{\text{MVA}}[D, N, M] = p_{\text{single}}[D, N] \tag{168}$$

where $p_{\text{single}}[D', N']$ is the recall probability for a single-head model ($M = 1$) of dimensions D', N', as detailed in Thm. 4.2.

Proof. We use the compressed recall weights as in Theorem 3.2, with slight adaptations to Mamba-2. Given a matrix $F \in \mathbb{R}^{N \times D}$, we choose weights $W^{\hat{x}}_{\text{conv}} = \big(0_D \mid 1_D\big)$, $W^{B}_{\text{conv}} = \big(1_D \mid 0_D\big)$, $W^{C}_{\text{conv}} = \big(0_D \mid 1_D\big)$ and $S_B = S_C = F$, such that values are $\hat{x}_t = E\, x_t$, keys are $B_t = FE\, x_{t-1}$ and queries are $C_t = FE\, x_t$. The value $\hat{x}_t$ is then split into M heads $\hat{x}^h_t$:

$$\hat{x}_t = \begin{pmatrix} \hat{x}^1_t \\ \vdots \\ \hat{x}^M_t \end{pmatrix} \tag{169}$$

In each SSM head, we have:

$$\hat{y}^h_t = \sum_{\tau}^{t} \hat{x}^h_\tau \, B_\tau^\top \, C_t \tag{170}$$

Assume, for simplicity, heads aggregation has trivial weights $W^h_y = 1$ (see Alg. 4); the SSM output is thus $\{\hat{y}^h_t\}$ stacked:

$$\hat{y}_t = \begin{pmatrix} \hat{y}^1_t \\ \vdots \\ \hat{y}^M_t \end{pmatrix} = \begin{pmatrix} \sum_\tau^t \hat{x}^1_\tau \, B_\tau^\top \, C_t \\ \vdots \\ \sum_\tau^t \hat{x}^M_\tau \, B_\tau^\top \, C_t \end{pmatrix} = \sum_{\tau}^{t} \begin{pmatrix} \hat{x}^1_\tau \\ \vdots \\ \hat{x}^M_\tau \end{pmatrix} B_\tau^\top \, C_t = \sum_{\tau}^{t} \hat{x}_\tau \, B_\tau^\top \, C_t \tag{171}$$

which is exactly the output we have for a single-head model of the same dimensions D, N.

□

Lemma H.3 (MQA recall circuit). *A single-layer simplified Mamba-2 model with M heads using MQA pattern and dimensions D, N (with $\text{expand} = 2$, $D_{\text{conv}} = 2$, as in Thm. 3.2) is **equivalent**, in term of associative recall (AR or MQAR tasks), to a similar single-head model of dimensions $D' = \frac{D}{M}$, $N' = N$. Specifically, the model probability of success satisfies:*

$$p_{\text{MQA}}[D, N, M] = p_{\text{single}}[\tfrac{D}{M}, N] \leq p_{\text{single}}[D, N] \tag{172}$$

(Namely, for MQA pattern, splitting into heads might hurt recall performance, despite using a larger query dimension MN.)

Proof. We choose convolution kernels as done in Lem. H.2, and fix $S_B = F$ for $F \in \mathbb{R}^{N \times D}$. We now have M distinct projections $S^h_C = F^h \in \mathbb{R}^{N \times D}$, for generally different matrices $\{F^h\}$. As before, we have shared keys $B_t = FE\, x_{t-1}$, but now distinct queries $C^h_t = F^h E\, x_t$.

Non-contracted values and attention view. Let us first ignore values $\hat{x}$ contraction, i.e. assume the SSM input is $\hat{x}_{\text{heads}} \in \mathbb{R}^{D \times L}$ instead of $\hat{x}_{\text{shared}} \in \mathbb{R}^{P \times L}$. Per time t, the input value is thus $\hat{x}_t = Ex_t$. Recall that each SSM head can be equivalently thought as of an attention head (see Ali et al. [51], Dao and Gu [48]), with attention matrix $\alpha^h_{\tau t} = {B^h_\tau}^\top C^h_t$. In our case:

$$\alpha^h_{\tau t} = {B_\tau}^\top C^h_t = x^\top_{\tau-1} E^\top F^\top F^h E\, x_t \tag{173}$$

The ideal attention matrix for associative recall (see Thm. 3.1) is $\alpha^h_{\tau t} = x^\top_{\tau-1} x_t$, which matches queries to keys. As in Thm. 3.2, this attention matrix can be approximated using JL matrices for compression. In our case, we should require $E^\top F^\top F^h E \approx I_V$. This can be achieved if $\tilde{E} \equiv FE$ is a JL matrix (Lem. C.3) and also $F^h = F$ for all heads h. Only this way, we can achieve $\alpha^h_{\tau t} \approx x^\top_{\tau-1} x_t$. Note, however, this implies our queries are not independent anymore, as now $C^h_t = FEx_t$ are in fact shared between all heads; the additional weights become redundant. This means the recall circuit is

equivalent to the one in Lem. H.2, thus performs equivalently to a single-head model with $D' = D$, $N' = N$.

Values contraction. We now consider the actual SSM input, which is given by:

$$\hat{x}_{\text{shared}} = \text{contract}(\hat{x}_{\text{heads}}, W_x) = \sum_{h=1}^{M} W_x^h \, \hat{x}_{\text{heads}}^h \in \mathbb{R}^{P \times L} \tag{174}$$

Note that in SSM terms, this transformation further compresses values $\hat{x}$ into a smaller dimension $P = \frac{D}{M}$, but does not affect key and query dimensions. Therefore, the multi-head model is equivalent to a single-head model with $N' = N$ and $D' = \frac{D}{M} \leq D$. This consequence yields Eq. 172. □

Lemma H.4 (MKA recall circuit). *A single-layer simplified Mamba-2 model with M heads using MKA pattern and dimensions D, N (with $\text{expand} = 2$, $D_{\text{conv}} = 2$, as in Thm. 3.2) is **equivalent**, in term of associative recall (AR or MQAR tasks), to a similar single-head model of dimensions $D' = \frac{D}{M}$, $N' = N$. Specifically, the model probability of success satisfies:*

$$p_{\text{MKA}}[D, N, M] = p_{\text{single}}[\tfrac{D}{M}, N] \leq p_{\text{single}}[D, N] \tag{175}$$

(Namely, for MKA pattern, splitting into heads might hurt recall performance, despite using a larger key dimension MN.)

Proof. We repeat the proof of Lem. H.3, but now $B_\tau^h = F^h E x_{\tau-1}$ is head-specific, while $C_t = F E x_t$ is shared across heads. The same constraint (forcing $F^h = F$) applies here in order to achieve recall. Hence, the quantitative analysis is the same. □

Lemma H.5 (MHA recall circuit). *A single-layer simplified Mamba-2 model with M heads using MHA pattern and dimensions D, N (with $\text{expand} = 2$, $D_{\text{conv}} = 2$, as in Thm. 3.2) can solve associative recall tasks (AR or MQAR) with a probability of success that satisfies:*

$$p_{\text{single}}[D, N] \leq p_{\text{MHA}}[D, N, M] \leq p_{\text{single}}[D, MN] \tag{176}$$

(Namely, for MHA pattern, splitting into heads improves recall performance, due to the larger key and query dimensions MN.)

Proof. We prove the lemma using two alternative explanations: effective state size, and statistical error analysis.

Effective state size. Let us carefully examine the model output computation. For simplicity, let us rely on the weights from Lem. H.2. We choose head-dependent key and query weights, $S_B^h = S_C^h = F^h$, where $F^h \in \mathbb{R}^{N \times D}$. Each head output is then

$$\hat{y}_t^h = \sum_{\tau=0}^{t} \hat{x}_\tau \, (F^h \, \hat{x}_{\tau-1})^\top (F^h \, \hat{x}_t) \,, \tag{177}$$

with $\hat{x}_t \equiv E \, x_t$. The overall SSM output is, hence, aggregation of $\{\hat{y}_t^h\}$:

$$\hat{y}_t = \sum_{h=1}^{M} W_y^h \sum_{\tau=0}^{t} \hat{x}_\tau \, (F^h \, \hat{x}_{\tau-1})^\top (F^h \, \hat{x}_t), \tag{178}$$

which is, since W_y^h and F^h are all independent, equivalent to:

$$\hat{y} = \sum_{\tau=0}^{t} \hat{x}_\tau \, (F' \, \hat{x}_{\tau-1})^\top (F' \, \hat{x}_t), \tag{179}$$

where $F' \in \mathbb{R}^{(MN) \times D} \equiv \mathbb{R}^{N' \times D}$. This implies that effectively, MHA patterns utilizes a larger state size $N' = MN$.

Statistical error analysis. Alternatively, let us quantify how the additional heads shrink the recall error. We have shown in Theorem 4.2 that the *correct*, *wrong* and *empty* entries of the output vector y_t, namely i, j, l respectively, are distributed as (see Eq. 98, Eq. 99 and Thm. F.4):

$$\begin{aligned} y_t^i &\sim \mathcal{N}(1, \; \sigma_d^2 + \tilde{\sigma}_d^2 + N_f \, \sigma_o^2 \, \tilde{\sigma}_o^2) \\ y_t^j &\sim \mathcal{N}(0, \; \sigma_o^2 + \tilde{\sigma}_o^2 + N_f \, \sigma_o^2 \, \tilde{\sigma}_o^2) \\ y_t^l &\sim \mathcal{N}(0, \; \sigma_o^2 + N_f \, \sigma_o^2 \, \tilde{\sigma}_o^2) \end{aligned} \tag{180}$$

The σ_o^2, σ_d^2 terms originate from embedding compression noise, via $\hat{x}_i^\top \hat{x}_j = x_i^\top E^\top E\, x_j \sim \mathcal{N}(\delta_{ij}, \frac{c}{D})$, with c a constant. This variance terms σ_o^2, $\sigma_d^2 = O(\frac{1}{D})$ cannot be reduced using additional heads, since they arise from input and output embedding E compression and decompression. However, the splitting into multiple SSM heads does reduce the variance. The $\tilde{\sigma}_o^2$, $\tilde{\sigma}_d^2$ terms arise from SSM compression noise, originally via $(F^h \hat{x}_i)^\top (F^h \hat{x}_j) = x_i^\top E^\top F^{h\top} F^h\, E\, x_j \sim \mathcal{N}(\delta_{ij}, \frac{c}{N})$. Now since we have M independent noise terms, each with variance $(\tilde{\sigma}^h)^2 = O(\frac{1}{N})$, their weighted average have a smaller variance, reduced by a factor of M, namely $\tilde{\sigma}^2 = O(\frac{1}{MN})$. Therefore, we can effectively think of this model as of a single-head model with state size $N' = MN$.

Probability bounds. The analysis above shows that there exist weights for which the multi-head model solves associative recall with success probability $p_{\text{single}}[D, MN]$. However, the model is not guaranteed to converge to this particular solution under SGD (or any other optimization method), so this quantity should be viewed only as an upper bound:

$$p_{\text{MHA}}[D, N, M] \leq p_{\text{single}}[D, MN] \tag{181}$$

Conversely, if all heads share the same weights, i.e., $F^h = F \in \mathbb{R}^{N \times D}$ for all h, then the model redundantly implements the MVA solution from Lem. H.2. This yields a lower bound:

$$p_{\text{MHA}}[D, N, M] \geq p_{\text{single}}[D, N]. \tag{182}$$

□

H.3 Notation and Dimensions

We present here the notation and dimensions to be used throughout the analysis of associative recall in Mamba-2. Several differences from Mamba (see Table 3) are worthy of explanation. Firstly, note that we begin already with a simplified model, hence omit state update matrix A, which is assumed an identity operation. Secondly, mind the different pattern of projections and convolutions used in Mamba-2 to produce SSM sequences $\hat{x}$, B and C: no expansion is used for input projection (thus D_{in} is not used), but rather three different projections and convolution kernels, one per $\hat{x}, B, C$. (In implementation, these are usually fused into larger, unified projection and kernel; here we separate notation for clarity.)

Table 6: Notation for Mamba-2

Symbol	Meaning	Value	Notes
D	Embedding size	= Number of SSM channels	
N	SSM State size	-	
D_{conv}	Convolution kernel size	-	
Λ	Number of layers	-	
M	Number of SSM heads	-	
P	SSM head size	$\frac{D}{M}$	
p	Head pattern	MHA, MKA, MQA or MVA	

Table 7: Dimensions for Mamba-2

Symbol	Meaning	Dimension	Notes
x	Model input sequence (one-hot)	$\mathbb{R}^{V\times L}$	$x_t \in \mathbb{R}^V$
y	Model output sequence (one-hot)	$\mathbb{R}^{V\times L}$	$y_t \in \mathbb{R}^V$
x^e	Model input sequence (embedded)	$\mathbb{R}^{D\times L}$	$x^e_t \in \mathbb{R}^D$
$\hat{x}$	SSM input sequence	$\mathbb{R}^{D\times L}$	$\hat{x}_t \in \mathbb{R}^D$
$\hat{y}$	SSM output sequence	$\mathbb{R}^{D\times L}$	$\hat{y}_t \in \mathbb{R}^D$
E	Model embedding	$\mathbb{R}^{D\times V}$	
$P_{\text{in}}^{\hat{x}}$	Input projection for $\hat{x}$	$\mathbb{R}^{D\times D}$	
$P_{\text{in}}^{B,C}$	Input projection for B, C	$\mathbb{R}^{D\times D}$	
$W_{\text{conv}}^{\hat{x}}$	Convolution kernel for $\hat{x}$	$\mathbb{R}^{D\times D_{\text{conv}}}$	
$W_{\text{conv}}^{B,C}$	Convolution kernel for B, C	$\mathbb{R}^{D\times D_{\text{conv}}}$	
S_B^h	B^h projection	$\mathbb{R}^{N\times D}$	Per head h; shared if $p \in \{\text{MQA}, \text{MVA}\}$
S_C^h	C^h projection	$\mathbb{R}^{N\times D}$	Per head h; shared if $p \in \{\text{MKA}, \text{MVA}\}$
B^h	SSM input coefficient	$\mathbb{R}^{N\times L}$	Per head h; shared if $p \in \{\text{MQA}, \text{MVA}\}$
C^h	SSM readout coefficient	$\mathbb{R}^{N\times L}$	Per head h; shared if $p \in \{\text{MKA}, \text{MVA}\}$
P_{out}	Output projection	$\mathbb{R}^{D\times D}$	
W_x	Head contraction weights	p-dependent	Used in $\text{contract}(\cdot, W_x)$
W_y	Head aggregation weights	p-dependent	Used in $\text{aggregate}(\cdot, W_y)$

I Ablation Studies and Generalizations

I.1 Recall Circuit Minimality Ablation

Circuit minimality experiments. To find the critical components of Mamba's recall circuit, we ablate the components of a single-layer Mamba model in the MQAR task. Each row in the table removes an additional component from the Mamba block. The *MQAR Accuracy* column shows the performance of the model, averaged after training over 5 different seeds. Also see Fig. 16. Besides the SSM itself, we find that the convolution block is the most critical component of Mamba's recall circuit. Notably, removing the convolution (E) does not collapse accuracy to zero but to $\approx 1/N_f$: without the conv the model cannot bind keys to values, yet its state still retains the bag of in-context values, and guessing among the $N_f = 16$ of them scores $1/N_f \approx 0.06$.

Table 8: **Circuit minimality experiment results.** See Fig. 16.

Model ID	Description	MQAR Accuracy
Base	1-Layer Mamba model w/o layer normalization	1.00 ± 0.00
A	Base model with $\bar{A}_t = I$	1.00 ± 0.00
B	Model A w/o gate	1.00 ± 0.00
C	Model B w/o activation function (post-Conv1D)	1.00 ± 0.00
D	Model C with $d_{\text{conv}} = 2$	1.00 ± 0.00
E	Model C with $d_{\text{conv}} = 1$ (effectively no Conv1D)	0.06 ± 0.00

Implementation details. We use a vocabulary size of $V = 128$; each sequence is $L = 64$ tokens long and contains $N_f = 16$ key-value pairs. The model has a single layer with $D = 64$ channels and a state dimension of $N = 16$. Training follows the setup of App. K, with a $100/1400/2500$-step warmup/flat/decay schedule and at most 4000 steps, early stopping at accuracy 0.999, weight decay 0.1, and gradient clipping 1.5. Each model is trained with 5 different seeds, and we report the mean and standard deviation. In models Base, A, B, and C, $d_{\text{conv}} = 4$, as in the default Mamba block. In model D, we use $d_{\text{conv}} = 2$, which is the paper default (the minimal value that enables recall). In model E, $d_{\text{conv}} = 1$: a size-one kernel cannot mix a token with its predecessor, effectively removing the Conv1D.

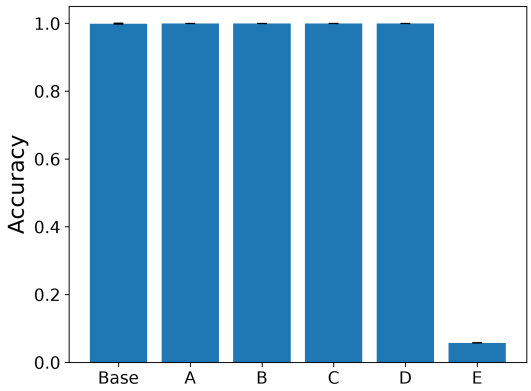


Figure 16: **Circuit minimality experiment results.** The MQAR accuracy of Table 8's ablated models.

I.2 MQAR with Repeated Keys

Repeated Keys. We study a generalization of MQAR where each key appears N_k times, each time paired with a different value, and the model must retrieve the value of the key's last occurrence, as in the following highlighted example ($N_k = 2$):

x	A	2	B	9	C	4	A	6	B	3	C	7	B	2	5	0	9	C	4	A	8	1
y	*	*	*	*	*	*	*	*	*	*	*	*	3	*	*	*	*	7	*	6	*	*

Experiment results. Results are summarized in Fig. 17. We see that the single-layer model performs no better than a mere guess among the N_k stored values: its accuracy is approximately $1/N_k$. We conclude that every occurrence writes its value into the same hidden-state slot, which ends up holding a roughly equal mix of all N_k values instead of just the last one. Depth resolves this: with two layers, the first layer can read back the currently stored value, and the second layer writes in the difference between the new value and the old one, overwriting the stale entry instead of blending with it.

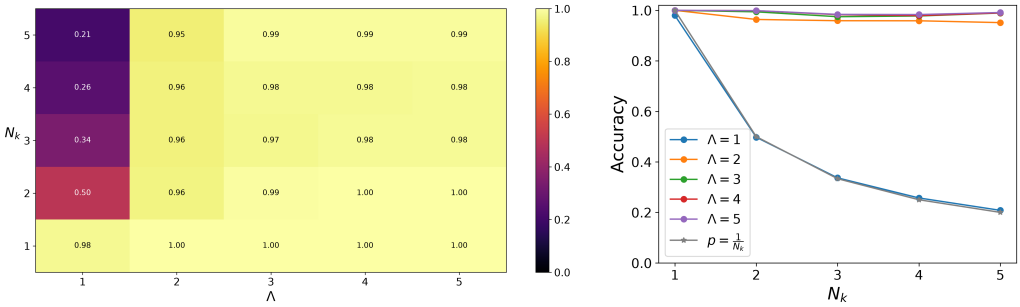


Figure 17: **MQAR with repeated keys.** Recall accuracy of trained linear models. In this experiment $V = 512$, $L = 128$, $N_f = 10$, $D = 100$, and $N_{\text{eff}} = \Lambda N = 60$ fixed (see Thm. 4.6), in order to keep total state size constant. **Left:** accuracy over the Λ-N_k grid. **Right:** accuracy against N_k per depth Λ, with a $1/N_k$ guessing baseline (gray).

J Similarities and Differences with *Understanding Input Selectivity in Mamba*

This section provides a methodological and technical comparison between our work and Huang et al. [30].

Starting with a bird-eye view, both works focus on Mamba model variants and their ability to perform associative recall. In addition, both works offer a detailed theoretical perspective, supported by experiments. Our study, however, proves *tighter bounds* for the dimensions required for the Mamba recall circuit. These tighter bounds enable both *mechanistic investigation* and theoretical derivation of highly-predictive AR *scaling laws* for Mamba, which are *empirically verified* - capabilities not possible with the bounds established by Huang et al. [30].

The main points of difference lie in both contribution and methodology:

1. We theoretically show that Mamba can learn a parameter-efficient solution for MQAR, i.e. a solution which requires feasible model dimensions, satisfying $ND = O(\log V)$ given a

fixed number of facts N_f. These are **tighter bounds** compared to the bounds proved by Huang et al. [30] for Mamba, which are $D = O(V_k + \log V_v)$ and $N = V_k$.

2. Our suggested recall circuit is validated by **mechanistic interpretability** experiments, based on trained model weights and activations. In other words, our interest lies in the mechanism the model *truly learns* during training, as opposed to those it *could, in principle, learn* based solely on its architectural capacity.
3. Our contribution includes the theoretical derivation of **scaling laws** for recall in Mamba, as well as empirical validation. The exact scaling laws (Theorem 4.2 and Remark 4.4), which align with experimental result, cannot be derived from Huang et al. [30] Theorem 1, since it is based on a different circuit.
4. In addition to single-layer models, our study generalizes to **multi-layer** Mamba models, aiming to contribute a further understanding of multi-layer recall circuits (Theorem 4.6). Our conclusions are validated through empirical examination in Section 5.3.
5. While Huang et al. [30] focus on single-head SSM mixers for Mamba-2, our work further contributes the study of **multi-head** recall circuits, as enabled by the Mamba-2 architecture (Theorem 4.7). The theoretical insights are further verified by experimental analysis in Section 5.4.
6. By methodology, we seek a **minimal recall circuit**, in order to isolate factors and find the most significant component for recall (see Appendix I.1). Upon removal of model components, we identify linear compression limits as the main factor, represented by the model dimensions D and N. Neither gating nor nonlinearities drastically overcome this inherent bottleneck (as shown in Figure 1).

Let us now carefully dive into a technical comparison between the works.
(Note that by *Mamba*, we specifically refer to *"Mamba-1"* architecture, as opposed to *Mamba-2*.)

Non-compressive recall circuits. Lemma 3 in Huang et al. [30] proves that a 1-layer *Mamba*-$\Delta^\top$ model can solve the Induction Heads task, if model dimensions are $D, N = O(V)$. Theorem 3.1 in our work leverages a similar construction for a 1-layer *Mamba*, showing it can solve MQAR, given $D, N = O(V)$. Importantly, however, this Theorem is not the key novelty of our work, but rather serves as an intermediate logical step towards Theorem 3.2, which addresses a realistic scenario where $ND = O(\log V) < O(V^2)$ (given a fixed number of facts N_f), thus applicable to actual MQAR tasks.

Compressive recall circuits. Theorem 2 in Huang et al. [30] shows that a 1-layer *Mamba* with dimensions $D = O(V_k + \log V_v)$ and $N = V_k$ can solve MQAR, by applying JL lemma to the values (but not to the keys, which remain uncompressed).
Firstly, Theorem 3.2 in our work improves the **efficiency** of the solution, by showing that a 1-layer *Mamba* model can solve MQAR with smaller, realistic dimensions, satisfying $ND = O(\log V)$ given a fixed number of facts N_f, by using a different construction, which allows us to apply JL lemma *twice*, to both values and keys. (Technically, we show *Mamba* can compress its inputs twice: first compression via embedding into $Ex_t \in \mathbb{R}^D$, used for values; then a second compression via S_B, S_C into $FEx_t \in \mathbb{R}^N$, used for keys and queries.) This difference is of major importance for realistic Mamba models and MQAR tasks, where $N = V_k$ and $D = O(V_k)$ are impractical and cannot be used due to memory limitation. A parameter-efficient solution is critical in practice; see, for instance, Figure 6, row (iii), where we empirically show that a 1-layer Mamba can perfectly solve an MQAR task of $V_k = \frac{V}{2} = 512$ given relatively small dimensions $N = 32 << V_k$ and $D = 64 << V_k$.
Secondly and most importantly, our **mechanistic analysis** in Section 3.2 (also see Figure 3) highlights the difference between the two constructions. The solution suggested by Huang et al. [30] implies that *each (hidden state) column corresponds to a specific key*: each embedded value Ev_τ is stored in a **single**, distinct key coordinate of the hidden state h_t (equivalent to H'_t in Section 3.2), out of $N = V_k$ coordinates in total. In contrast, our parameter-efficient construction suggests each embedded value Ev_τ is stored in a **distributed** manner, across a linear mixture of **all** $N \ll V_k$ key coordinates. The mechanistic experiments we conduct in Section 3.2 support our suggested circuit. It shows that *Mamba* actually learns a solution where writing a single key-value pair into H'_t affects **all** key coordinates, leading to the pattern observed for H'_t, where all entries are nonzero (Figure 3). If, conversely, the learned mechanism would have used a **single** key entry per value Ev_τ, one could have inspected the hidden state by decompressing value coordinates **only**, via $H''_t = E^\top H'_t$, since key coordinates would not interfere with one another. However, we experimentally show that a **double**-decompression is rather required, $H''_t = E^\top H'_t \tilde{E}$ (see Equation 14 and Figure 3), supporting our theory.

Thirdly, our theoretical and empirical **scaling laws** (Theorem 4.2 and Figure 1 respectively) indicate that *Mamba* recall capacity strongly depends on both dimensions N and D. The exact form of this dependence, where performance improves upon increasing $\frac{ND}{N_f}$, arises due to *noisy retrieval* of values out of the hidden state H'_t. Importantly, while the $\frac{1}{D}$ factor is caused by value reconstruction noise $v''_i = E^\top E v_i + O(\frac{1}{D})$, the $\frac{N_f}{N}$ factor arises from noisy query-key matching $\sum_p^{N_f} (\tilde{E}k_p)^\top (\tilde{E}k_m) = O(\frac{N_f}{N})$. This is only the case if values are stored as $H'_t = \sum_p^{N_f} (Ev_p)(\tilde{E}k_p)^\top$, where keys are compressed into $\tilde{E}k_p \in \mathbb{R}^N$. Without relying on such a *double-compressive* circuit (as in Theorem 3.2, as opposed to Theorem 1 in Huang et al. [30]), the derived scaling law would lack this important $\frac{N_f}{N}$ factor, which is clearly demonstrated in practice (Figure 1).

Minimal recall circuits. Our work focuses on finding a **minimal** Mamba model that can solve MQAR, in order to isolate the contributing factors. In our perspective, this approach is important in order to understand potential **bottlenecks** in Mamba architecture. Since Mamba have a variety of components (including discretization, gating, nonlinearities, normalization layers, input selectivity, etc.), there may *exist* many possible constructions combining them that can *possibly* enable Mamba to solve MQAR. Minimality is required if we aim to find a critical component.
Firstly, our ablation study (Appendix I.1; see table entry B) clearly shows that a Mamba model without gating is still able to achieve perfect accuracy on MQAR. This hints that, while probably participating in the recall process, the gate-based mechanism is empirically not a bottleneck for Mamba ability to perform recall.
This insight is further reinforced by the results presented in Figure 1. The conducted experiment compares between our gate-free simplified model and the full gated model, both trained on MQAR. Our findings show the models perform quite similarly on MQAR. This demonstrates that the main bottleneck in Mamba recall capacity is the model dimensions D and N (responsible for reliable compression-decompression schemes of keys and values), rather than the existence of a gate.

K Training and Implementation Details

This appendix provides all training and implementation details needed to reproduce our results. Also see *reproducibility statement*, App. L.2.

K.1 Datasets

We use the original MQAR dataset as implemented by Arora et al. [9], vendored with minor adaptations in the released code. To avoid generalization error and overfit phenomena, instead of generating large training datasets statically, we sample new batches from MQAR in any training step.

K.2 Models

As for the full Mamba model, we use `mamba-ssm` [1], unmodified. Our simplified linear Mamba model is implemented based on `mamba-tiny` [52], vendored with local modifications. We use PyTorch [53] for all experiments. Asset licenses are listed in App. L.3.

K.3 Training Hyperparameters

Unless stated otherwise, we use the hyperparameters in Table 9.

K.4 Compute and Environment

All experiments were run on 4×GPU and 8×GPU nodes with Nvidia B200 GPUs, with CUDA 12.8 and PyTorch 2.10.0, using multiprocessing of several training workers per GPU. Each grid was run on a single GPU (the multi-layer AR grid on two), and independent grids ran in parallel within a node. We fix `torch`, `mamba-ssm`, and `python` versions in the released setup script, and all remaining direct dependencies via a `constraints.txt`.
Each accuracy-grid figure or sub-figure (e.g. Fig. 1, 4, 5) corresponds to roughly 500-1000 grid points (each trained with 3-5 seeds), each individually small. Simplified linear model grids take 8.0 ± 5.5 hours wall-clock time per grid (single GPU); full Mamba model grids take 2.3 ± 1.0 days wall-clock time per grid (see Table 9). Across all experiments in the paper (single-layer, multi-layer, multi-head, and ablation studies), the total compute amounts to roughly 10 GPU-days.

Table 9: Training hyperparameters per experiment family.

Model Task	Linear MQAR	Linear AR	Full MQAR	Multi-layer MQAR	Multi-layer AR	Multi-head MQAR	Notes
Batch size	128	128	128	128	750	128	
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$						
Base learning rate	0.01						
Weight decay	0.0						
Label smoothing	0.1						
Total steps	2000	2000	20000	4000	10000	4000	
Warmup steps	100	100	100	100	250	100	0 to base LR
Flat-LR steps	400	400	5900	1400	2250	1400	at base LR
Decay steps	1500	1500	14000	2500	7500	2500	cosine decay to 0
Early stop at accuracy	1.0	1.0	0.999	1.0	1.0	1.0	
Gradient clipping	1.5	1.5	0.75	2.5	2.5	5	(global norm)
Seeds per grid point	3	3	5	3	3	3	
Per-grid GPU time	4.6 h	11.4 h	54.9 h	2.3 h	8.7 h	1.6 h	mean across grids

K.5 Seeds Aggregation

We briefly describe our considerations choosing best-of-seeds accuracy. In this work we investigate Mamba's recall *capacity*, therefore retaining the maximal result achieved by each configuration is the most relevant quantity for our analysis. In other words, we are interested in the *existence* of a recall solution, not the *probability* of achieving it by optimization. Therefore, it makes more sense to take into account only the trials where optimization succeeds. To demonstrate the effect of seed aggregation, in Figs. 18 and 19 we further provide a detailed analysis of a single full-model grid: the per-seed accuracy grids against the best-of-seeds grid, and the per-cell convergence rates near the capacity boundary.

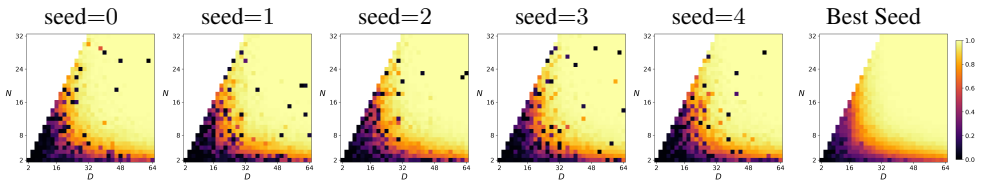


Figure 18: **Per-seed accuracy grids.** Accuracy of the full model over the D-N grid of regime (iii) of Fig. 6, shown separately for each of the 5 seeds, next to the best-of-seeds grid used throughout the paper. The recall phase boundary is consistent across seeds; seeds differ mainly in isolated optimization failures (analyzed in Fig. 19).

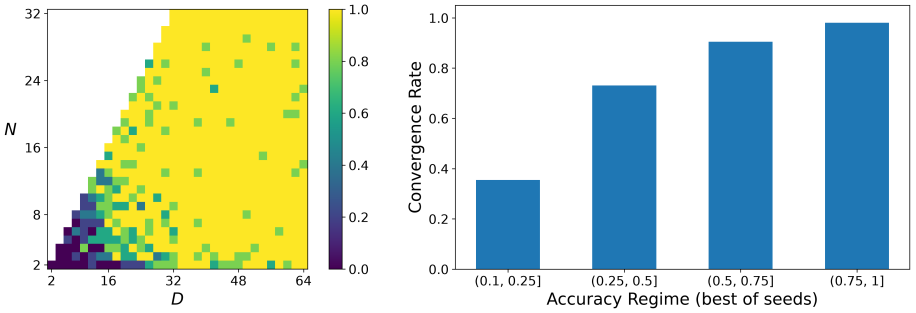


Figure 19: **Seed aggregation analysis.** The full model over the D-N grid of regime (iii) of Fig. 6, trained with 5 seeds per grid point; a seed is counted as converged if it reaches accuracy ≥ 0.1. **Left:** per-cell fraction of converged seeds. **Right:** per-seed convergence rate, binned by the cell's best-of-seeds accuracy.

L Statements and Declarations

L.1 Impact Statement

This work analyzes the recall capabilities of Mamba models and linear RNNs, which are crucial for deploying recurrent LLMs in real-world systems. Improving memory efficiency and retrieval understanding can lead to more reliable and controllable architectures, with broad benefits for AI applications. We do not anticipate any direct negative societal impacts beyond those generally associated with LLMs.

L.2 Reproducibility Statement

We provide the source code for the main experiments and include detailed descriptions of the configurations in the publicly released repository, linked on the first page. All datasets, models and metrics used in each experiment are explicitly stated in the paper to ensure full transparency and reproducibility. See also implementation details in App. K.

L.3 Licenses

Table 10: Open-source assets used in this work and their licenses.

Asset	Reference	License	URL
`mamba-ssm`	Gu and Dao [1]	Apache 2.0	`github.com/state-spaces/mamba`
Zoology / MQAR	Arora et al. [9]	Apache 2.0	`github.com/HazyResearch/zoology`
`mamba-tiny`	PeaBrane [52]	Apache 2.0	`github.com/PeaBrane/mamba-tiny`
PyTorch	Paszke et al. [53]	BSD 3-Clause	`github.com/pytorch/pytorch`

L.4 The Use of Large Language Models (LLMs)

ChatGPT [54] and Claude [55] were used as general-purpose writing and editing assistants in the preparation of this manuscript. Their role was limited to helping polish the presentation: rephrasing sentences for clarity, improving grammar and flow, suggesting alternative wording, and condensing or expanding explanations when asked. Importantly, they were not involved in the research ideation, design, implementation, analysis, or interpretation of results. Additionally, ChatGPT and Claude were used for coding assistance.

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract lists three concrete contributions: identifying the recall circuit via mechanistic interpretability, deriving recall scaling laws from the Johnson-Lindenstrauss lemma (given the vocabulary size and number of facts), and extending the analysis to multi-layer and multi-head models. All three are delivered by the theorems in Secs. 3 and 5 and the experiments in Sec. 5, with the key simplifying assumption (linear Mamba model) stated clearly.
   Guidelines:
   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: A dedicated Limitations paragraph appears at the end of Section 6. It notes that the analysis relies on a simplified linear Mamba model, that gating, nonlinearities, and discretization enter the bounds only through constants, and that direct transfer of the scaling laws from synthetic tasks to language modeling remains to be established.
   Guidelines:
   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
Answer: [Yes]
Justification: All theorems state their assumptions explicitly in the theorem statements. Full proofs for the recall circuit theorems are in App. C and full proofs for the scaling law theorems are in App. F. Proof sketches providing intuition are included in the main paper for each result.
Guidelines:

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

4. **Experimental result reproducibility**
Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
Answer: [Yes]
Justification: All hyperparameters, dataset details, and model configurations are given in App. K. The MQAR dataset is publicly available via the Zoology repository [9]. The code is publicly released in the repository linked on the first page; see the Reproducibility Statement in App. L.2.
Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
   Answer: [Yes]
   Justification: The code is publicly released in the repository linked on the first page. The MQAR dataset used for all experiments is publicly available via the Zoology repository [9]. See App. L.2.
   Guidelines:
   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?
   Answer: [Yes]
   Justification: All training hyperparameters are listed in Table 9 in App. K: AdamW optimizer, learning rate 10^{-2}, batch size 128, gradient clipping, and label smoothing. Data is sampled online from MQAR at each training step (no fixed split). The architecture dimensions varied per experiment are stated in each figure and its caption.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [No]
   Justification: The main scaling-law figures report best-of-seeds accuracy across multiple random seeds, because the goal is to measure model capacity (whether a recall solution exists) rather than optimization reliability; the rationale is explained in Sec. 5 and App. K.5. The ablation study in Table 8 does report mean $\pm$ std over 3 seeds.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [Yes]
   Justification: App. K reports the hardware (Nvidia B200 GPUs, CUDA 12.8, PyTorch 2.10.0), per-grid wall-clock times (8.0 ± 5.5 hours for linear model grids, 2.3 ± 1.0 days for full model grids), and an estimate of total compute across all experiments (roughly 10 GPU-days).
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The paper involves purely theoretical and empirical analysis on synthetic benchmarks. There are no human subjects, no personal data collection, and no harmful applications. The research conforms to the NeurIPS Code of Ethics in every respect.
   Guidelines:
   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [Yes]
    Justification: An Impact Statement is included in the Statements and Declarations appendix (App. L). The work improves understanding of memory mechanisms in recurrent language models, with positive implications for building more efficient and interpretable AI systems. No direct negative societal impacts beyond those generally associated with language models are anticipated.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release any pretrained models, scraped datasets, or other assets with significant potential for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All assets used in this work are cited and their licenses are listed in Table 10 in App. L.3: `mamba-ssm` [1] and the Zoology MQAR codebase [9] (both Apache 2.0), `mamba-tiny` [52] (Apache 2.0), and PyTorch [53] (BSD 3-Clause).

    Guidelines:

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

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

13. **New assets**
    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
    Answer: [Yes]
    Justification: The released code is documented alongside the asset: the repository linked on the first page (MIT license) includes a README with setup, quickstart, and figure-reproduction instructions, pinned dependencies, the experiment configurations and saved run data, a test suite, and a citation file (App. L.2). No new datasets or model checkpoints are released.
    Guidelines:
    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**
    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
    Answer: [N/A]
    Justification: The paper does not involve any crowdsourcing experiments or research with human subjects.
    Guidelines:
    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
    Answer: [N/A]
    Justification: The paper does not involve research with human subjects, so no IRB approval was required or obtained.
    Guidelines:
    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are studied but not used as core methodology components. ChatGPT and Claude were used for writing, editing, and coding only; see App. L.

    Guidelines:

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