How does Mamba Perform Associative Recall? A Mechanistic Study

Gregoire Le Corre
ENS Ulm

Ningyuan (Teresa) Huang
Flatiron Institute

Alberto Bietti
Flatiron Institute

Abstract

Mamba has recently emerged as a promising alternative to Transformers, demonstrating competitive performance in many language modeling tasks with linear-time computational complexity. Theoretical characterization of Mamba has largely focused on its approximation power for solving certain tasks through specific constructions. However, it remains unclear whether Mamba trained with gradient descent can learn such constructions. As a first step to address this gap, we perform a mechanistic study of simplified Mamba models on associative recall tasks. By analyzing the learned model weights and the hidden state evolution, we uncover the mechanisms used by simplified Mamba models to perform associative recall. We complement our study with theoretical analysis on the optimization dynamics of simplified Mamba models that give rise to such mechanisms.

1 Introduction

Recently, structured state space models (SSMs) have arisen as competitive sequence modeling architectures [7, 9, 8]. In particular, Mamba [6] has emerged as a promising general-purpose sequence model, demonstrating competitive performance on various language modeling tasks while reducing the quadratic complexity in Transformers [13] to linear time. Theoretical understanding of Mamba begins to emerge, centering on its *expressivity* using approaches such as formal language theory [11, 5] and approximation theory [3, 10].

Focusing on Mamba's associative recall capabilities, [10] showed that there exists a Mamba model with well-chosen weights that can solve certain associative recall tasks. However, it remains unclear whether training Mamba with gradient descent will find such a solution, or whether a different mechanism is found in practice. To fill this gap, we conduct a mechanistic study to probe how a simplified version of Mamba can learn to perform associative recall. Our main contributions can be summarized as follows:

- We confirm that in a simplified setup, Mamba can learn the theorized solution for solving associative recall tasks proposed in [10], by providing evidences from the learned model weights as well as the evolution of hidden state and outputs (Sec. 3).
- We support our empirical findings with a theoretical analysis on the optimization dynamics of (simplified) Mamba (Sec. 4).

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

Input: Alice used to live in Paris, then she moved to New York, and nowadays Alice lives in Barcelona.

Query: Where does Alice live?

Answer: Barcelona

Figure 1: Associative recall example to retrive the *latest* association.

$$\begin{array}{cc} \mathbf{x} & = & k_1 & v_1 & xx & k_2 & v_2 & xx & k_1 & v_3 \dots & k_2 \dots k_1 \\ \mathbf{y} & & & & & & & & v_1 & & v_1 \dots v_3 \end{array}$$

Figure 2: MQAR example, interleaving $k_i v_i$ key-value pairs with noise token x; upon query (seen keys) return the *latest* associated value.

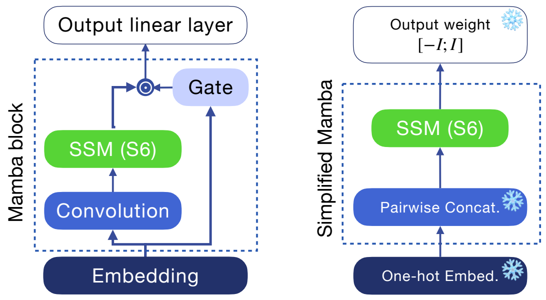


Figure 3: The original Mamba architecture (left), and our simplification (right) which disables the gating branches, and fixes the embedding, convolution, and output layers.

2 Problem Setup

2.1 The MQAR task

To evaluate the associative recall capabilities of Mamba, we adapt the *Multiple-Query Associative Recall* (MQAR [1, 10]) task and consider a more difficult variant, illustrated in Fig. 2 with full details described in App. A. In particular, a key $k_i \in K$ can be associated with many different values $v_{i_1}, \dots, v_{i_m}$ in the context. Upon querying the key k_i, the *latest* associated value v_{i_m} must be returned. This probes the ability of Mamba to store the key-value associations and recall the most recent associated value. This recency bias is particularly well-suited to the recurrent, state-based nature of Mamba, as opposed to attention, which would need additional positional information to distinguish the latest occurrence of the key from previous ones. This MQAR variant can be also seen as a simple version of the *Induction Heads* task [12, 10], where the key tokens and value tokens are drawn from disjoint vocabulary sets.

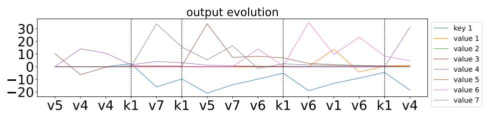
Our design of MQAR is motivated from sturctures in natural language (e.g.,Fig. 1), where language understanding requies recalling the appropriate reference of a pronoun or entity among many distractors. Such pattern is also common in algorithmic tasks where selective recall is a core primitive, as well as time-series analysis where relevant events must be retrieved despite background fluctuations.

2.2 Model Architectures

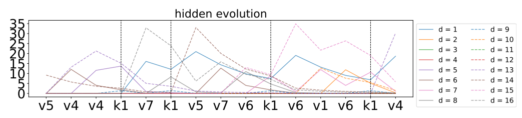
The main architecture of Mamba is illustrated in Fig. 3 (left). To probe the inner-workings of Mamba—inspired by [10, Theorem 2, Lemma 3], which provides explicit constructions of 1-layer Mamba solving MQAR and its variants—we simplify Mamba with justifications given as follows (see Fig. 3 right for an overview):

- **Embedding**: Use one-hot embedding with a doubling trick following the construction of [10, Lemma 3]. Concretely, $v \mapsto \begin{bmatrix}\mathbf{e}_v & \mathbf{e}_v\end{bmatrix}^\top \in \mathbb{R}^{2|V|}$ for token v drawn from the vocabulary V.
- **Convolution**: Use a fixed depthwise convolution with kernel size 2 and left padding, following the construction of [10, Lemma 3]. Concretely, given a sequence of token embeddings, the convolved token at time t is given by

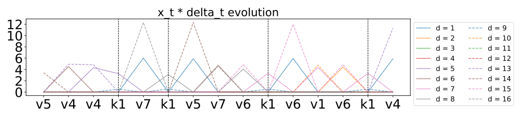
$$\hat{\mathbf{x}}_t \equiv \mathrm{conv}(\mathbf{x}_{t-1}, \mathbf{x}_t) = \mathbf{c}_0 \odot \mathbf{x}_{t-1} + \mathbf{c}_1 \odot \mathbf{x}_t, \qquad \text{where } \mathbf{c}_0 = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix}, \mathbf{c}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \quad (1)$$



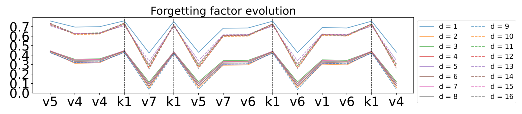
(a) Evolution of the output $\mathbf{y}_t W_o$



(b) Evolution of the hidden state $\mathbf{h}_t$



(c) Evolution of the input term $\hat{\mathbf{x}}_t \odot \Delta_t$



(d) Evolution of the forgetting factor $e^{\mathbf{\Lambda} \odot \Delta_t}$

Figure 4: Probing Mamba on MQAR with $|K| = 1$: (a) each line represents the correponding dimension $d = 0, \ldots, |V| - 1$ of the output $\mathbf{y}_t W_o \in \mathbb{R}^{|V|}$; (b)-(d) each line represents the dimension $d = 0, \ldots, 2|V| - 1$ of the hidden state, the input term, and the forgetting factor, respectively (dashed lines denote the first half $d = 0, \ldots |V| - 1$ while solid lines denote second half $d = |V|, \ldots, 2|V| - 1$).

Observe that such convolution on the $2|V|$-dimensional token embeddings is equivalent to a *concatenation* of two consecutive one-hot token embeddings, i.e.

$$\hat{\mathbf{x}} := \operatorname{conv}\big(\operatorname{embed}([k_1 v_1 x \ldots])\big) = \left[\begin{bmatrix} \mathbf{0} \\ \mathbf{e}_{k_1} \end{bmatrix} \begin{bmatrix} \mathbf{e}_{k_1} \\ \mathbf{e}_{v_1} \end{bmatrix} \begin{bmatrix} \mathbf{e}_{v_1} \\ \mathbf{e}_{x} \end{bmatrix} \ldots \right].$$

This also fixes the hidden model dimension as $d = 2|V|$. In our implementation, we simply use one-hot embedding followed by concatenating pairs of token embeddings.

- **Output unembedding**: Replace the learnable linear map with a fixed weight matrix $W_o = [-I \quad I] \in \mathbb{R}^{|V| \times 2|V|}$, motivated from our observations on trained models (App. B).
- **Gate**: Disable the gating branch, following the construction of [10, Lemma 3]

We preserve the key SSM layer in Mamba, defined as

$$\mathbf{h}_t = \exp(\mathbf{\Lambda} \odot (\Delta_t \otimes \mathbf{1}_N)) \odot \mathbf{h}_{t-1} + (\Delta_t \odot \hat{\mathbf{x}}_t) \otimes B(\hat{\mathbf{x}}_t), \quad \mathbf{y}_t = \mathbf{h}_t C(\hat{\mathbf{x}}_t). \tag{2}$$

where $\mathbf{h}_t, \mathbf{\Lambda} \in \mathbb{R}^{d \times N}, \Delta_t \in \mathbb{R}^d$ and $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t) \in \mathbb{R}^N$ (N is the state size). We set $\Delta_t = \operatorname{Softplus}(W_\Delta \hat{\mathbf{x}}_t)$, where $W_\Delta \in \mathbb{R}^{d \times d}$ is a learnable weight matrix controlling the discretization step.

We also remark that the assumed structure of embedding and convolution layers corresponds (up to rotation) to using embedding and convolution weights fixed at random initialization with infinite width, leading to the desired orthogonality (see [2] for an explanation in the context of Transformers).

3 Probing the Learned Mechanism

Case $|K| = 1$. We begin with the simple case $|K| = 1$ (i.e., one key only) in order to highlight Mamba's mechanism of learning and forgetting. To solve MQAR with $|K| = 1$, it is sufficient to use a state size $N = 1$ [10], reducing $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t)$ in (2) to scalars. Thus, we fix $B = C = 1$ which simplifies the SSM as

$$\mathbf{h}_t = e^{\mathbf{\Lambda} \odot \Delta_t} \odot \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t. \tag{3}$$

We train the simplified Mamba model on our MQAR variant with vocabulary size $|V| = 8$ (with the key token set $K = \{0\}$ and value token set $L = \{1, 2, \ldots, 7\}$), a maximum noise length of $n_\epsilon = 3$ tokens, and sequence length $T = 256$, until it reaches *100% accuracy* on the training set. We then evaluate the model *after training* to identify how Mamba can perfectly solve such task.

We probe the trained Mamba model behavior by plotting the evoluion (per each dimension d) of its *output after unembedding* $\mathbf{y}_t W_o$, *hidden state* $\mathbf{h}_t$, the *input term* $\Delta_t \odot \hat{\mathbf{x}}_t$, and the *forgetting factor* $e^{\mathbf{\Lambda} \odot \Delta_t}$ (acting on the previous hidden state). From the output evolution Fig. 4a, we see a large spike when a new value is being written. From the hidden state evolution Fig. 4b, we see similar large spikes upon encountering new value tokens (solid lines representing the first $d = |V|$ dimensions, and dashed lines representing the last $d = |V|$ dimensions). Fig. 4c corresponds to the input term, that is, what's being newly written. We clearly see a spike when a new value is written in last $d = |V|$

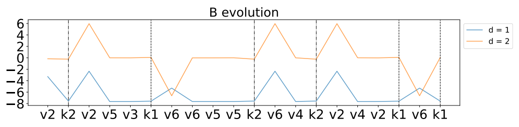
dimensions. Fig. 4d shows the forgetting factor, which seems to erase the hidden state as the new value is written.

Additionally, we observe a clear block structure in the W_Δ matrix, shown in Fig. 8 (see App. C), leading to the discretization step $\Delta_t = \mathrm{Softplus}(W_\Delta \, \mathrm{conv}(\mathbf{x}_{t-1}, \mathbf{x}_t))$ being close to zero unless a key token is present in the input embedding pair. Combining the evolution patterns with the W_Δ structure reveals the following mechanisms in the hidden state based on the input pair at the current position t (here we denote the key token as $k \in K$ and the value (noise) tokens be $v, v_1, v_2, x \in V \setminus K$):

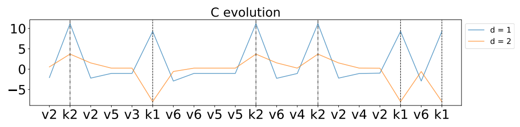
- (write) When $(x_{t-1}, x_t) = (k, v)$, erase (small forgetting factor) and write the new value (input term has large value on the $|V| + v$ coordinate);
- (read) When $(x_{t-1}, x_t) = (x, k)$, retrieve stored value from memory based on the largest difference between the $|V| + v$ and v coordinates, with almost no forgetting.
- (forget) When $(x_{t-1}, x_t) = (v_1, v_2)$, forget all the memories slowly (less forgetting than 'write', but more than 'read', especially on the second half of the coordinates).

We give a more detailed analysis of this mechanism in App. C. We remark that the slowly forgetting mechanism is crucial for Mamba to differentiate correctly the *latest* associated value by discounting the values seen in distant past.

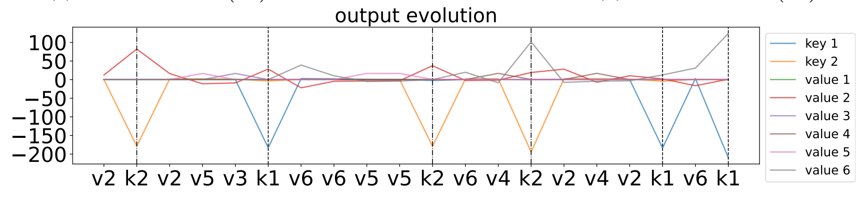
Case $|K| > 1$. In this setting, we use the original $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t) \in \mathbb{R}^N$ vectors without simplification, motivated from [10, Theorem 2] which proposes MQAR solution with a state size $N = |K|$. Figure 5 shows the evolution patterns of $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t)$ and the output for the case $|K| = 2$. We empirically observe the spikes in $B(\hat{\mathbf{x}}_t)$ upon encountering new key-value pairs (i.e. when $(x_{t-1}, x_t) = (k, v)$) and $C(\hat{\mathbf{x}}_t)$ upon query (i.e., when $(x_{t-1}, x_t) = (x, k)$), corresponding to the write and read mechanisms identified in the case $|K| = 1$. Figure 6 shows the evolution patterns of $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t)$ and the output for the case $|K| = 4$, with similar spiking behavior.



(a) Evolution of $B(\hat{\mathbf{x}}_t)$



(b) Evolution of $C(\hat{\mathbf{x}}_t)$



(c) Evolution of output $\mathbf{y}_t W_o$: key tokens are highlighted by vertical dashed lines.

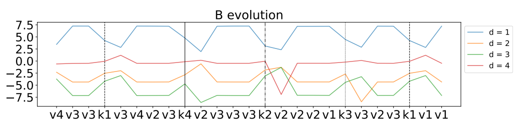
Figure 5: Probing Mamba on MQAR with $|K| = 2$: each line represents the component-wise evolution of the $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t) \in \mathbb{R}^N \equiv \mathbb{R}^{|K|}$, and the output $\mathbf{y}_t W_o \in \mathbb{R}^{|V|}$.

4 Theoretical Analysis

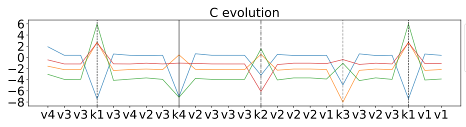
To support our empirical observation, we analyze the optimization dynamics of a simplified Mamba model (see Sec. 2.2) on our MQAR variant with $|K| = 1$. We focus on the parameter W_Δ (used to compute $\Delta_t = W_\Delta \hat{\mathbf{x}}_t$, where we dropped the softplus in this simplified theory setup) and fix the state matrix $\mathbf{\Lambda} = \mathbf{1}$ (i.e., no forgetting). Similar to [2], we consider the model after one gradient step over the population loss L with respect to W_Δ, starting from $W_\Delta = 0$:

$$W_\Delta = -\nabla_{W_\Delta} L = -\mathbb{E}_{(\mathbf{x}_0, \ldots, \mathbf{x}_T) \sim P} \left[\nabla_{W_\Delta} \ell \mid \mathbf{x}_0 = \mathbf{x}_T = 0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1} \neq 0 \right], \tag{4}$$

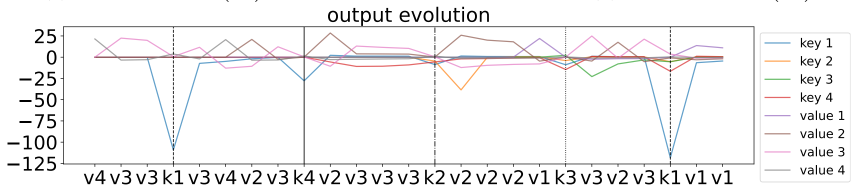
where the expectation is taken over all input sequences having the key token 0 appearing at the first and last positions, and ℓ is the cross-entropy loss for one such sequence. The following lemma (proved in Appendix D) shows that gradient descent can update W_Δ to perform the writing mechanism and solve the task, as W_Δ aligns with the key vector while remains almost orthogonal to non-key vectors.



(a) Evolution of $B(\hat{\mathbf{x}}_t)$



(b) Evolution of $C(\hat{\mathbf{x}}_t)$



(c) Evolution of output $\mathbf{y}_t W_o$: key tokens are highlighted by vertical dashed lines.

Figure 6: Probing Mamba on MQAR with $|K| = 4$: each line represents the component-wise evolution of the $B(\hat{\mathbf{x}}_t), C(\hat{\mathbf{x}}_t) \in \mathbb{R}^N \equiv \mathbb{R}^{|K|}$, and the output $\mathbf{y}_t W_o \in \mathbb{R}^{|V|}$.

Lemma 1. *Consider the simplified Mamba model with only trainable weights W_Δ (i.e., with one-hot embedding, fixed depthwise convolution akin to pairwise concatenation, fixed output layer $[-I; I]$), and no gating). The gradient of the loss with respect to $W_\Delta \in \mathbb{R}^{2|V| \times 2|V|}$ in (4) takes the form:*

$$\frac{\partial L}{\partial W_{\Delta_{i,j}}} = \begin{cases} \frac{-1}{T} & \textit{if } i = 0 \\ \frac{2}{T} & \textit{if } i = |V| \\ \approx \frac{T}{|V|^2} - \frac{1}{|V|} & \textit{if } 0 < i < |V| \\ \approx \frac{-T}{|V|^2} + \frac{1}{|V|} & \textit{if } i > |V| \end{cases} . \tag{5}$$

For $|V| \gg T$, we see that one gradient step mostly updates the weights on the $0, |V|$-th rows in W_Δ, and thus correctly succeeds at performing the read and write operations for associative recall. We remark that in practice the simplified Mamba model also succeeds for $|V| \leq T$ with $\Delta_t = \text{Softplus}(W_\Delta \hat{\mathbf{x}}_t)$, suggesting that analyzing the role of the softplus nonlinearity could extend the theory to more general settings.

5 Conclusion and Future Directions

Towards understanding how Mamba perform associative recall, we perform a mechanistic study to probe trained Mamba models (with suitable simplifications) for solving the MQAR tasks. We identify key associative-recall mechanisms in the hidden state of simplified Mamba models, including writing, retriveing, and forgetting behaviors based on the input. Our insights are supported via empirical results on probing the trained models and theoretical analysis of simplified Mamba's population gradient dynamics.

As a first step, our mechanistic study focuses on a simplified Mamba model; extending the analysis to the original Mamba and Mamba-2 [4] and considering more complicated tasks is a natural next step. Another interesting direction lies in comparing the associative-recall mechanisms in Mamba with other sequence model architectures, such as Transformers and other subquadratic variants.

Acknowledgments and Disclosure of Funding

The authors thank Federico Danieli for motivating this work. The authors also thank the anonymous NeurIPS Mechanistic Intepretability Workshop reviewers and area chair for their constructive feedback.

References

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

[2] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36:1560–1588, 2023.

[3] Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical Foundations of Deep Selective State-Space Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[4] Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *International Conference on Machine Learning*, volume 235, pages 10041–10071, 2024.

[5] Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025.

[6] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*, 2024.

[7] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent Memory with Optimal Polynomial Projections. *Advances in Neural Information Processing Systems*, 33:1474–1487, 2020.

[8] Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022.

[9] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv:2111.00396*, 2022.

[10] Ningyuan Teresa 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. In *Forty-second International Conference on Machine Learning*, 2025.

[11] William Merrill, Jackson Petty, and Ashish Sabharwal. The Illusion of State in State-Space Models. In *International Conference on Machine Learning*, 2024.

[12] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv:2408.14332*, 2024.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

A MQAR Task Details

Dataset Construction. We denote by V the vocabulary size, K the set of key tokens and L the set of value tokens, such that $K \bigcup L = V$ and $K \bigcap L = \emptyset$. We let $k_1, ..., k_i$ be the key tokens. Then the sequence are constructed with the following algorithm Alg. 1.

Algorithm 1 Synthetic MQAR dataset generation

```
Input: $[p_1, ..., p_k]$ the distribution of the keys, maximum noise length $n_\epsilon$, sequence length $T$
 1: Initialize an empty sequence $\mathbf{x} \leftarrow []$.
 2: while The sequence is shorter than $T - |K|$ do
 3:     Sample the noise size $s$ uniformly between $0$ and $n_\epsilon$
 4:     Sample $s$ noise token $x_1, ..., x_s$ in $L$
 5:     Append $x_1, ..., x_s$ to the sequence
 6:     Sample a token $k$ according to the distribution $[p_1, ..., p_k]$ in $K$
 7:     Sample $v$ uniformly in $L$
 8:     Append $k, v$ to the sequence $\mathbf{x}$
 9: end while
10: Append a random permutation of keys to $\mathbf{x}$
11: return The sequence $\mathbf{x}$
```

For example, with the maximum noise length $n_\epsilon = 3$, and vocabulary $V = \{A, B, X, Y, Z\}$ with the key set $K = \{A, B\}$ and the value set $L = \{X, Y, Z\}$, an example input sequence $\mathbf{x}$ and its correponding target output $\mathbf{y}$ are given as follows

$$\begin{array}{rcl} \mathbf{x} & = & [X \quad A \quad Y \quad Z \quad X \quad B \quad Z \quad X \quad X \quad B \quad X \quad A \quad B] \\ \mathbf{y} & = & [\,. \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad Z \quad . \quad Y \quad X] \end{array},$$

where dots in $\mathbf{y}$ meaning that the model output is not evaluated on this position. In practice, we enforce the model to output identity on this position as it provides more interpretable results.

Empirical Performance. We compare Mamba with Transformers (TF) variants with similar model sizes for solving MQAR, summarized in Tab. 1. We observe that Mamba (top row) and 2-layer Transformer with convolution layer and RoPE (bottom row) perform similarly, whereas Transformers without RoPE (second, third row) struggle to solve the task. From our analysis of the Mamba solution (see details in Sec. 3), we see that Mamba can distinguish the latest key-value association by slowly forgetting using the state matrix. It seems that Transformers can also achieve this via RoPE to differentiate the earlier versus later positions, and fail to solve the task without positional encoding.

Table 1: Summary of Performance in MQAR Task, varying across sequence length T, the number of keys κ, and architectures.

	Model		$T = 128$		$T = 256$		$T = 512$	
	Backbone	Conv.	$\kappa = 1$	$\kappa = 4$	$\kappa = 1$	$\kappa = 4$	$\kappa = 1$	$\kappa = 4$
Acc.(%) ↑	1-layer Mamba	✓	100	100	100	99	100	99
	2-layer TF	✗	30	42	22	21	15	18
	1-layer TF	✓	34	56	26	42	21	36
	2-layer TF w/ RoPE	✓	100	100	100	96	77	90

B Model Simplification Details

Convolution simplification. The Mamba block includes a depthwise convolution Fig. 3. We simplify this learnable convolution to a pairwise concatenation (with padding at the beginning). The convolved embedding at time t is given by

$$\hat{\mathbf{x}}_t = \begin{bmatrix} \mathbf{e}_{x_{t-1}} \\ \mathbf{e}_{x_t} \end{bmatrix}$$

Therefore, the hidden dimension is equals to $2 \cdot$ vocab_size. Due to our use of one-hot embedding vectors, we note that the input term in the Mamba SSM (2) can only write on the lines x_{t-1} and $|V| + x_t$, namely

$$\Delta_t \odot \mathbf{x}_t = \begin{pmatrix} 0 \\ \cdots \\ 0 \\ \Delta_{t,v_{t-1}} \\ 0 \\ \cdots \\ 0 \\ \Delta_{t,v_t+\text{vocab\_size}} \\ \cdots \\ 0 \end{pmatrix}$$

and then

$$(\Delta_t \odot \mathbf{x}_t) \otimes B_t \begin{pmatrix} 0 & \cdot & 0 \\ \vdots & \ddots & \vdots \\ \Delta_{t,v_{t-1}} \cdot B_1 & \vdots & \Delta_{t,v_{t-1}} \cdot B_N \\ 0 & \cdot & 0 \\ \vdots & \ddots & \vdots \\ \Delta_{t,v_{t+\text{vocab\_size}}} \cdot B_1 & \vdots & \Delta_{t,v_{t+\text{vocab\_size}}} \cdot B_N \\ 0 & \cdot & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdot & 0 \end{pmatrix}$$

Simplification of the output matrix. After y_t is computed, an output matrix is applied in order to retrieve the token. This matrix is supposed to be a learned parameter. However, when training the original Mamba model to solve MQAR , we find that the output matrix often admits very simple form, i.e. something close to $(-I, I)$ as shown in Fig. 7.

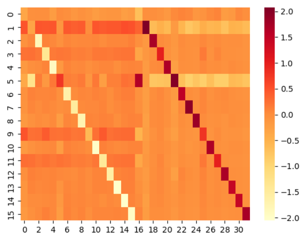


Figure 7: Unembedding matrix after training

C Empirical results details

Structure of the matrix W_Δ Recall that $\Delta_t = \text{Softplus}(W_\Delta x_t)$. After training, the matrix W_Δ exhibits a clear block structure (see Fig. 8). In fact, its weights are close to zero almost everywhere, except in the second part of rows 0 and V. This enables the emergence of specific write/retrieval mechanisms whenever token 0 (i.e., the key in this example) appears.

Mechanisms details The three different mechanisms appearing during the sequence processing are the following:

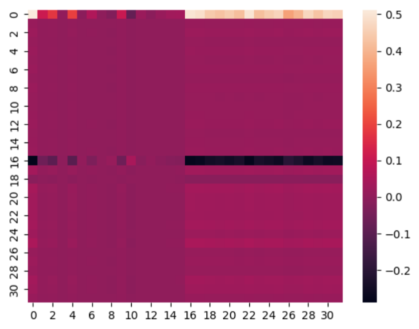


Figure 8: Plot of the matrix W_Δ after training

- (write) When $(x_{t-1}, x_t) = (k, v)$, erase and write the new value. In this case the forgetting term is equal to

$$\exp(\Lambda \odot \Delta_t) \approx 0.5e_0 + 1e_{|V|} + 0.05 \sum_{i=1}^{|V|-1} + 0.3 \sum_{i=|V|+1}^{2|V|-1} = \begin{pmatrix} 0.5 \\ 0.05 \\ 0.05 \\ \vdots \\ 1 \\ 0.3 \\ \vdots \\ 0.3 \end{pmatrix},$$

  so basically all the rows drop close to 0 in the hidden state. Then

$$\Delta_t \odot \hat{\mathbf{x}}_t \approx 15e_0 + 30e_{|V|+v} = \begin{pmatrix} 15 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 30 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

  It means that the row corresponding to the new value becomes much more important than all the other values $\rightarrow$ this new value is written in the hidden state.

- (read) When $(x_{t-1}, x_t) = (v, k)$, retrieve stored value from memory. In this case the model forget as little as possible, especially the second part of the hidden (i.e. the part that will be positive when the output is applied):

$$\exp(\Lambda \odot \hat{\mathbf{x}}_t) \approx 0.9(e_0 + e_{|V|}) + 0.4 \sum_{i=1}^{|V|-1} e_i + 0.8 \sum_{i=|V|+1}^{2|V|-1} = \begin{pmatrix} 0.9 \\ 0.4 \\ \vdots \\ 0.4 \\ 0.9 \\ 0.8 \\ \vdots \\ 0.8 \end{pmatrix} \text{ and}$$

$$\Delta_t \odot \hat{\mathbf{x}}_t \approx 10e_v + 10e_{k+|V|}.$$

- When $(x_{t-1}, x_t) = (v_1, v_2)$, forget all the memories slowly:

$$\exp(\Lambda \odot \hat{\mathbf{x}}_t) \approx 0.6 \sum_{i=0}^{|V|-1} e_i + 0.4 \sum_{i=|V|}^{2|V|-1} e_i = \begin{pmatrix} 0.6 \\ \vdots \\ 0.6 \\ 0.4 \\ \vdots \\ 0.4 \end{pmatrix}$$

and

$$\Delta_t \odot \hat{\mathbf{x}}_t \approx 10 e_{v_1} + 10 e_{v_2+|V|}.$$

D Theoretical Analysis Details

We consider the task MQAR with $|K| = 1$ (without loss of generality let $k = 0$ be the only key). To simplify the analysis, we consider the input sequence having the key appearing at the first and last positions, interleaving with other non-key tokens, i.e. $\mathbf{x}_0 = \mathbf{x}_T = 0, \mathbf{x}_t \neq 0$ for all $t = 1, \ldots, T-1$. To solve the task, the model needs to correctly output $\mathbf{x}_2$ at the query position $\mathbf{x}_T = 0$. One solution is to set $\Delta_0 > 0, \Delta_t = 0$ for all $t \geq 1$, such that *the hidden state only writes and stores the first key-value pair*. In what follows, we will show that one gradient step indeed updates the parameter W_Δ to achieve such condition.

Recall from our simplified Mamba SSM for MQAR $|K| = 1$ that sets $B = C = 1$ (3), namely

$$\mathbf{h}_t = \exp(\mathbf{\Lambda} \odot \Delta_t) \odot \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t, \quad \mathbf{h}_0 = 0, \tag{6}$$

where $\Delta_t = W_\Delta x_t$ [1]. As a first step, we focus on the optimization dynamics of W_Δ, and thus fixing the state matrix $\mathbf{\Lambda} = \mathbf{1}$ (i.e., no forgetting). In this case, the equation simplifies to

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t. \tag{7}$$

Unrolling the recurrence we obtain

$$\mathbf{h}_t = \sum_{i=0}^{t} \Delta_i \odot \hat{\mathbf{x}}_t. \tag{8}$$

Recall by construction, we have one-hot embedding vectors and a pairwise concatenation layer (i.e. a fixed depthwise convolution) to produce the input $\hat{\mathbf{x}}$ for the SSM. Therefore, the k-th component of the hidden state $\mathbf{h}_t \in \mathbb{R}^{2|V|}$ is given by

$$h_{t,k} = \sum_{i=0}^{t} (W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_i+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k \vee \mathbf{x}_i+|V|=k} \tag{9}$$

The training population loss is given by

$$\ell(W_\Delta) = \mathbb{E}_{(\mathbf{x}_1,\ldots,\mathbf{x}_T)\sim P}(\ell(\mathbf{x}_2, W_o\, \mathbf{h}_T) \mid \mathbf{x}_1 = \mathbf{x}_T = 1, \mathbf{x}_2, \ldots, \mathbf{x}_{T-1} \neq 0), \tag{10}$$

where ℓ is the cross-entropy loss, and W_o is the output matrix. Since the output matrix is chosen to be $W_o = [-I, I]$, we are interested in the quantity $\mathbf{h}_{t,k+|V|} - \mathbf{h}_{t,k}$ for all $k < |V|$ (which produces the output $\mathbf{y}_{t,k}$),

$$\begin{aligned} \mathbf{h}_{t,k+|V|} - \mathbf{h}_{t,k} = & \sum_{i=0}^{t} (W_{\Delta_{k+|V|,\mathbf{x}_{i-1}}} + W_{\Delta_{k+|V|,\mathbf{x}_i+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k+|V| \vee \mathbf{x}_i+|V|=k+|V|} \\ & - \sum_{i=0}^{n} (W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_i+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k \vee \mathbf{x}_i+|V|=k} \end{aligned}. \tag{11}$$

[1]The original Mamba proposes to compute $\Delta_t = \text{softplus}(W_\Delta x_t)$; we omit the softplus nonlinearity for simplicity

This can be simplified to the following:

$$\begin{aligned}\mathbf{h}_{t,k+|V|} - \mathbf{h}_{t,k} = &\sum_{i=0}^{t}(W_{\Delta_{k+|V|,\mathbf{x}_{i-1}}} + W_{\Delta_{k+|V|,\mathbf{x}_i+|V|}})\mathbb{1}_{\mathbf{x}_i=k} \\ &- \sum_{i=0}^{n}(W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_i+|V|}})\mathbb{1}_{\mathbf{x}_{i-1}=k}\end{aligned} \tag{12}$$

To simplify notation, we write $z_{n,k} = \mathbf{h}_{t,k+|V|} - \mathbf{h}_{t,k}$.
The gradient of the loss with respect to W_Δ is given by

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \sum_{k=1}^{|V|} \frac{\partial \ell}{\partial z_{T,k}} \frac{\partial z_{T,k}}{\partial W_{\Delta_{i,j}}} \tag{13}$$

From 12, we have

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \begin{cases} \frac{\partial \ell}{\partial z_{T,i}} \frac{\partial z_{T,i}}{\partial W_{\Delta_{i,j}}} & \text{if } i \leq |V| \\ \frac{\partial \ell}{\partial z_{T,i-|V|}} \frac{\partial z_{T,i-|V|}}{\partial W_{\Delta_{i,j}}} & \text{if } i > |V| \end{cases} \tag{14}$$

because all derivative are equals to 0 when $k \neq i$. We can then distinguish 4 cases:

- Either $i \leq |V|$ and $j \leq |V|$. Then

$$\frac{\partial z_{T,i}}{\partial W_{\Delta_{i,j}}} = -\sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_{l-1}=i} \tag{15}$$

- Either $i \leq |V|$ and $j > |V|$. Then

$$\frac{\partial z_{T,i}}{\partial W_{\Delta_{i,j}}} = -\sum_{l=0}^{T} \mathbb{1}_{j=\mathbf{x}_l+|V|} \mathbb{1}_{\mathbf{x}_{l-1}=i} \tag{16}$$

- Either $i > |V|$ and $j \leq |V|$. Then

$$\frac{\partial z_{T,i-|V|}}{\partial W_{\Delta_{i,j}}} = \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_l=i} \tag{17}$$

- Either $i > |V|$ and $j > |V|$. Then

$$\frac{\partial z_{T,i-|V|}}{\partial W_{\Delta_{i,j}}} = \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_l+|V|=j} \mathbb{1}_{\mathbf{x}_l=i} \tag{18}$$

Furthermore,

$$\frac{\partial \ell}{\partial z_{T,i}} = \begin{cases} -1 + \frac{e^{z_{T,i}}}{\sum_{m=1}^{|V|} e^{z_{T,m}}} & \text{if } i = \mathbf{x}_1 \\ \frac{e^{z_{T,i}}}{\sum_{m=1}^{|V|} e^{z_{T,m}}} & \text{otherwise} \end{cases} \tag{19}$$

At initialization, $z_T = 0$ so

$$\frac{\partial \ell}{\partial z_{T,i}} = \begin{cases} -1 + \frac{1}{|V|} & \text{if } i = \mathbf{x}_1 \\ \frac{1}{|V|} & \text{otherwise} \end{cases} \tag{20}$$

So we end up with

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \begin{cases} \frac{-(V-1)}{V} \cdot (-\sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_{l-1}=i}) & \text{if } i = \mathbf{x}_1, j \leq |V| \\ \frac{1}{|V|} \cdot (-\sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_{l-1}=i}) & \text{if } i \leq |V|, i \neq \mathbf{x}_1, j \leq |V| \\ \frac{-(V-1)}{V} \cdot \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_{l}=i} & \text{if } i = \mathbf{x}_1 + |V|, j \leq |V| \\ \frac{1}{|V|} \cdot \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_{l-1}=j} \mathbb{1}_{\mathbf{x}_{l}=i} & \text{if } i > |V|, i \neq \mathbf{x}_1 + |V|, j \leq |V| \\ \frac{-(V-1)}{V} \cdot (-\sum_{l=0}^{T} \mathbb{1}_{j=\mathbf{x}_l+|V|} \mathbb{1}_{\mathbf{x}_{l-1}=i}) & \text{if } i = \mathbf{x}_1, j > |V| \\ \frac{1}{|V|} \cdot (-\sum_{l=0}^{T} \mathbb{1}_{j=\mathbf{x}_l+|V|} \mathbb{1}_{\mathbf{x}_{l-1}=i}) & \text{if } i \leq |V|, i \neq \mathbf{x}_1, j > |V| \\ \frac{-(V-1)}{V} \cdot \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_l+|V|=j} \mathbb{1}_{\mathbf{x}_{l}=i} & \text{if } i = \mathbf{x}_1 + |V|, j > |V| \\ \frac{1}{|V|} \cdot \sum_{l=0}^{T} \mathbb{1}_{\mathbf{x}_l+|V|=j} \mathbb{1}_{\mathbf{x}_{l}=i} & \text{if } i > |V|, i \neq \mathbf{x}_1 = |V|, j > |V| \end{cases} \tag{21}$$

Then

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \mathbb{E}_{(\mathbf{x}_0,\ldots,\mathbf{x}_T)\sim P}\left(\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} | \mathbf{x}_0 = \mathbf{x}_T = 0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1} \neq 0\right) \tag{22}$$

$$= \begin{cases} \frac{-1}{n} & \text{if } i = 0 \\ \frac{2}{n} & \text{if } i = |V| \\ \approx \frac{n}{|V|^2} - \frac{1}{|V|} & \text{if } i < |V|, i \neq 0 \\ \approx \frac{-n}{|V|^2} + \frac{1}{|V|} & \text{if } i > |V| \end{cases} \tag{23}$$

So in the case $|V| \gg T$, the first gradient step will mostly modify the rows 0 and $|V|$ in W_Δ, aligning it with the key embedding vector while being orthogonal to the other non-key vectors. This explains our observation in Sec. 3 on how Mamba learns to solve the task via using the writing mechanism.