RESHAPE AND RECUR: IMPROVING SSMS WITH INPUT RESHAPING AND DEPTH RECURRENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

State Space Models (SSMs) are increasingly deployed in the Edge because they offer, at comparable performance, a smaller memory/training/inference footprint, compared to Large Language Models (LLMs). These three advantages are a direct consequence of the time recurrence inherent in the SSMs architecture. Here, we further improve this recurrent architecture by positively answering two previously underexplored, orthogonal questions: (1) Can we reduce SSMs memory-footprint without any performance penalty, by also employing depth recurrence? (2) Can we increase SSMs performance by using a fixed and consistent time-granularity across all tasks? The first question is somewhat unexpected, given that SSMs are already recurrent. However, the orthogonal depth recurrence further decreases SSMs memory footprint. We show that a looped SSM with k parameters adaptively iterated M times, achieves a performance comparable to a standard SSM with $k \cdot L$ independent parameters, where $M \leq L$. The second question is also unexpected given the time-recurrent nature of the SSMs architecture. However, it makes perfect sense for the time-parallel training of SSMs on the entire input sequence. We show that concatenating time steps for lower-dimensional sequence elements, or flattening and re-chunking the joint feature-time dimension for high-dimensional ones, can improve the baseline by enhancing the way information is presented to the model. Our results for both extensions lead to consistent benefits across four representative SSM architectures: LRU, S5, LinOSS, LrcSSM.

1 INTRODUCTION

State Space Models (SSMs) have emerged as a compelling alternative to (transformer-based) Large Language Models (LLMs), for tasks requiring efficient processing of long sequences (Gu et al., 2022; Hasani et al., 2023; Orvieto et al., 2023). By maintaining a compact recurrent state updated across the sequence dimension, SSMs have a very small memory footprint, linear-time-learning and constant-time-inference complexity, respectively, while retaining strong empirical performance. These desirable properties of the SSMs have been demonstrated across a wide range of domains, including language, audio, and time series, and have lead to their increased use in the Edge (Gu & Dao, 2024; Rusch & Rus, 2025; Farsang & Grosu, 2026; Mercedes-Benz North America, 2026).

In this paper, we demonstrate how to further improve SSMs accuracy and reduce SSMs memory footprint, by positively answering two previously underexplored and orthogonal questions, regarding the recurrent architecture of SSMs: (1) Can we still reduce SSMs memory-footprint without any performance penalty by also taking advantage of layers-depth recurrence? (2) Can we still increase SSMs performance by using a fixed and consistent time-granularity across all tasks?

Asking the first question is not trivial, given that SSMs are already recurrent in the time dimension. This recurrence is central to SSMs and implicitly constrains how researchers think about scaling them up. The general design pattern is simple, usually stacking L independent SSM layers, each with its own parameters, and letting depth provide representational diversity. However, layers-depth recurrence is completely orthogonal to time recurrence, and further decreases SSMs memory footprint. We show that a looped SSM with k parameters adaptively iterated M times, achieves a performance comparable to a standard SSM with $k \cdot L$ independent parameters, where $M \leq L$.

Asking the second question is also not trivial, given that (nonlinear) recurrent neural networks (RNNs) are the original roots of SSMs, and RNNs were trained in a sequential fashion. In RNNs, ac-

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

quisition and training have the same granularity, and this is a property of the incrementally-processed input sequence. However in SSMs, training happens in parallel, on the entire input sequence. Hence, it makes sense to ask if reshaping the input sequence in chunks that are consistent across all tasks considered can improve accuracy? After all, the SSMs see the entire sequence anyway, and consistency might be beneficial. Indeed, we show that concatenating time steps for lower-dimensional sequence elements, or flattening and re-chunking the time steps for high-dimensional elements, can considerably improve the baseline, by enhancing information presentation to the model.

The interaction between input-dimensionality, sequence-length, and model-capacity is still poorly understood in SSMs, and this work sheds some light on this interaction. We show that depth recurrence decreases their memory footprint without any penalty, and input reshaping positively affects SSMs performance. Moreover, the combination of both further improves SSM accuracy. Our experiments were conducted on four representative SSM baselines: LRU (Orvieto et al., 2023), S5 (Smith et al., 2023), LinOSS (Rusch & Rus, 2025), and LrcSSM (Farsang & Grosu, 2026), across six time series benchmarks spanning input dimensionalities between 6 to 63, and sequence lengths between 400 to 50,000. Our experimental results show that both extensions lead to consistent benefits.

Our main contributions in this paper are as follows:

- *We systematically study depth-recurrence across representative SSM architectures*, showing that recurrence across depth provides a distinct design axis from their native sequence-level recurrence. We evaluate three parameter-sharing strategies under two supervision schemes (final-layer and block-wise loss) across four architectures and six benchmarks.
- *We demonstrate the use of adaptive inference*, by chosing the number of depth iterations at inference time according to the change between successive predictions, allowing depth-recurrent SSMs to automatically adjust their computational depth without retraining.
- *We show that depth-recurrent SSMs achieve comparable performance to independently parameterized models* while using substantially fewer parameters. A block with k parameters iterated L times uses as little as $1/L$ of the unique SSM-block parameters of a model with L independent blocks. Parameter sharing across depth is thus a beneficial inductive bias.
- *We introduce input reshaping as a complementary design axis for SSMs*, studying how changing the information density (and thus sequence length) presented to the model affects SSMs performance across different tasks with differnt input-dimensionality regimes.

2 Depth Recurrence in SSMs

SSMs already employ a recurrent state transition along the sequence dimension. Here, we introduce a second recurrence along layers depth: rather than applying a sequence of independently parameterized blocks, SSM blocks are periodically reused along depth. Thus, depth recurrence is not simply a reduction in parameter count through weight sharing; it turns the depth dimension itself into an iterative computation. Weight sharing follows from this recurrent construction.

Let $\mathbf{z}_0 \in \mathbb{R}^{T\times d}$ denote the input-sequence representation, where T is the sequence length, and d is the dimension its elements, and let f_{θ_ℓ} denote an SSM block at depth ℓ. A depth-L SSM computes:

$$\mathbf{z}_\ell = f_{\theta_\ell}(\mathbf{z}_{\ell-1}), \qquad \ell = 1, \dots, L. \tag{1}$$

Depending on the architecture, the time-dependent state recurrence within f_{θ_ℓ} may be linear (LRU and S5), based on forced linear second-order ODEs (LinOSS), or nonlinear (LrcSSM).

Independent layers. In a classic depth-L SSM, every depth step has independent parameters

$$\theta_1, \theta_2, \dots, \theta_L, \tag{2}$$

giving a total parameter count of kL when each block contains k parameters.

Periodic Depth Recurrence. We define the *recurrence period* m as the number of unique SSM blocks before the depth-wise computation repeats. For $m \in \mathrm{Div}(L)$, the parameters at depth ℓ follow

$$\theta_\ell = \theta_{((\ell-1) \bmod m)+1}, \qquad \ell = 1, \dots, L. \tag{3}$$

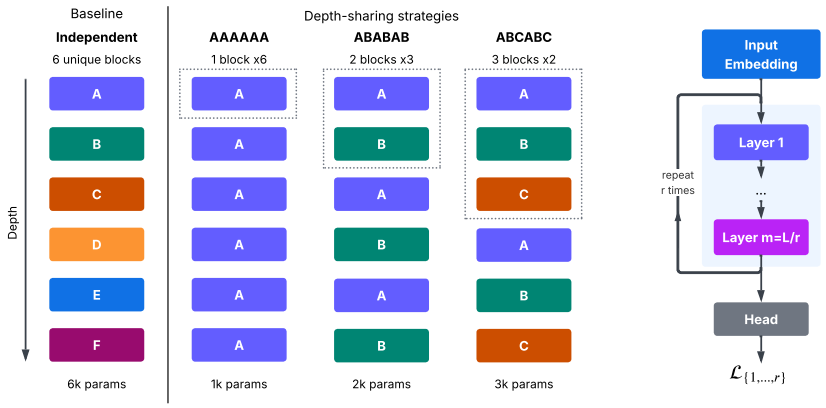


Figure 1: Depth-recurrent SSM configurations for $L = 6$. The baseline ABCDEF has recurrence period $m = 6$, while AAAAAA, ABABAB, and ABCABC correspond to $m = 1$, $m = 2$, and $m = 3$, respectively. We evaluate each depth-recurrent configuration using final or block-wise supervision.

The resulting model contains m unique blocks, each applied $r = L/m$ times. As a consequence the model has km parameters. The full depth-recurrence model corresponds to $m = 1$, while the independently parameterized model corresponds to $m = L$.

For a fixed depth L, the valid periodic configurations are given by the divisors of L. When $L = 6$,

$$m \in \mathrm{Div}(6) = 1, 2, 3, 6, \tag{4}$$

which leads to the following depth-recurrent model configurations:

$$m = 1 \to \texttt{AAAAAA}, \quad m = 2 \to \texttt{ABABAB}, \quad m = 3 \to \texttt{ABCABC}, \quad m = 6 \to \texttt{ABCDEF}. \tag{5}$$

Each letter denotes an independently parameterized SSM block. Thus, these configurations exhaust the periodic depth-recurrence design space for $L = 6$, from a single block recurrently applied across all depth steps to the standard six-block independent model. Patterns whose period does not divide L, such as ABCDAB, contain an incomplete final period and lie outside this periodic family.

2.1 Supervision and Adaptive Inference

For a recurrence period m, let $r = L/m$ denote the number of repetitions and let $\mathbf{h}^{(j)}$ denote the representation after the j-th complete period, $j = 1, \dots, r$. Given a prediction head g_ϕ and target y, we consider the following two supervision strategies.

Final supervision. The loss is computed only after the final depth repetition:

$$\mathcal{L}^{\text{final}} = \mathcal{L}\left(g_\phi(\mathbf{h}^{(r)}), y\right). \tag{6}$$

Block-wise supervision. The same prediction head is applied after every complete repetition:

$$\mathcal{L}^{\text{block}} = \frac{1}{r}\sum_{j=1}^{r} \mathcal{L}\left(g_\phi(\mathbf{h}^{(j)}), y\right). \tag{7}$$

The prediction head g_ϕ is shared across all supervision points. Block-wise supervision therefore provides intermediate training signals along the depth-recurrence trajectory without introducing additional prediction heads. The choice of supervision is independent of the recurrence period m.

Adaptive inference. Depth recurrence also permits the number of iterations to be varied at inference time. Let $\mathbf{p}^{(j)}$ denote the prediction after repetition j. We terminate the depth iteration when the relative change between consecutive predictions falls below a threshold ϵ:

$$\frac{\left|\mathbf{p}^{(j)} - \mathbf{p}^{(j-1)}\right|_2}{\left(\left|\mathbf{p}^{(j)}\right|_2 + \left|\mathbf{p}^{(j-1)}\right|_2\right)/2} < \epsilon. \tag{8}$$

The adaptive stopping criterion therefore allows the depth of the inference to adapt, based on the normalized change between two successive predictions.

2.2 Expressivity of Periodic Depth Recurrence

For fixed L and per-block parameter budget k, let $\mathcal{H}_m$ denote the hypothesis class induced by recurrence period m. Given this notation, the independently parameterized model is therefore $\mathcal{H}_L$, while the fully depth-recurrent model is $\mathcal{H}_1$.

Proposition 1 (Monotone containment under divisibility) *Let m and m' be divisors of L such that $m \in \mathrm{Div}(m')$. Then the following holds:*

$$\mathcal{H}_m \subseteq \mathcal{H}_{m'}. \tag{9}$$

Consequently, for every $m \in \mathrm{Div}(L)$, one has that:

$$\mathcal{H}_1 \subseteq \mathcal{H}_m \subseteq \mathcal{H}_L. \tag{10}$$

The containment follows because a period-m' model can reproduce any period-m model, whenever $m \in \mathrm{Div}(m')$, by periodically tying its parameters according to Equation 3. We provide the full proof of this containment in Appendix A.1.

For $L = 6$, this gives the following parameterized model containments:

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_6, \qquad \mathcal{H}_1 \subseteq \mathcal{H}_3 \subseteq \mathcal{H}_6, \tag{11}$$

However, $\mathcal{H}_2$ and $\mathcal{H}_3$ are not ordered, because neither $2 \in \mathrm{Div}(3)$ nor $3 \in \mathrm{Div}(2)$. Thus, every depth-recurrent configuration considered in our experiments is contained in the independently parameterized baseline. Matching or outperforming that baseline cannot be therefore explained by a greater expressivity of the depth-recurrent model. Instead, it indicates a beneficial inductive bias which constrains the recurrent computation to use the same parameters across depth.

3 Input Reshaping in SSMs

Our second design axis modifies the input-sequence (input time-series) representation, independently of the recurrence period m. We use a (desirably optimal) concentration factor c, to control how many scalar input values are grouped into an input vector at every time step. This trades sequence length for per-step information density. Figure 6 depicts the two versions described below.

Low-dimensional inputs. For an input $\mathbf{X} \in \mathbb{R}^{T \times d}$, we choose c as a multiple of d and concatenate c/d consecutive timesteps along the feature dimension. The resulting sequence has following shape:

$$(T, d) \longrightarrow \left(\left\lceil \frac{Td}{c} \right\rceil, c \right), \qquad c > d, \quad d \in \mathrm{Div}(c), \tag{12}$$

with zero-padding where necessary. This reshaping thus concatenates the complete feature vectors at each original time steps, and shortens the sequence by approximately a factor of c/d.

High-dimensional inputs. For high-dimensional inputs $\mathbf{X} \in \mathbb{R}^{T \times D}$, we first flatten the joint feature-and-time sequence along its scalar values, and re-chunk it into vectors of width c:

$$(T, D) \longrightarrow \left(\left\lceil \frac{TD}{c} \right\rceil, c \right) \qquad c < D. \tag{13}$$

Unlike temporal concatenation, this operation deliberately allows feature and temporal identities to mix within a single input vector. Zero-padding is applied when TD is not divisible by c.

Depth recurrence and input reshaping therefore modify two distinct aspects of the recurrent SSM model: the depth-recurrence period m controls how computation is repeated across depth, while the concentration factor c controls how information is presented along the sequence (or time) dimension. In the next section, we study their effects both independently and in combination.

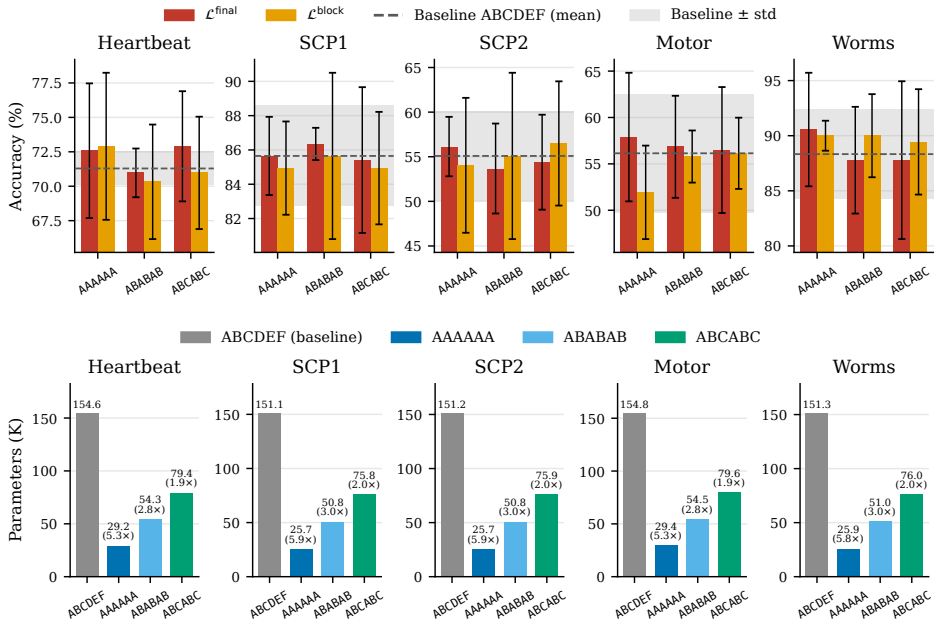


Figure 2: Classification accuracy (%) across datasets for different LRU depth-recurrence configurations, averaged over 5 seeds (top), with the corresponding parameter reductions (bottom). Depth-recurrent models achieve comparable performance with substantially fewer parameters.

4 Empirical Results

4.1 Datasets and Experimental Setup

We evaluate periodic depth recurrence and input reshaping both on multivariate time-series classification datasets from the UEA Multivariate Time Series Classification Archive Bagnall et al. (2018) and on the heart-rate estimation regression task of PPG-DaLiA Reiss et al. (2019). These benchmarks span a wide range of sequence lengths and input dimensionalities and are commonly used to evaluate the performance of sequence and SSM models (Walker et al., 2024a; Moreno-Pino et al., 2024; Nzoyem et al., 2025; Rusch & Rus, 2025; Boyer et al., 2025; Pourcel & Ernoult, 2026; Karuvally et al., 2026). We use fixed 70/15/15 train/validation/test splits generated with the same predefined random seeds. Dataset details are provided in Table 1 in Appendix A.2.1.

We evaluate the following representative SSM models: S5 (Smith et al., 2023), LRU (Orvieto et al., 2023), LinOSS (Rusch & Rus, 2025), and LrcSSM (Farsang & Grosu, 2026). We fix the architecture across models, and recurrence periods, using state size 64, hidden dimension 64, and depth $L = 6$, which gives the periodic recurrence periods $m \in \{1, 2, 3, 6\}$. We tune only the learning rate over $\{10^{-3}, 10^{-4}, 10^{-5}\}$. Further dataset, model, and training details are provided in Appendix A.2.2.

4.2 Depth Recurrence

We first evaluate depth recurrence without input reshaping, by varying the recurrence period over $m \in \{1, 2, 3\}$ and comparing against the independently parameterized $m = 6$ baseline. By Proposition 2, each depth-recurrent hypothesis class is contained in that of the independent baseline. Thus, matching or exceeding the baseline cannot be attributed to greater expressivity.

Figure 2 shows that depth-recurrent models frequently remain close to the independent baseline despite substantial parameter reductions. In particular, $m = 1$ (AAAAAA) reduces the total parameter count by 5.3–5.9× while remaining competitive across several datasets. Across recurrence periods, neither supervision strategy uniformly dominates. Complete numerical results across all classification datasets and models are reported in Table 2 in the Appendix.

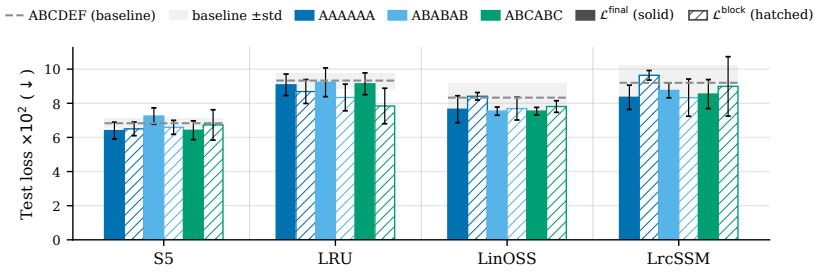


Figure 3: Test loss ($\times 10^2$, lower is better) on PPG-DaLiA across S5, LRU, LinOSS, and LrcSSM depth-recurrence configurations, averaged over 5 seeds. Bars show AAAAAA, ABABAB, and ABCABC with $\mathcal{L}^{\text{final}}$ (solid) and $\mathcal{L}^{\text{block}}$ (hatched). The dashed line and band denote the ABCDEF baseline mean ± 1 std; error bars denote ± 1 std.

To verify that this behavior is not specific to classification, we additionally evaluate depth recurrence on the PPG-DaLiA regression task. Figure 3 shows that all depth-recurrent configurations match or outperform the accuracy of the corresponding independently-parameterized SSM baselines, indicating that the observed behavior extends beyond classification tasks.

4.3 Adaptive Depth at Inference

So far, we have evaluated models at their fixed training depth of $L = 6$. We next investigate whether the same trained model can adapt its computational depth at inference time, considering both fewer than six recurrent applications and extrapolation beyond the training depth, up to a maximum of 12 applications. We evaluate adaptive depth for the two most parameter-efficient depth-recurrent configurations, AAAAAA and ABABAB, using the LRU models. The results are reported in Figure 4, for AAAAAA, and in Table 8 of the Appendix, for ABABAB.

At inference time, we repeatedly apply the trained recurrent block and stop when the relative change between consecutive predictions falls below the threshold defined in Section 2.1, or when the maximum number of applications is reached. It is important to note that this relatively simple stopping criterion does not imply convergence of the hidden states or predictions. It merely detects when successive predictions differ by less than the prescribed stopping threshold.

For the AAAAAA model, the stopping criterion typically selects between 2 and 5 repetitions, compared with the fixed 6 repetitions used during training, while maintaining classification performance. Since fewer repetitions require less computation, this results in a substantial reduction in GFLOPs, as shown in Figure 4. These results show that depth-recurrent SSMs support adaptive computation at inference time: the same trained model can use a variable number of recurrent applications based on the relative change between successive predictions, without requiring retraining for each depth.

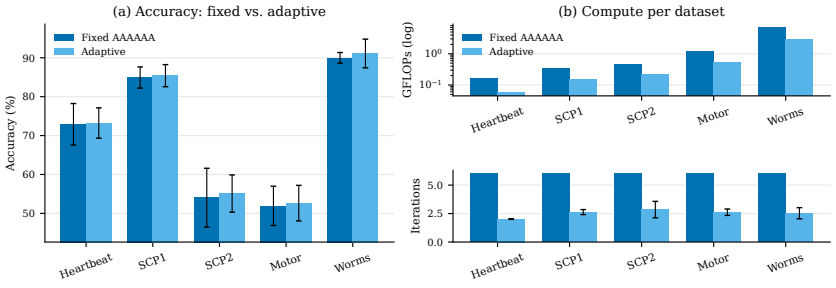
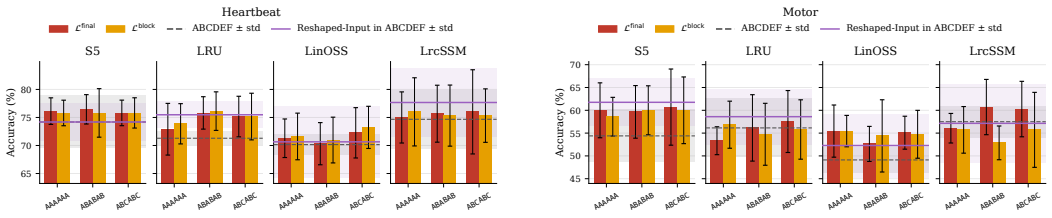


Figure 4: Adaptive inference versus fixed 6-iteration AAAAAA across five benchmarks. (a) Adaptive inference maintains comparable performance. (b) It reduces GFLOPs by typically stopping after 2-3 iterations (mean $\pm$ std over five seeds), compared with the fixed budget of six.



(a) Heartbeat (b) MotorImagery

Figure 5: Classification accuracy (%) on Heartbeat and MotorImagery with input reshaping and depth recurrence. Bars show AAAAAA, ABABAB, and ABCABC with $\mathcal{L}^{\text{final}}$ (red) and $\mathcal{L}^{\text{block}}$ (orange). Lines show the original (grey dashed) and reshaped-input (purple) ABCDEF baselines. Bands and error bars denote ± 1 std over 5 seeds.

4.4 Input Reshaping

We next isolate the effect of input reshaping within the baseline architecture, without any depth recurrence. This provides a complementary way to adapt the input representation, without tuning architectural dimensions such as the hidden size, state size, or number of layers.

Figure 5 compares the original ABCDEF baseline (grey dashed line) with the corresponding reshaped-input ABCDEF baseline (purple solid line). Note that input reshaping alone, considerably improves the mean of several models, most notably for LRUs on the Heartbeat dataset ($71.3\% \rightarrow 75.5\%$), and for S5 on the MotorImagery dataset ($54.4\% \rightarrow 61.8\%$).

4.5 Combining Input Reshaping and Depth Recurrence

We next apply depth recurrence on top of the reshaped inputs. Using the same comparison in Figure 5, the depth-recurrent configurations with reshaped inputs consistently improve over their counterparts using the original inputs. On Heartbeat, LRU reaches 76.1% with the combined approach, compared with 72.9% for the corresponding depth-recurrent configurations without reshaping. On MotorImagery, S5 and LrcSSM reach 60.7% and 60.7%, respectively, compared with 55.8% and 58.2%, respectively, with depth recurrence alone. These results show that the benefits of input reshaping are retained when combined with depth recurrence.

Complete numerical results for this experiment are reported in Table 5. Medium-dimensional datasets (SCP1, SCP2, and Worms) show no substantial gains from reshaping, suggesting that reshaping is most beneficial when the original input representation is poorly matched to the fixed architecture (state-size and hidden dimension). Results for these datasets are reported in Tables **??** and **??** in the Appendix. Additional input reshaping experiments are provided in Appendix A.4.

Input reshaping can also be applied during online inference, with a corresponding change in the input-output frequency. For lower-dimensional inputs, concatenating consecutive timesteps requires buffering multiple observations before forming a model input, introducing a delay in the timesteps. For high-dimensional inputs, rechunking can instead map a single original timestep to multiple model inputs, from which predictions can be produced at the corresponding model steps.

Taken together, these results suggest that reshaping and depth-recursion are complementary: reshaping lifts the baseline by improving how information is presented to the model, while depth-recursion can extract additional value from the resulting representation with a reduced parameter count.

4.6 Ablations on Parameter-Matched Setups and Damped Iterative Updates

We further investigate whether the benefits of depth recurrence can be explained by the reduced parameter count alone. We focus on LRUs, which showed the most consistent performance under depth recurrence. Figure 7 in Appendix A.5 shows that depth-recurrent models can substantially outperform parameter-matched single-block baselines.

We also consider a more conservative update rule based on damped iterative updates following the Krasnosel'skii-Mann iteration (Krasnosel'skii, 1955; Mann, 1953). This update rule preserves part

of the previous state while incorporating the newly computed update with a smaller weight. We find that damping does not lead to substantially different results. The full results and analysis the Krasnosel'skii-Mann iteration are provided in Appendix A.7.

5 Discussion and Future Work

On the mismatch between model input and data structure. A recurring observation motivating this work is that standard SSM pipelines typically process one input vector per time step, regardless of the dimensionality of that vector. For low-dimensional inputs (small d), each time step contains relatively little information, resulting in long sequences of small input vectors. For high-dimensional inputs (large D), each time step instead contains a large number of features that must be processed jointly. Input reshaping provides a simple way to modify this tradeoff through the concentration factor c, without changing the underlying architecture. Our results show clear gains on the high-dimensional Heartbeat and MotorImagery datasets, while the medium-dimensional datasets benefit less. This suggests that the effectiveness of reshaping strongly depends on the mismatch between the state size of the model and the input dimension of the sequence elements.

On the benefits of depth recurrence. Across the main experiments, depth-recurrent SSMs frequently remain competitive with independently parameterized baselines despite using substantially fewer parameters. By Proposition 2, the corresponding depth-recurrent hypothesis classes are contained in that of the independent baseline, so these results cannot be explained by greater expressivity. Moreover, the parameter-matched ablation in Appendix A.5 shows that the observed gains cannot be explained by parameter reduction alone: depth-recurrent LRUs can outperform both the full independent model and parameter-matched single-block baselines. Together, these results suggest that a repeated computation with shared parameters across depth, is most likely a very useful inductive bias, rather than simply a parameter-saving mechanism.

Adaptive computation. Depth recurrence also enables the amount of computation to vary at inference time without retraining. Our adaptive inference experiments show that the prediction-change stopping criterion often terminates before the six applications used during training, while maintaining comparable performance and reducing GFLOPs. Importantly, this stopping criterion measures only the relative change between successive predictions and does not imply convergence. This provides a practical additional benefit of treating depth as an iterative computation.

Architecture-specific behavior. The gains from depth recurrence are not uniform across architectures. LRU benefits most consistently, while LinOSS and LrcSSM show a more mixed picture, including drops on some benchmarks. We hypothesize that this reflects an interaction between the recurrence structure of each architecture and the inductive bias introduced by repeated parameter sharing, but a principled explanation remains open. Understanding when and why this interaction is favorable is an important direction for future work, as we note in Remark 4.

Limitations and future work. Our experiments focused on time-series classification and regression tasks, and extending the analysis to other sequence modeling tasks remains an important next step. We also fix the training depth to $L = 6$ and treat the concentration factor c as a tuned scalar hyperparameter. Future work could characterize when depth recurrence is beneficial, investigate how the appropriate concentration factor depends on the structure of the input, and study training directly with variable depth. Finally, reweighting the intermediate loss terms in $\mathcal{L}^{\text{block}}$ could provide a curriculum-like training signal and is another direction worth exploring.

6 Related Work

State Space Models. Structured State Space Models (SSMs) emerged as a scalable alternative to attention-based Transformers for long-range sequence modeling. S4 (Gu et al., 2022) introduced a parameterization of linear SSMs via the HiPPO framework, achieving strong performance on the Long-Range-Arena benchmark. S5 (Smith et al., 2023) streamlined this design by replacing the independent SISO systems with a single MIMO SSM solved via parallel scan. The Linear Recurrent Unit (LRU) (Orvieto et al., 2023) demonstrated that careful initialization and normalization of a simple diagonal linear RNN, is sufficient to match deep SSMs accuracy, without any continuous-time or HiPPO motivation. LinOSS (Rusch & Rus, 2025) used forced second-order linear ODE dynam-

ics with strong performance, inspired by cortical oscillations, and most recently, LrcSSMs (Farsang & Grosu, 2026) extended SSMs to biologically plausible nonlinear recurrent dynamics, forcing a diagonal Jacobian structure allowing parallel training while retaining nonlinear expressivity.

The main focus of all these previous works, was on the design of increasingly powerful individual SSM layers, stacked independently. In this work instead, we study what happens when these layers are reused across depth. There is no previous work on SSMs systematically addresing this topic.

Depth Recurrence. In feed-forward neural networks such as transformers, the use of layer-depth recurrence occurred as a necessity, to reducing the sheer size of these networks. (Dehghani et al., 2019) first showed how to apply the same block repeatedly and adaptively at computation time. This work was considerably revived in test-time scaling. (Zhu et al., 2025) scaled this approach to 2.6B parameters matching a 8B standard transformer, and Saunshi et al. (2025) showed that looped transformers can match much deeper non-looped models on reasoning tasks. Geiping et al. (2026) trained a 3.5B-parameter model with a fixed recurrent block iterated up to 32 times at inference, showing that effective depth can be scaled independently of parameter count. (Prairie et al., 2026) addressed training instability in looped models, established scaling laws for stable looped architectures, and (Pappone et al., 2025; Blayney et al., 2026) studied the geometry of iterates inside looped blocks, finding that updates perform fine-grained latent refinements rather than coarse directional pushes.

In time-recurrent neural networks such as SSMs, depth recurrence was largely ignored. Here, we thoroughly study adaptive depth-recurrence explicitly, as an architectural design choice for SSMs, by systematically comparing parameter-shared and independently parameterized depth across multiple SSM families. On a related but different line of work, Schöne et al. (2025) iterates Mamba-2 blocks in depth towards a fixed-point, in order to obtain an implicit language model. Recurrent Highway Networks Zilly et al. (2017), repeatedly apply a transformation within a single recurrent transition, inside a layer. In contrast, our approach replaces a stack of independent SSM layers with repeated applications of the same SSM block, introducing recurrence along the depth dimension.

Input Reshaping. Reshaping input data before feeding it into a sequence model, can significantly affect model performance. However, this topic has remained quite underexplored relative to architectural design. If the input is an image, partitioning this into patches is not only necessary, but also relatively intuitive, as the image has no time dimension. For example, ViT (Dosovitskiy et al., 2021) partitions images into patches that mix spatial positions within each token. Subsequent work further explores varying patch sizes and hierarchical structures (Beyer et al., 2023; Liu et al., 2021).

If the input is a time-series however, the most natural thing to do is to use as patches the vectors available at each moment of time. This is precisely the approach used by all SSMs so far. In contrast, our novel reshaping strategy is much less obvious from a time-series perspective. It concatenates low-dimensional sequence vectors across time-steps, and flattens and re-partitions high-dimensional sequence vectors across time-steps. This way it achieves an optimal vector size per sequence element, that is presented to the model. This interaction between original vector size, sequence length, and optimal vector size, has not been studied in SSMs so far. Related ideas have been explored for time series in PatchTST (Nie et al., 2022), which aggregates consecutive time-step patches into a single patch. However, PatchTST did not consider splitting a time step in several ones, with according patches. Channel-mixing versus channel-independence has been studied as an architectural choice in forecasting models (Han et al., 2024), but this is less related to our current work.

7 Conclusion

We study two complementary design axes for SSMs: recurrence across depth and the structure of the input representation. Across multiple SSM architectures, depth recurrence can preserve or improve performance while substantially reducing parameter count, despite operating within a hypothesis class contained in that of the independently parameterized baseline. It also enables adaptive inference, allowing the same trained model to use fewer depth iterations when successive predictions change only marginally. Independently, input reshaping can improve performance without modifying the underlying architecture, and combining the two can provide further gains. Together, these results show that the way an SSM reuses computation across depth, and how its input sequence is represented, are important design choices alongside the recurrent dynamics of the SSM itself.

AI USE STATEMENT

In this work, we used generative AI tools to assist with parts of the code development, with parts of the verification of theoretical claims, and the improvement of the manuscript's readability. All AI-assisted content was reviewed by the authors, and generated code was independently verified. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide the code used to reproduce our experiments as part of the supplementary materials. Detailed descriptions of the datasets, model configurations, training setup, and hyperparameters, with complete numerical results and additional ablations, are provided in the Appendix.

REFERENCES

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14496–14506, 2023.

Niels Birbaumer, Nimr Ghanayim, Thilo Hinterberger, Iver Iversen, Boris Kotchoubey, Andrea Kübler, Juri Perelmouter, Edward Taub, and Herta Flor. A spelling device for the paralysed. *Nature*, 398(6725):297–298, 1999.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Jared Boyer, T Konstantin Rusch, and Daniela Rus. Learning to dissipate energy in oscillatory state-space models. *arXiv preprint arXiv:2505.12171*, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Mónika Farsang and Radu Grosu. Parallelization of non-linear state-space models: Scaling up liquid-resistance liquid-capacitance networks for efficient sequence modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ENYvdnyhLl.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=S3GhJooWIC.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):7129–7142, 2024.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2023.

Yoo-Min Jung and Leekyung Kim. MambaSL: Exploring single-layer mamba for time series classification. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YDl4vqQqGP.

Arjun Karuvally, Franz Nowak, T. Anderson Keller, Carmen Amo Alonso, Terrence Sejnowski, and Hava T Siegelmann. Bridging expressivity and scalability with adaptive unitary SSMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=s4zitEu2R8.

Mark Aleksandrovich Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi matematicheskikh nauk*, 10(1):123–127, 1955.

Thomas Lal, Thilo Hinterberger, Guido Widman, Michael Schröder, N Hill, Wolfgang Rosenstiel, Christian Elger, Niels Birbaumer, and Bernhard Schölkopf. Methods towards invasive human brain computer interfaces. *Advances in neural information processing systems*, 17, 2004.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

W Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

Mercedes-Benz North America. Next generation in-car intelligence: Scaling embedded, on-device intelligence for Mercedes-Benz models. https://group.mercedes-benz.com/technology/innovation/collaboration/liquid-ai.html, 2026. Joint Announcement of Mercedes-Benz and Liquid AI.

Fernando Moreno-Pino, Álvaro Arroyo, Harrison Waldon, Xiaowen Dong, and Álvaro Cartea. Rough transformers: Lightweight and continuous time series modelling through signature patching. *Advances in Neural Information Processing Systems*, 37:106264–106294, 2024.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Roussel Desmond Nzoyem, Nawid Keshtmand, Enrique Crespo Fernandez, Idriss Tsayem, Raul Santos-Rodriguez, David AW Barton, and Tom Deakin. Weight-space linear recurrent neural networks. *arXiv preprint arXiv:2506.01153*, 2025.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Francesco Pappone, Donato Crisostomi, and Emanuele Rodolà. Two-scale latent dynamics for recurrent-depth transformers. *arXiv preprint arXiv:2509.23314*, 2025.

Guillaume Pourcel and Maxence Ernoult. Learning long range dependencies through time reversal symmetry breaking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=w1ihNiIBOc.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026.

Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep ppg: Large-scale heart rate estimation with convolutional neural networks. *Sensors (Basel, Switzerland)*, 19, 2019. URL https://api.semanticscholar.org/CorpusID:198192740.

T Konstantin Rusch and Daniela Rus. Oscillatory state-space models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are RNNs: Balancing parallelization and expressivity. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5EbiopWH6e.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. In *ICLR*, 2023.

Benjamin Walker, Andrew Donald McLeod, Tiexin Qin, Yichuan Cheng, Haoliang Li, and Terry Lyons. Log neural controlled differential equations: The lie brackets make a difference. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=0tYrMtQyPT.

Benjamin Walker, Andrew Donald McLeod, Tiexin Qin, Yichuan Cheng, Haoliang Li, and Terry Lyons. Log neural controlled differential equations: The lie brackets make a difference. In *Forty-first International Conference on Machine Learning*, 2024b.

Eviatar Yemini, Tadas Jucikas, Laura J Grundy, André EX Brown, and William R Schafer. A database of caenorhabditis elegans behavioral phenotypes. *Nature methods*, 10(9):877–879, 2013.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnık, and Jürgen Schmidhuber. Recurrent highway networks. In *International conference on machine learning*, pp. 4189–4198. PMLR, 2017.

A Technical Appendices and Supplementary Material

A.1 Theoretical Analysis

We fix a depth L and a per-block parameter budget k. For each recurrence period m dividing L, let $\mathcal{H}_m$ denote the hypothesis class of depth-L models with m unique blocks repeated periodically across depth. Thus, $\mathcal{H}_1$ corresponds to full depth recurrence (`AAAAAA` for $L = 6$), while $\mathcal{H}_L$ corresponds to the independently parameterized model (`ABCDEF` for $L = 6$).

Proposition 2 (Monotone containment under divisibility) *Let m and m' be divisors of L with $m \mid m'$. Then*

$$\mathcal{H}_m \subseteq \mathcal{H}_{m'}. \tag{14}$$

In particular, for any divisor m of L,

$$\mathcal{H}_1 \subseteq \mathcal{H}_m \subseteq \mathcal{H}_L. \tag{15}$$

Proof 1 *Since $m \mid m'$, write $m' = sm$ for some integer $s \geq 1$. Any model in $\mathcal{H}_m$ has m unique parameter sets $(\theta_1, \ldots, \theta_m)$ repeated periodically across the L depth positions. The same model is recovered in $\mathcal{H}_{m'}$ by setting*

$$\theta'_j = \theta_{((j-1) \bmod m)+1}, \qquad j = 1, \ldots, m'. \tag{16}$$

The resulting period-m' model therefore reproduces the same period-m parameter pattern across all L depth positions.

Remark 1 *The divisibility condition $m \mid m'$ is important. For $L = 6$, the recurrence periods $m = 2$ (`ABABAB`) and $m = 3$ (`ABCABC`) satisfy neither $2 \mid 3$ nor $3 \mid 2$, and their hypothesis classes are therefore not ordered by Proposition 2. For the configurations considered in our experiments,*

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_6, \qquad \mathcal{H}_1 \subseteq \mathcal{H}_3 \subseteq \mathcal{H}_6. \tag{17}$$

Here, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, and $\mathcal{H}_6$ correspond to `AAAAAA`, `ABABAB`, `ABCABC`, and `ABCDEF`, respectively.

Remark 2 (Expressivity and inductive bias) *By Proposition 2, every depth-recurrent configuration considered here is contained in the independently parameterized hypothesis class $\mathcal{H}_L$. Consequently, matching or exceeding the performance of the independent baseline cannot be attributed to greater expressivity. The empirical advantage must therefore arise despite the additional parameter constraints imposed by depth recurrence, suggesting that repeated computation with shared parameters provides a useful inductive bias.*

Remark 3 (Gradient aggregation under parameter sharing) *For a model with recurrence period m, let $r = L/m$ denote the number of repetitions. Because the same parameters are reused across repetitions, gradients from multiple depth positions accumulate into updates of the same parameters. Under block-wise supervision $\mathcal{L}^{\text{block}}$, the gradient with respect to the parameters θ of a shared block takes the form*

$$\frac{\partial \mathcal{L}^{\text{block}}}{\partial \theta} = \frac{1}{r} \sum_{j=1}^{r} \frac{\partial \mathcal{L}\big(g_\phi(h^{(j)}), y\big)}{\partial \theta}, \tag{18}$$

where $h^{(j)}$ denotes the representation after the j-th repetition. Under final supervision $\mathcal{L}^{\text{final}}$, the loss is applied only after the final repetition, while gradients propagate through all repetitions of the shared parameters. Thus, the two objectives provide different supervision signals while retaining parameter sharing across depth.

Remark 4 (Depth iteration and nonlinearity) *Although the state recurrences of LRU, S5, and LinOSS are linear, the complete SSM blocks are nonlinear due to the nonlinear operations surrounding the state recurrence. Depth recurrence therefore iterates nonlinear block maps, whose behavior cannot in general be characterized by the spectral properties of the state-transition matrix alone. A theoretical characterization of when and why such repeated computation is beneficial remains an open question.*

A.2 Datasets and Experimental Setup

A.2.1 Dataset Details

The UEA classification benchmarks used in this work are *EigenWorms* (Worms) (Yemini et al., 2013), *SelfRegulationSCP1* (SCP1) and *SelfRegulationSCP2* (SCP2) (Birbaumer et al., 1999), *MotorImagery* (Motor) (Lal et al., 2004), and *Heartbeat* (Goldberger et al., 2000). Their sequence lengths and input dimensionalities are summarized in Table 1.

Table 1: PPG-DaLiA regression (Reiss et al., 2019) and UEA multivariate time-series classification datasets (Bagnall et al., 2018) used in our experiments, ordered by input dimensionality.

	PPG-DaLiA	Worms	SCP1	SCP2	Heartbeat	Motor
Sequence length T	49,920	17,984	896	1,152	405	3,000
Input dimension	6	6	6	7	61	63

Relative to the fixed hidden and state dimensions of 64 used throughout our experiments, we categorize PPG-DaLiA, Worms, SCP1, and SCP2 as medium-dimensional inputs, and Heartbeat and Motor as high-dimensional inputs. We denote their input dimensions by d and D, respectively.

A.2.2 Models and Training Setup

Previous SSM evaluations have used extensive architecture-specific hyperparameter searches, varying the number of layers, state size, and hidden dimension (Walker et al., 2024b; Rusch & Rus, 2025; Farsang & Grosu, 2026). In contrast, we fix the state size and hidden dimension to 64 and the depth to $L = 6$ across all models. This controlled setup isolates the effect of depth recurrence from changes in model capacity and allows input reshaping to modify the input representation without altering the underlying architecture.

We tune only the learning rate over $\{10^{-3}, 10^{-4}, 10^{-5}\}$. Our implementation builds on the LinOSS codebase (Rusch & Rus, 2025), which includes S5, LRU, and LinOSS, and the LrcSSM codebase (Farsang & Grosu, 2026), extending them with depth recurrence and input reshaping.

Training was conducted on NVIDIA A40 and A100 GPUs with 40 GB and 80 GB of memory, respectively. Each training run completed in under 20 minutes, depending on the dataset and model, with early stopping applied.

A.3 Additional Depth-Recurrence Results

A.3.1 Complete Numerical Results

Table 2 reports the complete numerical results across all benchmarks. Depth-recurrent SSMs match or exceed the baseline mean in the majority of settings (19 of 24 for the fully shared configuration), while showing lower accuracy in a few cases. More broadly, across all 144 depth-recurrent configurations, 132 (91.7%) either achieve a numerically better mean than the independently parameterized baseline or remain within one standard deviation of its mean.

A.3.2 Parameter and Runtime Analysis

Table 3 reports parameter counts and measured training time for the fixed-depth configurations. Depth recurrence reduces the number of parameters substantially, with `AAAAAA` using approximately $5.3 - 5.9\times$ fewer parameters than `ABCDEF`. The reduction is slightly smaller than the ideal $6\times$ factor because the input and output projections are not shared.

At fixed depth $L = 6$, depth recurrence does not reduce the number of block applications. Consequently, the dominant forward-pass computation remains approximately unchanged across recurrence periods, as summarized in Table 4. The measured training times are also similar, with modest reductions for the depth-recurrent configurations.

Adaptive inference changes this tradeoff by reducing the number of block applications. Computational cost therefore decreases with the selected inference depth; for example, applying the `AAAAAA`

Table 2: Performance across all benchmarks for the independent ABCDEF baseline and the depth-recurrent AAAAAA, ABABAB, and ABCABC configurations under $\mathcal{L}^{\text{final}}$ and $\mathcal{L}^{\text{block}}$. Classification tasks report accuracy (%, higher is better), while PPG-DaLiA reports test loss ($\times 10^2$, lower is better). Results are mean $\pm$ std over 5 seeds; bold indicates an improved mean over the corresponding ABCDEF baseline.

Dataset	Model	Baseline ABCDEF $\mathcal{L}^{\text{final}}$	Depth recurrence AAAAAA $\mathcal{L}^{\text{final}}$	AAAAAA $\mathcal{L}^{\text{block}}$	ABABAB $\mathcal{L}^{\text{final}}$	ABABAB $\mathcal{L}^{\text{block}}$	ABCABC $\mathcal{L}^{\text{final}}$	ABCABC $\mathcal{L}^{\text{block}}$
Heartbeat (↑)	S5	74.19 ± 4.67	$\mathbf{75.16 \pm 3.90}$	72.90 ± 5.24	73.55 ± 2.99	73.87 ± 4.9	$\mathbf{76.13 \pm 3.29}$	73.87 ± 1.58
	LRU	71.29 ± 1.21	$\mathbf{72.58 \pm 4.89}$	$\mathbf{72.90 \pm 5.34}$	70.97 ± 1.77	70.32 ± 4.16	$\mathbf{72.90 \pm 4.00}$	70.97 ± 4.08
	LinOSS	70.16 ± 1.80	$\mathbf{73.23 \pm 4.85}$	$\mathbf{70.32 \pm 2.62}$	$\mathbf{72.58 \pm 4.45}$	$\mathbf{73.87 \pm 4.49}$	$\mathbf{73.55 \pm 3.16}$	$\mathbf{72.58 \pm 5.20}$
	LrcSSM	74.67 ± 5.2	$\mathbf{78.33 \pm 5.00}$	$\mathbf{75.67 \pm 6.19}$	$\mathbf{76.33 \pm 4.77}$	$\mathbf{77.00 \pm 5.58}$	$\mathbf{75.67 \pm 3.46}$	$\mathbf{76.67 \pm 5.00}$
SCP1 (↑)	S5	90.12 ± 2.05	88.94 ± 1.91	86.59 ± 4.05	89.65 ± 2.28	88.71 ± 2.64	88.24 ± 1.97	87.29 ± 2.62
	LRU	85.65 ± 2.92	$\mathbf{85.65 \pm 2.28}$	84.94 ± 2.72	$\mathbf{86.35 \pm 0.94}$	$\mathbf{85.65 \pm 4.85}$	85.41 ± 4.25	84.94 ± 3.28
	LinOSS	85.88 ± 4.27	85.41 ± 1.20	$\mathbf{86.12 \pm 3.82}$	83.76 ± 4.36	84.47 ± 1.37	85.18 ± 0.12	83.76 ± 3.89
	LrcSSM	84.52 ± 4.29	$\mathbf{86.19 \pm 2.99}$	$\mathbf{84.52 \pm 3.57}$	84.05 ± 2.87	$\mathbf{84.52 \pm 3.26}$	$\mathbf{85.71 \pm 4.61}$	$\mathbf{85.00 \pm 4.95}$
SCP2 (↑)	S5	54.74 ± 5.25	$\mathbf{57.54 \pm 3.40}$	$\mathbf{57.89 \pm 7.36}$	$\mathbf{55.09 \pm 6.43}$	$\mathbf{55.09 \pm 7.74}$	$\mathbf{54.74 \pm 4.06}$	$\mathbf{56.14 \pm 4.57}$
	LRU	55.09 ± 5.04	$\mathbf{56.14 \pm 3.33}$	54.04 ± 7.56	53.68 ± 5.04	55.09 ± 9.32	54.39 ± 5.32	$\mathbf{56.49 \pm 6.96}$
	LinOSS	54.39 ± 6.37	51.58 ± 4.09	$\mathbf{54.74 \pm 7.48}$	$\mathbf{54.39 \pm 3.51}$	49.47 ± 6.22	$\mathbf{55.09 \pm 6.89}$	$\mathbf{55.79 \pm 2.05}$
	LrcSSM	57.14 ± 5.79	50.71 ± 6.75	50.89 ± 2.31	54.64 ± 9.99	55.00 ± 6.96	56.43 ± 6.00	50.36 ± 8.51
Motor (↑)	S5	54.39 ± 5.77	$\mathbf{55.09 \pm 3.06}$	54.04 ± 11.12	53.68 ± 5.16	53.68 ± 11.13	$\mathbf{55.79 \pm 5.91}$	$\mathbf{54.39 \pm 10.05}$
	LRU	56.14 ± 6.37	$\mathbf{57.89 \pm 6.93}$	51.93 ± 5.04	$\mathbf{56.84 \pm 5.50}$	55.79 ± 2.81	$\mathbf{56.49 \pm 6.79}$	$\mathbf{56.14 \pm 3.84}$
	LinOSS	49.12 ± 2.77	$\mathbf{51.93 \pm 6.34}$	$\mathbf{51.23 \pm 4.76}$	$\mathbf{56.49 \pm 4.35}$	$\mathbf{54.04 \pm 2.33}$	$\mathbf{55.09 \pm 2.11}$	$\mathbf{53.33 \pm 7.24}$
	LrcSSM	57.50 ± 3.43	$\mathbf{57.86 \pm 3.91}$	53.57 ± 5.50	56.43 ± 5.30	$\mathbf{58.21 \pm 6.13}$	55.71 ± 2.33	55.71 ± 6.85
Worms(↑)	S5	86.67 ± 5.10	$\mathbf{89.44 \pm 4.08}$	$\mathbf{87.78 \pm 5.44}$	85.56 ± 3.24	82.78 ± 6.19	83.89 ± 5.39	84.44 ± 5.15
	LRU	88.33 ± 4.08	$\mathbf{90.56 \pm 5.15}$	$\mathbf{90.00 \pm 1.36}$	87.78 ± 4.84	$\mathbf{90.00 \pm 3.77}$	87.78 ± 7.16	$\mathbf{89.44 \pm 4.78}$
	LinOSS	95.83 ± 3.11	90.00 ± 6.48	93.89 ± 4.78	83.33 ± 5.56	91.67 ± 7.24	82.78 ± 11.3	87.22 ± 13.90
	LrcSSM	83.33 ± 5.56	$\mathbf{85.00 \pm 5.41}$	$\mathbf{85.56 \pm 3.62}$	$\mathbf{86.67 \pm 7.19}$	81.67 ± 5.76	$\mathbf{83.89 \pm 7.71}$	82.22 ± 8.0
PPG-DaLiA (↓)	S5	6.83 ± 0.27	$\mathbf{6.40 \pm 0.49}$	$\mathbf{6.50 \pm 0.40}$	7.25 ± 0.48	$\mathbf{6.59 \pm 0.41}$	$\mathbf{6.42 \pm 0.55}$	$\mathbf{6.73 \pm 0.89}$
	LRU	9.33 ± 0.44	$\mathbf{9.08 \pm 0.63}$	$\mathbf{8.69 \pm 0.70}$	$\mathbf{9.23 \pm 0.84}$	$\mathbf{8.34 \pm 0.78}$	$\mathbf{9.14 \pm 0.64}$	$\mathbf{7.84 \pm 1.04}$
	LinOSS	8.33 ± 0.84	$\mathbf{7.65 \pm 0.79}$	$\mathbf{8.41 \pm 0.22}$	$\mathbf{7.54 \pm 0.24}$	$\mathbf{7.69 \pm 0.67}$	$\mathbf{7.54 \pm 0.22}$	$\mathbf{7.81 \pm 0.34}$
	LrcSSM	9.20 ± 1.00	$\mathbf{8.35 \pm 0.71}$	9.64 ± 0.28	$\mathbf{8.75 \pm 0.43}$	$\mathbf{8.33 \pm 1.09}$	$\mathbf{8.54 \pm 0.85}$	$\mathbf{8.99 \pm 1.74}$

Table 3: Parameter counts (thousands) and training time per epoch for each depth-recurrence configuration. Runtime is measured for LRU with $\mathcal{L}^{\text{final}}$ on an NVIDIA A40 GPU. All configurations use depth $L = 6$.

Dataset	ABCDEF Params (K)	ABCDEF Time (s)	AAAAAA Params (K)	AAAAAA Time (s)	ABABAB Params (K)	ABABAB Time (s)	ABCABC Params (K)	ABCABC Time (s)
Heartbeat	154.6	8.12	29.2	5.81	54.3	6.17	79.4	7.69
Worms	151.3	211.82	25.9	209.32	51.0	210.02	76.0	210.54
Motor	154.8	36.63	29.4	35.60	54.5	35.64	79.6	35.71
SCP1	151.1	12.04	25.7	11.18	50.8	11.41	75.8	11.48
SCP2	151.2	15.97	25.7	14.48	50.8	14.78	75.9	14.81
Param. ratio	$1\times$	—	$\sim 1/6\times$	—	$\sim 1/3\times$	—	$\sim 1/2\times$	—

model only twice (AA) requires substantially fewer FLOPs than the fixed six-application setting. Thus, depth recurrence primarily provides parameter savings at fixed depth, while adaptive inference can additionally reduce computation.

A.4 Additional Input Reshaping Results

Figure 6 illustrates the input-reshaping operation for different input dimensionalities.

A.4.1 Complete Numerical Results for Input Reshaping

The upper block of Table 5 reports the complete results for the datasets evaluated in the main text, both with and without depth recurrence. The lower block reports additional results on Worms, SCP1, and SCP2, where input reshaping by concatenating two consecutive timesteps does not provide consistent or substantial improvements across models. Notably, this behavior cannot be attributed to input dimensionality alone, as the medium-dimensional PPG-DaLiA dataset in the upper block benefits substantially from reshaping.

Across the upper-block datasets, 51 of 72 (70.8%) depth-recurrent configurations without reshaping exceed the mean performance of the corresponding original-input baseline (see Table 2). When

Table 4: Computational cost of different layer configurations.

Config.	Params.	FLOPs	Latency	Peak Memory
ABCDEF (6 independent)	151-155k	1.00×	1.00×	1.00×
ABCABC (3 unique × 2)	76-80k	≈ 1.00×	≈ 1.00×	≈ 1.00×
ABABAB (2 unique × 3)	51-54k	≈ 1.00×	≈ 1.00×	≈ 1.00×
AAAAAA (1 unique × 6)	26-29k	≈ 1.00×	≈ 1.00×	≈ 1.00×
AA (2 adaptive layers)	26-29k	0.35×	0.33×	0.55×

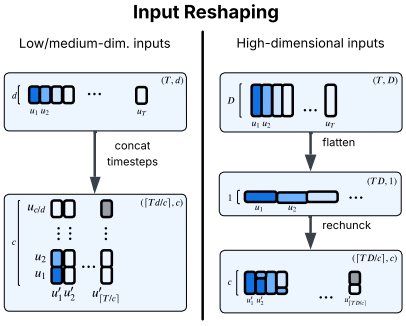


Figure 6: Input reshaping with concentration factor c. For lower- and medium-dimensional inputs, consecutive timesteps are concatenated; for high-dimensional inputs, the joint feature-time representation is flattened and rechunked.

depth recurrence is combined with input reshaping (Table 5), this increases to 57 of 72 (79.2%) configurations. Moreover, 69 of 72 (95.8%) combined configurations are either numerically better than the original-input baseline or remain within one standard deviation of its mean.

In contrast, for Worms, SCP1, and SCP2 in the lower block, only 24 of 72 (33.3%) combined configurations have a numerically better mean than the original-input baseline. Nevertheless, 62 of 72 (86.1%) are either numerically better or remain within one standard deviation of the baseline mean. Overall, these results suggest that the benefits of input reshaping are dataset-dependent rather than determined solely by input dimensionality.

Table 5: Performance with input reshaping and depth recurrence. The first ABCDEF column gives the original-input baseline and the second the reshaped-input baseline. Depth-recurrent configurations use the same reshaped input. Results are mean ± std over 5 seeds. Blue indicates an improved mean of the reshaped baseline over the original baseline; bold indicates an improved mean over the reshaped baseline. The lower block contains the medium-dimensional datasets, for which input reshaping did not yield substantial improvements over the original-input baseline.

Dataset	Model	Baseline ABCDEF $\mathcal{L}^{\text{final}}$	Reshaped-Input Baseline ABCDEF $\mathcal{L}^{\text{final}}$	Depth recurrence with reshaped input AAAAAA $\mathcal{L}^{\text{final}}$	AAAAAA $\mathcal{L}^{\text{block}}$	ABABAB $\mathcal{L}^{\text{final}}$	ABABAB $\mathcal{L}^{\text{block}}$	ABCABC $\mathcal{L}^{\text{final}}$	ABCABC $\mathcal{L}^{\text{block}}$
Motor (↑)	S5	54.39 ± 5.77	**61.75 ± 5.25**	60.00 ± 6.02	58.60 ± 4.24	59.65 ± 5.77	60.00 ± 5.37	60.70 ± 8.35	60.00 ± 7.31
	LRU	56.14 ± 6.37	**58.60 ± 6.04**	53.33 ± 3.06	56.84 ± 5.16	56.14 ± 7.28	54.74 ± 6.79	57.54 ± 6.79	55.79 ± 6.51
	LinOSS	49.12 ± 2.77	**52.28 ± 6.96**	**55.44 ± 5.72**	**55.44 ± 3.44**	**52.63 ± 3.84**	**54.39 ± 7.92**	**55.09 ± 3.61**	**54.74 ± 5.25**
	LrcSSM	57.50 ± 3.43	57.14 ± 8.66	56.07 ± 3.24	55.71 ± 5.11	**60.71 ± 6.06**	52.86 ± 3.70	**60.27 ± 6.08**	55.71 ± 8.22
Heartbeat (↑)	S5	74.19 ± 4.67	74.19 ± 3.38	**76.13 ± 2.37**	**75.80 ± 2.28**	**76.45 ± 2.62**	**75.81 ± 4.33**	**75.81 ± 2.28**	**75.81 ± 2.70**
	LRU	71.29 ± 1.21	**75.48 ± 2.37**	72.90 ± 4.61	73.87 ± 3.59	**75.81 ± 2.89**	**76.13 ± 3.44**	75.16 ± 3.62	75.16 ± 4.16
	LinOSS	70.16 ± 1.80	**70.65 ± 6.40**	**71.29 ± 3.44**	**71.61 ± 4.16**	70.32 ± 3.76	**70.97 ± 4.08**	**72.26 ± 4.49**	**73.23 ± 3.76**
	LrcSSM	74.67 ± 5.2	**77.67 ± 6.08**	75.00 ± 4.56	76.00 ± 6.08	75.67 ± 5.08	75.33 ± 5.45	76.00 ± 7.51	75.33 ± 4.77
PPG-DaLiA (↓)	S5	6.83 ± 0.27	**6.77 ± 0.24**	6.87 ± 0.53	7.47 ± 0.15	**6.48 ± 0.66**	7.52 ± 1.13	**6.57 ± 0.37**	**6.38 ± 0.61**
	LRU	9.33 ± 0.44	**7.07 ± 1.51**	7.72 ± 0.73	8.01 ± 0.33	8.30 ± 0.91	8.08 ± 0.24	8.09 ± 0.62	7.84 ± 1.04
	LinOSS	8.33 ± 0.84	8.96 ± 1.08	**7.80 ± 0.15**	**8.31 ± 0.41**	**8.15 ± 0.59**	**8.09 ± 0.33**	8.88 ± 1.26	**8.30 ± 0.73**
	LrcSSM	9.20 ± 1.00	**8.60 ± 0.59**	8.96 ± 0.16	9.46 ± 1.28	9.21 ± 0.67	9.45 ± 1.21	8.88 ± 1.08	**7.79 ± 0.36**
Worms (↑)	LRU	88.33 ± 4.08	**88.89 ± 6.33**	87.78 ± 6.48	**90.00 ± 7.58**	**91.11 ± 5.09**	**90.56 ± 7.97**	**91.11 ± 5.93**	**88.89 ± 5.83**
	S5	86.67 ± 5.10	83.89 ± 4.44	**85.00 ± 4.51**	83.33 ± 5.83	82.78 ± 3.24	**85.56 ± 2.72**	**84.44 ± 5.15**	82.22 ± 4.51
	LinOSS	95.83 ± 3.11	70.00 ± 31.93	**94.44 ± 3.04**	**91.67 ± 4.65**	**88.33 ± 16.52**	**93.33 ± 5.72**	**77.78 ± 31.38**	**86.67 ± 13.77**
	LrcSSM	83.33 ± 5.56	80.00 ± 7.19	**82.78 ± 4.12**	**82.78 ± 6.02**	**82.78 ± 2.32**	**83.89 ± 4.56**	**82.22 ± 1.52**	**84.44 ± 5.76**
SCP1 (↑)	LRU	85.65 ± 2.92	**86.59 ± 4.74**	86.35 ± 3.12	85.18 ± 2.42	**87.06 ± 3.87**	85.88 ± 3.79	**87.29 ± 4.30**	84.94 ± 3.75
	S5	90.12 ± 2.05	88.94 ± 1.19	87.76 ± 2.53	86.35 ± 2.74	88.47 ± 2.92	86.35 ± 4.25	87.06 ± 3.72	86.35 ± 3.98
	LinOSS	85.88 ± 4.27	85.18 ± 4.12	**87.06 ± 1.29**	**85.41 ± 3.98**	**86.12 ± 5.92**	84.94 ± 1.37	**86.59 ± 2.64**	85.18 ± 3.98
	LrcSSM	84.52 ± 4.29	**84.76 ± 1.00**	**85.71 ± 4.37**	83.33 ± 4.61	84.52 ± 2.23	**85.00 ± 1.99**	**85.00 ± 1.06**	**85.48 ± 2.84**
SCP2 (↑)	LRU	55.09 ± 5.04	**55.09 ± 3.61**	**56.14 ± 1.92**	53.33 ± 7.33	**56.84 ± 2.38**	52.98 ± 6.12	**56.84 ± 1.79**	54.74 ± 10.01
	S5	54.74 ± 5.25	**55.09 ± 3.94**	53.68 ± 3.06	53.33 ± 2.38	55.09 ± 3.06	52.63 ± 2.48	**55.44 ± 2.85**	54.74 ± 5.81
	LinOSS	54.39 ± 6.37	**56.49 ± 5.59**	51.93 ± 1.79	51.23 ± 5.70	54.04 ± 6.11	54.39 ± 5.32	**57.89 ± 5.55**	52.28 ± 2.05
	LrcSSM	57.14 ± 5.79	51.07 ± 1.60	50.71 ± 8.24	**55.00 ± 8.51**	**54.29 ± 9.83**	**55.36 ± 9.11**	**53.21 ± 7.41**	**52.50 ± 5.30**

Table 6: Classification accuracy (%) for different concentration factors c using LrcSSM, reported as mean $\pm$ std over 5 seeds.

Dataset	$c = 2d$	$c = 8$	$c = 16$	$c = 32$
Worms	80.00 ± 7.19 (12)	78.33 ± 6.92	78.89 ± 8.24	76.67 ± 2.48
SCP1	84.76 ± 1.00 (12)	79.29 ± 4.88	80.48 ± 2.47	76.90 ± 5.62
SCP2	51.07 ± 1.60 (14)	48.57 ± 4.96	47.14 ± 7.21	50.00 ± 8.09
Heartbeat	Too large	77.67 ± 6.08	75.00 ± 4.56	71.33 ± 7.76
Motor	Too large	57.14 ± 8.66	54.29 ± 4.82	54.64 ± 5.45

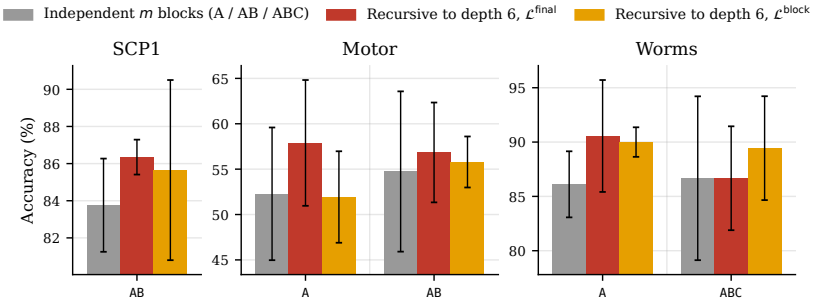


Figure 7: LRU under parameter-matched conditions. Depth-recurrent configurations are compared with independent models containing the same number of unique blocks. Mean $\pm$ std over 5 seeds.

A.4.2 Effect of the Concentration Factor

We additionally vary the concentration factor c using LrcSSM, with results reported in Table 6. For the medium-dimensional datasets, concatenating two complete timesteps ($c = 2d$) performs better than the fixed choices $c \in \{8, 16, 32\}$ overall. For the high-dimensional Heartbeat and Motor datasets, $c = 8$ gives the strongest results among the evaluated values. We also compared zero-padding with discarding trailing values when TD is not divisible by c and observed comparable performance; we therefore use zero-padding to retain all input values.

A.5 Parameter-Matched Ablations

Depth-recursion Under Parameter-matched Conditions The experiments in Section 4.2 compare depth-recurrent models against an independently parameterized 6-layer baseline with more parameters. To isolate the effect of repeated computation from parameter count, we additionally compare each LRU configuration with an independent model containing the same number of unique blocks: `A` vs. `AAAAAA`, `AB` vs. `ABABAB`, and `ABC` vs. `ABCABC`.

Figure 7 shows that depth recurrence can improve over these parameter-matched baselines. The clearest gains occur for the most constrained configuration: applying a single block six times (`AAAAAA`, $\mathcal{L}^{\text{final}}$) improves the mean over the single-block model by 5.6% on Motor and 4.5% on Worms. Improvements also occur with two and three unique blocks, although their magnitude depends on the dataset and supervision strategy. Overall, these results show that the performance of depth-recurrent models cannot be explained by their lower parameter count alone.

Low-Capacity Architectures We further evaluate whether reducing model depth alone can account for the observed behavior using Mamba (Gu & Dao, 2024) and MambaSL (Jung & Kim, 2026), a Mamba variant designed for stronger single-layer performance. Table 7 compares the independently parameterized 6-layer model (`ABCDEF`), a single-block model (`A`), and the same block applied six times (`AAAAAA`). MambaSL improves over Mamba in the single-block setting on both datasets, consistent with its single-layer design. However, reducing the architecture to a single block does not improve over the 6-layer baseline. Reusing that block across depth, in contrast, recovers or improves mean performance without increasing the number of unique SSM-block parameters.

Table 7: Classification accuracy (%) for Mamba (Gu & Dao, 2024) and MambaSL (Jung & Kim, 2026) with six independent blocks (ABCDEF), one block (A), or one block applied six times (AAAAAA). Results are mean ± std over 5 seeds.

Dataset	Model	Baselines		Depth-recursion	
		ABCDEF	Single-layer A	AAAAAA $\mathcal{L}^{\text{final}}$	AAAAAA $\mathcal{L}^{\text{block}}$
Heartbeat (↑)	Mamba	72.58 ± 3.38	69.67 ± 5.34	$\mathbf{73.23 \pm 1.94}$	$\mathbf{73.23 \pm 3.75}$
	MambaSL	70.97 ± 5.30	71.61 ± 5.36	$\mathbf{72.25 \pm 3.73}$	$\mathbf{72.58 \pm 3.38}$
SCP1 (↑)	Mamba	82.12 ± 2.72	78.35 ± 3.76	$\mathbf{83.05 \pm 4.25}$	$\mathbf{84.71 \pm 1.29}$
	MambaSL	84.71 ± 3.65	83.29 ± 4.17	$\mathbf{85.41 \pm 1.76}$	$\mathbf{84.71 \pm 3.07}$

This provides additional evidence that the benefit of depth recurrence is not explained by parameter reduction alone.

A.6 Additional Adaptive-Inference Results

We additionally evaluate adaptive inference for the LRU ABABAB configuration, complementing the AAAAAA results in Section 4.3. The model is trained with three repetitions of AB and, at inference time, is allowed to execute up to six repetitions (12 block applications). Table 8 compares stopping after any block application with stopping only after a complete AB repetition.

Adaptive inference selects different depths across datasets. When stopping is restricted to complete AB repetitions, the average inference depth ranges from approximately 4.4 to 11.1 block applications, compared with the fixed training depth of six. Performance remains close to or even slightly exceeds the fixed-depth baseline on most datasets, although the behavior is dataset-dependent.

We further consider two ablations that restrict adaptive inference to depths of at least six or at most six block applications. Neither restriction consistently improves performance across datasets, indicating that allowing the stopping criterion to select depths both below and above the training depth provides the most flexible inference setting.

Table 8: Adaptive inference for LRU trained with the ABABAB configuration under $\mathcal{L}^{\text{block}}$. The model may execute up to 12 block applications at inference time. Results are classification accuracy (%), reported as mean ± std over 5 seeds. Bold indicates an improved mean over fixed-depth ABABAB.

Dataset	ABABAB $\mathcal{L}^{\text{block}}$	Stop at any block (1-12)	Stop after full AB (2, 4, …, 12)	Avg. depth	Min. depth 6 (6-12)	Max. depth 6 (1-6)
Heartbeat	70.32 ± 4.16	$\mathbf{72.26 \pm 4.13}$	$\mathbf{72.90 \pm 4.26}$	5.25 ± 0.26	65.16 ± 4.16	$\mathbf{71.61 \pm 3.32}$
SCP1	85.65 ± 4.85	79.29 ± 6.68	85.41 ± 4.05	5.04 ± 0.49	81.88 ± 3.03	$\mathbf{86.35 \pm 4.97}$
SCP2	55.09 ± 9.32	$\mathbf{57.19 \pm 4.79}$	$\mathbf{56.14 \pm 6.18}$	11.08 ± 1.23	$\mathbf{56.49 \pm 6.88}$	55.09 ± 6.04
Motor	55.79 ± 2.81	$\mathbf{57.89 \pm 4.44}$	$\mathbf{57.54 \pm 3.40}$	10.50 ± 0.36	$\mathbf{57.54 \pm 3.02}$	54.39 ± 2.72
Worms	90.00 ± 3.77	87.22 ± 5.72	89.44 ± 6.89	4.43 ± 0.18	86.67 ± 5.39	90.00 ± 5.15

A.7 Damped Iterative Updates

Our depth-recurrent models use the direct iterative update

$$h^{(r+1)} = f_\theta\left(h^{(r)}\right), \tag{19}$$

where r denotes the depth iteration and f_θ is the shared SSM block. We investigate whether a more conservative update improves repeated application of the same block.

Specifically, we consider a damped update based on the Krasnosel'skii–Mann iteration (Krasnosel'skii, 1955; Mann, 1953):

$$h^{(r+1)} = (1-\alpha)h^{(r)} + \alpha f_\theta\left(h^{(r)}\right), \qquad \alpha \in [0,1]. \tag{20}$$

Here, α controls how strongly each iteration incorporates the newly computed representation, with $\alpha = 1$ recovering the direct update used in our main experiments. We evaluate both fixed values of α and a learned α.

Table 9 reports the results for the LRU `AAAAAA` configuration. Damping improves the mean accuracy on Motor, with the strongest fixed setting ($\alpha = 0.75$) improving from 51.93% to 56.84%. However, it does not provide consistent improvements across the other datasets, and learning α does not consistently improve over the fixed update. We therefore use the simpler $\alpha = 1$ update throughout the main experiments.

Table 9: Classification accuracy (%) for LRU with the `AAAAAA` configuration under damped depth updates. The direct update used in the main experiments corresponds to $\alpha = 1$. Results are mean $\pm$ std over 5 seeds.

Dataset	$\alpha = 1$	$\alpha = 0.25$	$\alpha = 0.75$	$\alpha = 0.9$	Learned α
Heartbeat	72.90 ± 5.34	71.61 ± 4.63	71.61 ± 2.19	71.94 ± 3.90	72.25 ± 3.73
SCP1	84.94 ± 2.72	84.00 ± 3.29	83.53 ± 2.98	84.94 ± 3.28	84.47 ± 3.28
SCP2	54.04 ± 7.56	53.33 ± 8.60	53.33 ± 5.83	54.39 ± 7.19	51.93 ± 5.91
Motor	51.93 ± 5.04	55.79 ± 5.48	56.84 ± 3.78	53.33 ± 4.24	56.14 ± 2.48
Worms	90.00 ± 1.36	87.78 ± 3.77	87.22 ± 4.51	88.33 ± 3.24	87.22 ± 9.06