

Scan Order Is a Reasoning Prior: Geometry-Aware Multi-Directional Mamba-2 for Tiny Recursive Reasoning

Zhenxuan Yu*
Rikkyo University

Abstract

Constraint-satisfaction tasks such as Sudoku require predictions that are globally consistent: a single incorrect cell invalidates a solution. Recursive reasoning models address this requirement through repeated latent-state updates, while state-space operators offer sequence mixing with linear complexity in sequence length. Their traversal order, however, determines how information propagates across a structured input. We study this choice within the Tiny Recursive Model on Sudoku-Extreme. With a common recursive scaffold and optimizer settings, two-, four-, and six-direction Mamba-2 operators use the same parameterized mixer architecture with different scan orderings, each with 5.81M parameters. In a single-seed comparison, exact-match accuracy rises from 31.42% for bidirectional row scans to 64.49% for row and column scans, but falls to 42.96% when box-major scans are added. An equal-parameter unidirectional configuration reaches 0.00%. Combining shared directional weights, a second mixer, and learned positions reaches 80.83% at 5.86M parameters without attention or a token-mixing MLP, compared with 81.80% for our Mamba-2/MLP-t baseline at 5.96M. These results identify scan geometry as an effective inductive bias in this Sudoku setting; robustness across seeds and transfer to other tasks remain untested.

1 Introduction

A recurrent sequence model processes positions in a specified order. Full self-attention can connect every pair of positions, with positional encodings or masks supplying ordering information. State-space models instead propagate information through a recurrence with linear complexity in sequence length (Gu et al., 2022; Gu and Dao, 2023). Their traversal order therefore determines the paths through which information reaches each

position. This choice is particularly relevant when a shared operator is applied recursively (Deng et al., 2026; Yu et al., 2026) to inputs with two-dimensional structure.

We study traversal within the Tiny Recursive Model (TRM; Jolicoeur-Martineau, 2025), which builds on the Hierarchical Reasoning Model (Wang et al., 2025). TRM refines a latent state and a candidate answer through repeated applications of a small shared operator. Wang and Reid (2026) replace its Transformer blocks with Mamba-2 hybrids (Dao and Gu, 2024). This framework is useful for studying traversal for two reasons. First, the operator is applied 21 times per supervision step, repeatedly exposing the states to the same traversal bias. Second, Sudoku-Extreme has explicit row, column, and 3×3 box constraints, and exact match requires every cell to be correct. It provides a controlled setting for studying spatial constraint satisfaction, although it cannot establish generalization beyond Sudoku.

Traversal determines which cells can influence each output within a scan. A row-major sweep makes cells in each row consecutive, while cells in a column are farther apart. Multi-directional scanning is established in vision (Zhu et al., 2024; Liu et al., 2024; Huang et al., 2024; Yang et al., 2024). Here we examine whether aligning scans with constraint axes changes the performance of a recursively applied operator, and whether adding another constraint-aligned ordering necessarily helps.

At a fixed budget of 5,810,786 parameters, bidirectional row scans reach 31.42% exact match, row and column scans reach 64.49%, and adding box-major scans reduces accuracy to 42.96%. These three configurations use the same mixer arrangement. Equal-parameter variants also examine unidirectional scanning and shared directional weights, with different mixer depths. The strongest scan-only configuration reaches 80.83%

*E-mail: 24wr006m@rikkyo.ac.jp

after adding learned positions, compared with 81.80% for our reimplemented hybrid. We report these as single-seed observations rather than evidence of statistical equivalence or consistent superiority.

Hybrid language models often combine recurrent operators with attention to improve recall and copying (Jelassi et al., 2024; Merrill et al., 2024; Waleffe et al., 2024). Our study asks whether scan geometry can reduce the need for an additional token mixer in a structured reasoning task. The relevance to language generation remains a motivation for future experiments, as discussed in Section 5.¹

2 Related Work

Parameter sharing supports repeated computation in Universal Transformers (Dehghani et al., 2019), deep equilibrium models (Bai et al., 2019), and looped Transformers (Giannou et al., 2023; Saunshi et al., 2025; Zhu et al., 2025; Jeddi et al., 2026; Deng et al., 2026). HRM and TRM apply related ideas to puzzles (Wang et al., 2025; Jolicoeur-Martineau, 2025; Sghaier et al., 2026); earlier Sudoku solvers use explicit relational or logical structure (Palm et al., 2018; Wang et al., 2019). State-space and related recurrent models provide efficient sequence mixing (Gu et al., 2022; Gu and Dao, 2023; Dao and Gu, 2024; Fu et al., 2023; Peng et al., 2023; Sun et al., 2023; Yang et al., 2023), although fixed-size recurrent states can limit copying and state tracking (Jelassi et al., 2024; Merrill et al., 2024). Hybrid models supplement them with attention (Waleffe et al., 2024; Lieber et al., 2024), and recent work uses state-space operators in latent recursion (Yu et al., 2026; Wang and Reid, 2026). We adapt multi-directional scans from vision (Zhu et al., 2024; Liu et al., 2024; Huang et al., 2024; Yang et al., 2024) to test constraint-aligned traversal within a recursive operator. Appendix A gives further context.

3 Method

3.1 The recursive scaffold

We retain the TRM scaffold (Jolicoeur-Martineau, 2025). The embedded input is x , the answer state is z_H , and the latent state is z_L . Both states are initialized from fixed random vectors stored as

buffers. A shared operator f performs

$$z_L \leftarrow f(z_L + z_H + x), \quad (1)$$

$$z_H \leftarrow f(z_H + z_L). \quad (2)$$

Equation 1 is applied six times, followed by one application of Eq. 2. This cycle repeats three times, giving $3 \times (6 + 1) = 21$ operator calls per supervision step. Only the final cycle is differentiated; the first two run without gradient tracking.

The output head decodes z_H , and a halting head uses its first position to predict whether the answer is exactly correct (Graves, 2016). Each supervision step uses a stablemax cross-entropy loss on the answer and a binary halting loss with weight 0.5 (Lee et al., 2014). The states are detached and carried to the next supervision step, with at most 16 steps before reset. Early halting is used during training; evaluation always runs all 16 refinement steps. All configurations share this recursion schedule.

3.2 The operator

Following Wang and Reid (2026), the operator contains Mamba-2 mixers (Dao and Gu, 2024), an optional additional token mixer, and a channel MLP. Each is a post-norm residual sublayer, $h \leftarrow \text{RMSNorm}(h + \text{sublayer}(h))$ (Zhang and Sennrich, 2019; Shazeer, 2020). The optional component is non-causal self-attention (Vaswani et al., 2017; Su et al., 2021), an MLP along the token axis (MLP-t), or absent. We reimplement the Mamba-2/MLP-t hybrid as a baseline under our training protocol. All scan-geometry variants omit this additional mixer, so information passes between cells only through the scan-based sublayers.

3.3 Orderings and directions

Let Π be a set of orderings, each a permutation of the 81 board positions, and let P_π be the operator that applies ordering π , R the reversal of a sequence, and $S_{\rightarrow}, S_{\leftarrow}$ two Mamba-2 modules. A mixer computes

$$y = \sum_{\pi \in \Pi} P_\pi^{-1} (S_{\rightarrow}(P_\pi h) + R S_{\leftarrow}(R P_\pi h)). \quad (3)$$

For each ordering, forward and reversed outputs are aligned and mapped back to row-major order before summation. Each P_π fixes the 16 prefix positions, whereas R reverses the full sequence, in-

¹Code repository: https://github.com/YU-ZHENXUAN-uc1lm/scan_order_reasoning_prior.

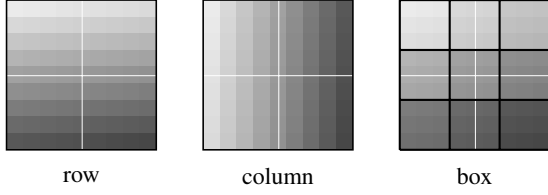


Figure 1: The three orderings on the 9×9 board. Cells are shaded by their position in the traversal, from first (light) to last (dark); the heavy lines on the right grid mark the 3×3 boxes. Each ordering in Π is scanned in both directions (Eq. 3), and each makes exactly one of Sudoku’s three constraint types contiguous.

cluding the prefix. Grid directions are batched together rather than necessarily executed as separate sequential calls.

We use three orderings (Figure 1). Row-major is the identity. Column-major visits (r, c) in order of c and then r . Box-major visits the nine 3×3 boxes in row-major order and the cells within each box in row-major order. Each makes the cells of one constraint type consecutive. In a row-major sequence, vertically adjacent cells are nine positions apart, while the first and last cells of a box are twenty positions apart. There are $2|\Pi|$ directions: two for $\{\text{row}\}$, four for $\{\text{row}, \text{col}\}$, and six for $\{\text{row}, \text{col}, \text{box}\}$. A unidirectional variant omits the reverse term in Eq. 3.

3.4 Holding the parameter count fixed

Each mixer uses one module for all forward orderings and one for all reverse orderings, or a single module if the two directions share weights. Adding orderings therefore adds scan computation but no parameters. The two-, four-, and six-direction untied configurations use one mixer with two modules and differ only in Π . To retain the same parameter count, the unidirectional variant stacks two forward-only mixers, while the tied four-direction variant stacks two mixers, each sharing its forward and reverse weights. These two variants also change mixer depth and are not pure direction ablations. The implementation defaults to activation checkpointing for six-direction scans, recomputing activations during backpropagation to reduce memory use.

Table 1 makes these differences explicit. Each configuration contains two independently parameterized Mamba-2 modules, totaling 5,810,786 dense parameters with the shared components. This count excludes the identical, separately optimized 512-parameter puzzle embedding. Learned

Configuration	Mixers	Tied	Scans
1 direction	2	—	2
2 directions	1	no	2
4 directions	1	no	4
6 directions	1	no	6
4 directions, tied	2	yes	8

Table 1: Equal-parameter configurations. **Tied** indicates forward/reverse sharing within a mixer; different mixers have separate weights. **Scans** counts directional sequence passes per input and operator call, not GPU kernel invocations.

positions and MLP-t are counted separately in Table 3; Appendix B provides the component totals.

4 Experiments

4.1 Setup

We use Sudoku-Extreme as released with TRM: 1,000 training puzzles, each with its original instance and 1,000 augmentations, and 422,786 held-out puzzles. Every run is evaluated on the same first 76,800 held-out puzzles (100 batches). Boards are flattened into 81 row-major cells after the 16-position puzzle-embedding prefix. All runs use AdamATan2 (Everett et al., 2024) with learning rate 10^{-4} , held constant after 2,000 warmup steps, global batch size 768, and 65,100 optimizer updates. Weight decay and gradient clipping are both 1.0. Evaluation uses an exponential moving average (EMA) of the dense parameters with decay 0.999; the sparse puzzle embedding retains its current weights. Appendix C lists the full configuration. Each configuration has one completed run, with seed 0; additional seed runs could not be completed within the available time and compute budget.

We report per-cell accuracy and whole-grid exact match. Unless a peak checkpoint is explicitly identified, results are from the final evaluation. The 76,800-puzzle evaluation subset is fixed across configurations, but performance on the full held-out set is not measured here.

4.2 Scan geometry at a fixed parameter count

Scan configuration has a large observed effect on exact match (Table 2). The unidirectional variant reaches 50.61% per-cell accuracy but 0.00% whole-grid exact match on the evaluation subset. The bidirectional row variant reaches 31.42%. Within the three configurations that hold mixer depth and weight sharing fixed, adding column

Orderings scanned	Dir.	Cell	Exact
row (forward only)	1	50.61	0.00
row	2	76.63	31.42
row, column	4	87.40	64.49
row, column, box	6	81.44	42.96
row, column (tied)	4	91.75	77.53

Table 2: Scan geometry at an identical dense parameter count of 5,810,786. **Dir.** is the number of directions per mixer, **Cell** is per-cell accuracy, and **Exact** is whole-grid exact match, both in percent. The last row shares forward/reverse weights within each of two stacked mixers.

scans increases exact match to 64.49%, while adding box scans reduces it to 42.96%.

Box-major traversal makes another valid Sudoku constraint contiguous, yet the six-direction configuration performs worse than the four-direction configuration in this run. One possible explanation is interference among outputs with different locality patterns when they are summed. The additional scans also change computation and activation scale, so this experiment does not isolate that mechanism. The supported observation is that adding a constraint-aligned ordering did not improve accuracy under this training protocol.

Additional checks. The six-direction operator performs 50% more scans than the four-direction operator, so accuracy does not increase monotonically with scan count. This is not a comparison at equal computational cost. Replacing the sum in Eq. 3 with a mean changes neither parameter count nor scan count, but reduces final exact match from 64.49% to 24.63% for four directions and from 42.96% to 36.61% for six directions. Aggregation therefore also matters in these runs; the mean control does not establish why summation performs better.

4.3 Replacing the token mixer

Replacing one untied four-direction mixer with two tied four-direction mixers raises exact match from 64.49% to 77.53%, a gain of 13.04 percentage points at the same parameter count. This jointly changes weight sharing, mixer depth, and the number of scans per operator call (Table 1); the gain cannot be attributed to weight sharing alone. Learned absolute position embeddings add 49,664 parameters and increase exact match to 80.83%. The implementation adds these embeddings to the input embeddings and rescales the sum by $1/\sqrt{2}$.

Operator	Mixer	Params	Exact
row-major scan	MLP-t	5.96M	81.80
row-major scan [†]	attention	6.86M	66.50
4-way grid scan	none	5.81M	77.53
+ learned pos.	none	5.86M	80.83

Table 3: Comparison with Mamba-2 hybrids. **Mixer** is the cross-token component that follows the scans; MLP-t is an MLP applied along the token axis. The lower two rows use tied directional weights and two stacked mixers, with no attention or token-mixing MLP. [†]As reported by Wang and Reid (2026) under their protocol; all other rows are our runs (Section 4).

This supplies explicit positions but also changes input scaling.

Our Mamba-2/MLP-t baseline reaches 81.80% with 5,959,778 dense parameters (Table 3). The scan-only model with learned positions reaches 80.83% with 5,860,450, approximately 1.7% fewer parameters. Its 0.97-percentage-point gap to the baseline is a descriptive comparison: without repeated seeds, neither statistical equivalence nor a reliable performance difference can be established. The published attention-hybrid result (66.5%) uses the protocol of Wang and Reid (2026) and is included only as context, not as a matched comparison. Our reimplementations also differs from their reported 6.28M MLP-t configuration.

4.4 Training dynamics

Figure 2 shows the ten evaluations for the main configurations. The unidirectional variant remains at 0.00% exact match throughout. The six-direction variant fluctuates substantially and peaks at 47.08% at step 52,080, below the four-direction variant’s final 64.49%. Other curves also show occasional decreases; all configurations in Tables 2 and 3 trained by us, except the six-direction variant, reach their best observed accuracy at the final evaluation. These curves describe variation across checkpoints within a run, not variation across random seeds.

The four-direction mean-aggregation control, not plotted, peaks at 51.66% at step 52,080 before falling to 24.63%. This suggests training instability in that run, but does not establish its cause or how consistently it occurs across seeds.

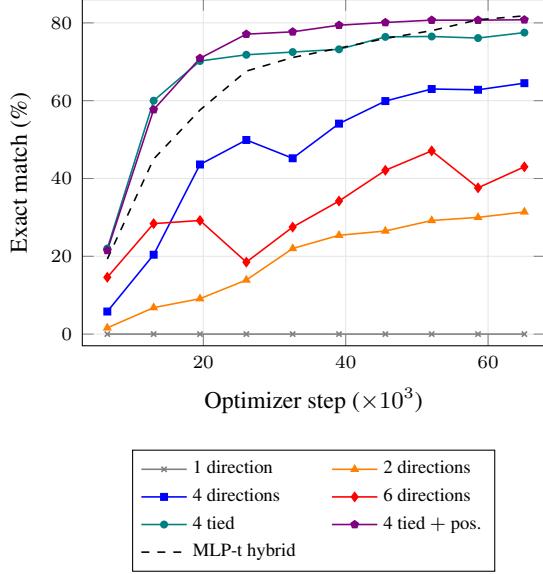


Figure 2: Exact match over training: the five equal-parameter configurations of Table 2, the learned-position variant, and our MLP-t hybrid baseline (dashed). All curves use seed 0 and the same evaluation subset. Points are rounded to one decimal place; the tables report final results to two decimal places.

5 Discussion and Conclusion

Implications for structured generation. Tables, code, and document infilling can contain dependencies that are not local in a left-to-right order. The Sudoku results motivate studying traversal choices for such outputs when using recurrent operators. However, known grid constraints provide stronger structural guidance than most language tasks, and bidirectional scans require access to a full input or candidate sequence. Applying them to generation would require a suitable formulation, such as iterative refinement of a candidate, and an evaluation of output quality and computational cost. This study provides no evidence of improved natural language generation.

Conclusion. In a single-seed study on Sudoku-Extreme, changing the scan orderings of a fixed Mamba-2 mixer changes exact match from 31.42% with row scans to 64.49% with row and column scans, and to 42.96% with additional box scans. An equal-parameter unidirectional configuration reaches 0.00%. Combining tied directions, two mixers, and learned positions reaches 80.83% without attention or a token-mixing MLP. These findings support scan geometry as a useful inductive bias in this setting, while showing that additional constraint-aligned directions need not help.

Multi-seed validation, evaluation on other tasks, and learned traversal policies are priorities for future work.

Limitations

Task scope and generalization. Completed experiments cover only Sudoku-Extreme, whose regular and known constraint geometry makes hand-designed scans straightforward. They do not establish transfer to other constraint-satisfaction problems, irregular structures, or natural language generation. The Maze-30 $\times$ 30-Hard study in Appendix D lacked a completed, validated control and provides no evidence of cross-task generalization. Evaluation uses 76,800 of the 422,786 held-out puzzles.

Single-seed evaluation. All reported completed runs use seed 0. Due to limited time and computational resources, we could not complete repeated training across multiple random seeds. We therefore cannot estimate run-to-run variance, provide confidence intervals across seeds, or establish the statistical significance of the reported differences. The 0.97-percentage-point gap to our hybrid baseline should not be interpreted as evidence of equivalence or a reliable advantage for either model. The larger scan-geometry differences also require multi-seed validation before they can be regarded as robust.

Controls and computational cost. Matching parameter counts does not match computation. Additional directions require additional scans, and the unidirectional and tied variants also alter mixer depth. Thus, the tied variant does not isolate the effect of sharing weights. The learned-position variant also rescales its input, so its gain does not isolate positional information alone. The proposed interference between box-major and row/column scans is a hypothesis rather than an established mechanism. We do not report matched-FLOP experiments, latency, or throughput measurements.

Hand-designed structure. The orderings use task structure known in advance. We do not learn traversal policies or provide a method for selecting scans when dependencies are unknown. Testing learned orderings and their interaction with aggregation remains future work.

References

- Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2019. [Deep equilibrium models](#). In *Advances in Neural Information Processing Systems*.
- Tri Dao and Albert Gu. 2024. [Transformers are SSMS: Generalized models and efficient algorithms through structured state space duality](#). In *International Conference on Machine Learning*.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2019. [Universal transformers](#). In *International Conference on Learning Representations*.
- Chunyuan Deng, Yizhe Zhang, Rui-Jie Zhu, Yuanyuan Xu, Jiarui Liu, T. S. Eugene Ng, and Hanjie Chen. 2026. [LT2: Linear-time looped transformers](#). *arXiv preprint arXiv:2605.20670*.
- Katie Everett, Lechao Xiao, Mitchell Wortsman, et al. 2024. [Scaling exponents across parameterizations and optimizers](#). In *International Conference on Machine Learning*.
- Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023. [Hungry hungry hippos: Towards language modeling with state space models](#). In *International Conference on Learning Representations*.
- Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. 2023. [Looped transformers as programmable computers](#). *arXiv preprint arXiv:2301.13196*.
- Alex Graves. 2016. [Adaptive computation time for recurrent neural networks](#). *arXiv preprint arXiv:1603.08983*.
- Albert Gu and Tri Dao. 2023. [Mamba: Linear-time sequence modeling with selective state spaces](#). *arXiv preprint arXiv:2312.00752*.
- Albert Gu, Karan Goel, and Christopher Ré. 2022. [Efficiently modeling long sequences with structured state spaces](#). In *International Conference on Learning Representations*.
- Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2024. [LocalMamba: Visual state space model with windowed selective scan](#). *arXiv preprint arXiv:2403.09338*.
- Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. 2026. [LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation](#). In *International Conference on Learning Representations*.
- Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. [Repeat after me: Transformers are better than state space models at copying](#). *arXiv preprint arXiv:2402.01032*.
- Alexia Jolicoeur-Martineau. 2025. [Less is more: Recursive reasoning with tiny networks](#). *arXiv preprint arXiv:2510.04871*.
- Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2014. [Deeply-supervised nets](#). *arXiv preprint arXiv:1409.5185*.
- Opher Lieber, Barak Lenz, Hofit Bata, et al. 2024. [Jamba: A hybrid transformer-Mamba language model](#). *arXiv preprint arXiv:2403.19887*.
- Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. 2024. [VMamba: Visual state space model](#). In *Advances in Neural Information Processing Systems*.
- William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. [The illusion of state in state-space models](#). In *International Conference on Machine Learning*.
- Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. 2018. [Recurrent relational networks](#). In *Advances in Neural Information Processing Systems*.
- Bo Peng, Eric Alcaide, Quentin Anthony, et al. 2023. [RWKV: Reinventing RNNs for the transformer era](#). *arXiv preprint arXiv:2305.13048*.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. [Reasoning with latent thoughts: On the power of looped transformers](#). In *International Conference on Learning Representations*.
- Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. 2026. [Probabilistic tiny recursive model](#). *arXiv preprint arXiv:2605.19943*.
- Noam Shazeer. 2020. [GLU variants improve transformer](#). *arXiv preprint arXiv:2002.05202*.
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. [RoFormer: Enhanced transformer with rotary position embedding](#). *arXiv preprint arXiv:2104.09864*.
- Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. [Retentive network: A successor to transformer for large language models](#). *arXiv preprint arXiv:2307.08621*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*.
- Roger Waleffe, Wonmin Byeon, Duncan Riach, et al. 2024. [An empirical study of Mamba-based language models](#). *arXiv preprint arXiv:2406.07887*.
- Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi-Yadkori. 2025. [Hierarchical reasoning model](#). *Preprint, arXiv:2506.21734*.

- Po-Wei Wang, Priya L. Donti, Bryan Wilder, and Zico Kolter. 2019. [SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver](#). In *International Conference on Machine Learning*.
- Wenlong Wang and Fergal Reid. 2026. [Tiny recursive reasoning with Mamba-2 attention hybrid](#). In *ICLR 2026 Workshop on Latent & Implicit Thinking*.
- Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J. Crowley. 2024. [PlainMamba: Improving non-hierarchical mamba in visual recognition](#). In *British Machine Vision Conference*.
- Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2023. [Gated linear attention transformers with hardware-efficient training](#). *arXiv preprint arXiv:2312.06635*.
- Zhenxuan Yu, Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. 2026. [CHASE: Cache-hole-adapted skip exit for looped state-space language models](#). *arXiv preprint arXiv:2607.10110*.
- Biao Zhang and Rico Sennrich. 2019. [Root mean square layer normalization](#). In *Advances in Neural Information Processing Systems*.
- Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. [Vision Mamba: Efficient visual representation learning with bidirectional state space model](#). In *International Conference on Machine Learning*.
- Rui-Jie Zhu, Zixuan Wang, Kai Hua, et al. 2025. [Scaling latent reasoning via looped language models](#). *arXiv preprint arXiv:2510.25741*.

A Extended Related Work

Recursive and latent reasoning. The Universal Transformer (Dehghani et al., 2019) repeatedly applies a shared layer with adaptive halting (Graves, 2016), while deep equilibrium models (Bai et al., 2019) formulate computation through a fixed point. Looped Transformers have been studied as programmable models (Giannou et al., 2023), as a way to increase reasoning depth (Saunshi et al., 2025), and for language modeling (Zhu et al., 2025), elastic depth (Jeddi et al., 2026), and linear-time computation (Deng et al., 2026). HRM uses two recurrent modules operating at different rates (Wang et al., 2025); TRM uses a single small operator and two states (Jolicoeur-Martineau, 2025), with a probabilistic extension studied by Sghaier et al. (2026). Recurrent relational networks (Palm et al., 2018) and SATNet (Wang et al., 2019) also solve Sudoku iteratively but explicitly encode relational or logical structure.

Our focus is the traversal used inside a shared recurrent operator.

State-space operators and hybrid models.

Structured and selective state-space models (Gu et al., 2022; Gu and Dao, 2023; Dao and Gu, 2024), together with related efficient sequence models (Fu et al., 2023; Peng et al., 2023; Sun et al., 2023; Yang et al., 2023), offer alternatives to full self-attention. Their recurrent structure can limit copying and state tracking (Jelassi et al., 2024; Merrill et al., 2024). Empirical studies and hybrid architectures combine recurrence with attention to address such limitations (Waleffe et al., 2024; Lieber et al., 2024). Yu et al. (2026) study looped state-space language models, while Deng et al. (2026) study linear-time looped Transformers. Closest to our setting, Wang and Reid (2026) use Mamba-2 with attention or an MLP-t inside TRM. We examine whether multiple spatial traversals can provide effective mixing without those additional components on Sudoku.

Multi-directional scanning. Vim (Zhu et al., 2024) scans a patch sequence in both directions, VMamba (Liu et al., 2024) uses a four-way cross-scan, LocalMamba (Huang et al., 2024) combines windowed scans with direction search, and PlainMamba (Yang et al., 2024) uses a continuous traversal to preserve spatial adjacency. Our scan mechanism follows this line of work. The experimental setting differs: a shared operator is reused 21 times per supervision step, and the target requires satisfaction of explicit constraints across an entire board. The observed decline when box-major scans are added motivates studying interactions among scan orderings, rather than assuming that more structural coverage improves performance.

B Parameter Accounting

Each configuration in Table 2 contains two independently parameterized Mamba-2 modules, arranged as shown in Table 1. At hidden size 512, state size 128, expansion 2, and head dimension 64, each module has 1,719,600 parameters. The pair contributes 3,439,200. The shared components are a token embedding (5,632), an output head (5,632), a halting head (1,026), and a channel MLP (2,359,296), giving 5,810,786 dense parameters. The separately optimized puzzle embedding is a 512-parameter vector stored as a buffer

for a single generic puzzle identifier. It is padded with fifteen zero vectors to form the 16-position prefix before positional embeddings are added. It is identical in size across configurations and is excluded from the reported dense counts; including it gives 5,811,298 trainable values. The two fixed initial-state buffers add 1,024 stored values but no trainable parameters.

Learned absolute position embeddings add $97 \times 512 = 49,664$ parameters, giving 5,860,450 dense parameters. Rotary position embeddings operate in the attention component; the scan-only variants instead receive position information implicitly through traversal, with explicit learned embeddings in the positional variant. The baseline’s MLP-t uses a gated projection and a down-projection along the 97-position token axis, adding 148,992 parameters and giving 5,959,778 dense parameters. Its residual normalization also operates on the transposed token axis. Including the common puzzle embedding gives totals of 5,860,962 and 5,960,290, respectively.

C Training Configuration

Table 4 lists the shared data, recursion, and optimization settings for all completed Sudoku-Extreme runs. Operator differences are given in Tables 1 and 3. The attention-hybrid result cited from prior work is not a run under this protocol.

D Maze-30×30-Hard: An Incomplete Control

We attempted Maze-30×30-Hard to test whether the Sudoku findings extend to another structured task. Wang and Reid (2026) report 80.6% exact match for a Mamba-2/attention hybrid, 60.8% for their Transformer baseline, and 0.0% for both MLP-t variants. The TRM paper reports 85.3% for its attention model (Jolicoeur-Martineau, 2025). This difference between published baselines makes a validated control under our own protocol especially important.

We prepared the released maze-30x30-hard-1k dataset: 1,000 training mazes, each a 30×30 grid flattened into 900 positions, with a six-token vocabulary and one generic puzzle identifier. We used the scaffold, optimizer, and halting settings in Table 4, except for four inner cycles instead of six. Row and column scans extend directly to this grid. The additional tiled ordering groups cells into 5×5

Setting	Value
<i>Data</i>	
Training puzzles	1,000
Augmentations / puzzle	1,000
Test puzzles	422,786
Puzzles per evaluation	76,800 (first 100 batches)
Sequence length	97 (16 prefix + 81 cells)
Vocabulary	11
<i>Scaffold</i>	
Hidden size	512
MLP expansion	4 (SwiGLU inner 1,536)
Operator blocks	1
Outer / inner cycles	3 / 6
Training refinements	up to 16
Evaluation refinements	16 (fixed)
Halt exploration probability	0.1
Prefix positions	16 (one vector, zero-padded)
Activation precision	bf16
<i>Mamba-2</i>	
State size	128
Convolution width	4
Expansion	2
Head dimension	64
Groups / chunk size	1 / 256
Parameter precision	float32
<i>Optimization</i>	
Dense optimizer	AdamATan2
Learning rate	10^{-4} , constant after warmup
Warmup steps	2,000
(β_1, β_2)	(0.9, 0.95)
Weight decay	1.0
Gradient clipping	1.0
Embedding optimizer	sign-SGD, lr 10^{-4} , wd 1.0
Global batch size	768
Epochs / updates	50,000 / 65,100
Evaluation every	5,000 epochs (6,510 steps)
EMA rate	0.999 (evaluated on EMA)
Seed	0

Table 4: Training configuration, shared by every Sudoku-Extreme run in this paper. Each epoch samples one instance per puzzle group. Dropping the incomplete batch in each 5,000-epoch iteration gives $10 \lfloor 5,000,000 / 768 \rfloor = 65,100$ updates; the configured budget is 65,104.

regions; unlike Sudoku boxes, these regions encode locality rather than a task constraint.

Fourteen runs were launched across the operator variants, and all recorded 0.00% exact match at their available evaluations. Separately, the original two-layer TRM control reached 2.80% at step 12,967 before the training job terminated. This was approximately one fifth of the planned 65,000-step budget. The control was not restarted because of time and compute constraints.

These incomplete runs do not support a comparison of architectures on Maze. The interrupted control does not validate the training and evaluation pipeline, and zero exact match could reflect incomplete optimization, a pipeline problem, or an architectural limitation. We cannot distinguish these explanations and do not claim that recursive Mamba-2 fails on this task.

An unresolved issue is operator depth. The Maze hybrid used one block to approximate the published 6.86M parameter budget, whereas the original TRM control used two blocks. The recorded counts are 6,822,914 for the two-block Transformer, 6,854,242 for the one-block Mamba-2 hybrid, and 13,701,314 for the two-block hybrid. At these widths, matching parameter count does not also match block depth. Whether this difference affects Maze performance remains untested.