How does Mamba Perform Associative Recall? A Mechanistic Study

Anonymous Author(s)

Affiliation Address email

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 with appropriate 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 Mamba on associative recall tasks. By analyzing the learned model weights and the hidden state evolution, we uncover the mechanisms used by trained Mamba models to perform associative recall. We complement our study with theoretical analysis on the optimization dynamics of Mamba that give rise to such mechanisms.

Introduction 12

2

3

4

6

8

9

10

11

24

25

26

27

- Recently, structured state space models (SSMs) have arisen as competitive sequence modeling 13 architectures [7, 9, 8]. In particular, Mamba [6] has emerged as a promising general-purpose 14 sequence model, demonstrating competitive performance on various language modeling tasks while 15 reducing the quadratic complexity in Transformers [12] to linear time. Theoretical understanding of 16 Mamba begins to emerge, centering on its expressivity using approaches such as formal language 17 theory [11, 5] and approximation theory [3, 10].
- Focusing on Mamba's associative recall capabilities, [10] showed that there exists a Mamba model 19 with well-chosen weights that can solve certain associative recall tasks. However, it remains unclear 20 whether training Mamba with gradient descent will find such a solution, or whether a different 21 mechanism is found in practice. To fill this gap, we conduct a mechanistic study to probe how Mamba 22 can learn to perform associative recall. Our main contributions can be summarized as follows: 23
 - 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). 28

Problem Setup 29

2.1 The MQAR task

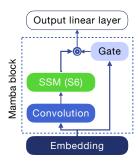
To evaluate the associative recall capabilities of Mamba, we adapt the Multiple-Query Associative Recall (MQAR [1, 10]) task and consider a more difficult variant, illustrated in Fig. 2 with full **Input**: Alice used to live in Paris, then she moved to New York, and nowadays Alice lives in Barcelona.

Query: Where does Alice live?

Answer: Barcelona

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

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.



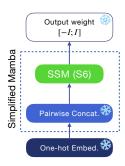


Figure 3: The original Mamba architecture (left), and our simplification (right) which disables the gating branches, and fixes the embedding, convolution, and output layers.

details described in App. A. In particular, a key $k_i \in K$ can be associated with many different values v_{i_1}, \ldots, v_{i_m} in the context. Upon querying the key k_i , the *latest* associated value v_{i_m} must be

values v_{i_1}, \dots, v_{i_m} in the context. Spon querying the key v_i , the tatest 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.

39 Our design of MQAR is motivated from sturctures in natural language (e.g., Fig. 1), where language

40 understanding requies recalling the appropriate reference of a pronoun or entity among many distrac-

41 tors. 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

43

46

47

48

51

52

53

The main architecture of Mamba is illustrated in Fig. 3 (left). To probe the inner-workings of Mamba inspired by [10], we simplify Mamba as follows (see Fig. 3 right for an overview):

- **Embedding**: One-hot embedding $v \mapsto \mathbf{e}_v$ for token v drawn from the vocabulary V.
- **Convolution**: Replace the depthwise learnable convolution with a *concatenation* of two consecutive tokens (i.e. a fixed convolution of kernel size 2 and left padding), i.e.

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

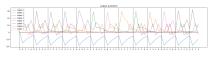
The convolved token at t groups the original tokens at positions (t-1,t), as $\hat{\mathbf{x}}_t = \begin{bmatrix} \mathbf{e}_{x_{t-1}} & \mathbf{e}_{x_t} \end{bmatrix}^\top \in \mathbb{R}^{2|V|}$. This also fixes the hidden model dimension as d=2|V|.

- Output linear layer: Replace the learnable linear map with a fixed weight matrix $W = \begin{bmatrix} -I & I \end{bmatrix} \in \mathbb{R}^{|V| \times 2|V|}$, motivated from our observations on trained models (App. B).
- **Gate**: Disable the gating branch.

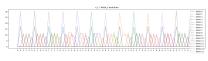
We preserve the key SSM layer in Mamba, defined as

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

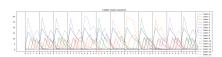
where $\Lambda \in \mathbb{R}^{d \times N}$, $\Delta_t \in \mathbb{R}^d$ and $B(x_t), C(x_t) \in \mathbb{R}^N$ (N is the state size). We set $\Delta_t = \text{Softplus}(W_{\Delta}\hat{x}_t)$, where $W_{\Delta} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix.



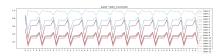
(a) Evolution of output y_t (token 0 is the key)



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



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



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

Figure 4: Probing Mamba on MQAR with K=1: Each line represents the correponding dimension with $d=0,\ldots,|V|-1$ for the output $\mathbf{y}_t \in \mathbb{R}^{|V|}$, and $d=0,\ldots,2|V|-1$ for the rest (dashed lines denote the first half $d=0,\ldots,|V|-1$ while solid lines denote second half $d=|V|,\ldots,2|V|-1$).

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

59 width, leading to the desired orthogonality (see [2] for an explanation in the context of Transformers).

60 3 Probing the Learned Mechanism

61 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 (we set $K=\{0\}$

in our experiments), it is sufficient to use a state size N=1 [10], reducing $B(x_t)$, $C(x_t)$ in (1) to

scalars. Thus, we fix B=C=1 which simplifies the SSM as

$$\mathbf{h}_t = e^{\mathbf{\Lambda} \odot \Delta_t} \odot \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t. \tag{2}$$

We train the model on our MQAR variant with vocabulary size V=8, a maximum noise length of

 $n_{\epsilon} = 3$ tokens, and sequence length T = 256, until it reaches 100% accuracy on the training set. We

67 then evaluate the model after training to identify how Mamba can perfectly solve such task.

68 We probe the trained Mamba model behavior by plotting the evoluion (per each dimension) of its

output \mathbf{y}_t , 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

70 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

73 the last d = |V| dimensions). Fig. 4c corresponds to the input term, that is, what's being newly

vitten. We clearly see a spike when a new value is written in last d = |V| dimensions. Fig. 4d shows

75 the forgetting factor, which seems to fully erase everything right before writing a new value.

⁷⁶ Concretely, we observe a clear block structure in the W_{Δ} matrix, shown in Fig. 7 (see App. C),

leading to the following key mechanisms in the hidden state based on the input pair at the current

position t (here we let the key token be k=0 and value (noise) tokens be $v, v_1, v_2, x \in V \setminus \{0\}$):

• (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

87 the values seen in distant past.

79

80

81

82

83

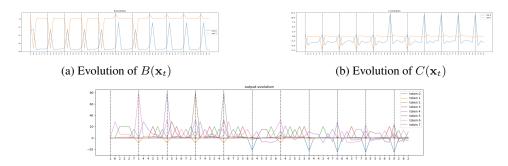
84

88 Case |K|=2. In this setting, we use the original $B(\mathbf{x}_t), C(\mathbf{x}_t) \in \mathbb{R}^N$ vectors without simplifica-

tion. The N=|K| solution proposed in [10] for MQAR with |K|=2 treats $B(\mathbf{x}_t)$ as a routing vec-

tor, sending the first key-value pair to the first hidden state dimension by $B([\mathbf{e}_{k_1},\mathbf{e}_{v_1}]^\top) \equiv B_1 = \mathbf{e}_1$,

and the second key-value pair to the second dimension by $B([\mathbf{e}_{k_2},\mathbf{e}_{v_2}]^\top) \equiv B_2 = \mathbf{e}_2$; Retrieval is done similarly by $C(\mathbf{x}_t)$ where $C([\mathbf{e}_x,\mathbf{e}_{k_1}]^\top) \equiv C_1 = \mathbf{e}_1, C([\mathbf{e}_v,\mathbf{e}_{k_2}]^\top) \equiv C_2 = \mathbf{e}_2$. The task can in general be solved using $N = O(\log |K|)$ state size, via almost orthogonal routing and retrieval 91 92 vectors [10]. We empirically observe this behavior: Fig. 5a shows $B_1 \approx [-1, 0.4]^{\top}, B_2 \approx [-3, -7]^{\top}$ and thus $\langle B_1, B_2 \rangle \approx 0$; Fig. 5b shows $C_1 \approx [0, -7.5], C_2 \approx [12.5, 0]$ and thus $\langle C_1, C_2 \rangle \approx 0$.



(c) Evolution of our y_t : the tokens 0, 1 represent the two keys, highlighted by vertical dashed lines.

Figure 5: Probing Mamba on MQAR with K = 2.

Theoretical Analysis 96

102

103

104

105

106

107 108

111

To support our empirical observation, we analyze the optimization dynamics of a simplified Mamba 97 model (see Sec. 2.2) on MQAR with |K|=1. We focus on the parameter W_{Δ} (used to compute 98 $\Delta_t = W_{\Delta} \hat{\mathbf{x}}_t$, 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 $W_{\Delta} = 0$: 99 100 101

$$W_{\Delta} = -\nabla_{W_{\Delta}} L = -\mathbb{E}_{(\mathbf{x}_0, \dots, \mathbf{x}_T) \sim P} \left[\nabla_{W_{\Delta}} \ell \mid \mathbf{x}_0 = \mathbf{x}_T = 0, \mathbf{x}_1, \dots, \mathbf{x}_{T-1} \neq 0 \right], \tag{3}$$

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. **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 (3) takes the form:

$$\frac{\partial L}{\partial W_{\Delta_{i,j}}} = \begin{cases}
\frac{-1}{T} & \text{if } i = 0 \\
\frac{2}{T} & \text{if } i = |V| \\
\approx \frac{T}{|V|^2} - \frac{1}{|V|} & \text{if } 0 < i < |V| \\
\approx \frac{-T}{|V|^2} + \frac{1}{|V|} & \text{if } i > |V|
\end{cases}$$
(4)

For $|V| \gg T$, wee see that one gradient step mostly updates the weights on the 0, |V|-th rows in W_{Δ} , 109 and thus correctly succeeds at performing the read and write operations for associative recall. 110

Conclusion and Future Directions

Towards understanding how Mamba perform associative recall, we perform a mechanistic study 112 to probe trained Mamba models (with suitable simplifications) for solving the MQAR tasks. We 113 identify key associative-recall mechanisms in Mamba's hidden state, including writing, retriveing, 114 and forgetting behaviors based on the input. Our insights are supported via empirical results on 115 probing the trained models and theoretical analysis of Mamba's population gradient dynamics. 116 As a first step, our mechanistic study focuses on a simplified Mamba model; extending the analysis to 117

the original Mamba and Mamba-2 [4] and considering more complicated tasks is a natural next step. 118 Another interesting direction lies in comparing the associative-recall mechanisms in Mamba with other sequence model architectures, such as Transformers and other subquadratic variants.

References

121

- [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, Hervé Jégou, and Léon Bottou. Birth of a transformer: A memory viewpoint. *arXiv:2306.00802*, 2023.
- 127 [3] Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons.
 128 Theoretical Foundations of Deep Selective State-Space Models. In *Advances in Neural Infor-*129 *mation 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.
- 133 [5] Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano 134 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] 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

153

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_{ϵ} , 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_{ϵ}
- 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 x
- 11: **return** The sequence 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 ${\bf x}$ and its corresponding target output ${\bf y}$ are given as follows

where dots in 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 2-layer TF 1-layer TF	✓ × ✓	100 30 34	100 42 56	100 22 26	99 21 42	100 15 21	99 18 36
	2-layer TF w/ RoPE	✓	100	100	100	96	77	90

169 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

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

Therefore, the hidden dimension is equals to 2 · vocab_size. Due to our use of one-hot embedding vectors, we note that the input term in the Mamba SSM (1) can only write on the lines x_{t-1} and 171 $|V| + x_t$, namely 172

$$\Delta_t \odot \mathbf{x}_t = \begin{pmatrix} 0 & \dots & 0 \\ \Delta_{t,v_{t-1}} & 0 & \dots & 0 \\ 0 & \dots & 0 & \dots & 0 \\ \Delta_{t,v_t + \text{vocab_size}} & \dots & 0 \end{pmatrix}$$

and then

174

175

176

177

178

179

180

181 182

183

$$(\Delta_t \odot \mathbf{x}_t) \otimes B_t egin{pmatrix} 0 & \cdot & 0 \\ \vdots & \cdot \cdot & \vdots \\ \Delta_{t,v_{t-1}} \cdot B_1 & \vdots & \Delta_{t,v_{t-1}} \cdot B_N \\ 0 & \cdot & 0 \\ \vdots & \cdot \cdot & \vdots \\ \Delta_{t,v_{t+ ext{vocab_size}}} \cdot B_1 & \vdots & \Delta_{t,v_{t+ ext{vocab_size}}} \cdot B_N \\ 0 & \cdot & 0 \\ \vdots & \cdot \cdot & \vdots \\ 0 & \cdot & 0 \end{pmatrix}$$

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. 6.

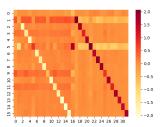


Figure 6: Unembedding matrix after training

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. 7). 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: 184

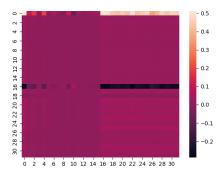


Figure 7: Plot of the matrix W_{Δ} after training

• (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

185

186

187

188

189

190

191

192

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

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:

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

195 and

194

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

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 (2), namely

$$\mathbf{h}_t = \exp(\mathbf{\Lambda} \odot \Delta_t) \odot \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t, \quad \mathbf{h}_0 = 0, \tag{5}$$

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 $\Lambda = 1$ (i.e., no forgetting). In this case, the equation simplifies to

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \Delta_t \odot \hat{\mathbf{x}}_t. \tag{6}$$

207 Unrolling the recurrence we obtain

$$\mathbf{h}_t = \sum_{i=0}^t \Delta_i \odot \hat{\mathbf{x}}_t. \tag{7}$$

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

$$h_{t,k} = \sum_{i=0}^{t} (W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_{i+|V|}}}) \mathbb{1}_{\mathbf{x}_{i-1} = k \vee \mathbf{x}_{i} + |V| = k}$$
(8)

211 The training population loss is given by

$$\ell(W_{\Delta}) = \mathbb{E}_{(\mathbf{x}_1, \dots, \mathbf{x}_T) \sim P}(\ell(\mathbf{x}_2, W_o \, \mathbf{h}_T) \mid \mathbf{x}_1 = \mathbf{x}_T = 1, \mathbf{x}_2, \dots, \mathbf{x}_{T-1} \neq 0), \tag{9}$$

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 $\mathbf{y}_{t,k}$),

$$\mathbf{h}_{t,k+|V|} - \mathbf{h}_{t,k} = \sum_{i=0}^{t} (W_{\Delta_{k+|V|,\mathbf{x}_{i-1}}} + W_{\Delta_{k+|V|,\mathbf{x}_{i}+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k+|V|\vee\mathbf{x}_{i}+|V|=k+|V|} - \sum_{i=0}^{n} (W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_{i}+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k\vee\mathbf{x}_{i}+|V|=k}$$
(10)

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

215 This can be simplified to the following:

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

$$- \sum_{i=0}^{n} (W_{\Delta_{k,\mathbf{x}_{i-1}}} + W_{\Delta_{k,\mathbf{x}_{i}+|V|}}) \mathbb{1}_{\mathbf{x}_{i-1}=k}$$
(11)

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

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \sum_{k=1}^{|V|} \frac{\partial \ell}{\partial z_{T,k}} \frac{\partial z_{T,k}}{\partial W_{\Delta_{i,j}}}$$
(12)

218 From 11, we have

$$\frac{\partial \ell}{\partial W_{\Delta_{i,j}}} = \begin{cases} \frac{\partial \ell}{\partial z_{T,i}} \frac{\partial z_{T,i}}{\partial W_{\Delta_{i,j}}} & \text{if } i \leq |V| \\ \frac{\partial \ell}{\partial z_{T,i-|V|}} \frac{\partial z_{T,i-|V|}}{\partial W_{\Delta_{i,j}}} & \text{if } i > |V| \end{cases}$$
(13)

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

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

$$\tag{14}$$

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

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

$$\tag{15}$$

• Either i > |V| and $j \le |V|$. Then

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

$$\tag{16}$$

• Either i > |V| and j > |V|. Then

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

$$\tag{17}$$

224 Furthermore,

$$\frac{\partial \ell}{\partial z_{T,i}} = \begin{cases}
-1 + \frac{e^{z_{T,i}}}{\sum_{m=1}^{|V|} e^{z_{T,m}}} & \text{if } i = \mathbf{x}_1 \\
\frac{e^{z_{T,i}}}{\sum_{m=1}^{|V|} e^{z_{T,m}}} & \text{otherwise}
\end{cases}$$
(18)

225 At initialization, $z_T = 0$ so

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

226 So we end up with

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

227 Then

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

$$= \begin{cases}
\frac{-1}{n} & \text{if } i = 0 \\
\frac{2}{n} & \text{if } i = |V| \\
\approx \frac{n}{|V|^2} - \frac{1}{|V|} & \text{if } i < |V|, i \neq 0 \\
\approx \frac{-n}{|V|^2} + \frac{1}{|V|} & \text{if } i > |V|
\end{cases}$$
(21)

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.