Scaling Up Liquid-Resistance Liquid-Capacitance Networks for Efficient Sequence Modeling

Mónika Farsang [1] **Ramin Hasani** [2 3] **Radu Grosu** [1]

Abstract

We present LrcSSM, a *nonlinear* recurrent model that processes long sequences as fast as today's linear state-space layers. By forcing the state-transition matrix to be diagonal and learned at every step, the full sequence can be solved in parallel with a single prefix-scan, giving $\mathcal{O}(TD)$ time and memory and only $\mathcal{O}(\log T)$ sequential depth, for input-sequence length T and a state dimension D. Moreover, LrcSSM offers a formal gradient-stability guarantee that other input-varying systems such as Liquid-S4 and Mamba do not provide. Lastly, for network depth L, as the forward and backward passes cost $\Theta(T\,D\,L)$ FLOPs, with its low sequential depth and parameter count $\Theta(D\,L)$, the model follows the compute-optimal scaling law regime ($\beta \approx 0.42$) recently observed for Mamba, outperforming quadratic-attention Transformers at equal compute while avoiding the memory overhead of FFT-based long convolutions. We show that on a series of long-range forecasting tasks, LrcSSM outperforms LRU, S5 and Mamba.

1. Introduction

With the advent of linear state space models (LSSMs), more and more architectures have emerged, with increasingly better accuracy and efficiency. While LSSMs can be efficiently parallelized, for example with the aid of the parallel scan operator, this is considerably more difficult for traditional, nonlinear state-space models (NSSMs). This lead to a decreasing interest in NSSMs, although these should arguably capture input correlations in a more refined way through their state.

Fortunately, recent work has shown how to apply the parallel scan operator to NSSMs, by linearizing them in every time step, and by implementing this idea in their DEER framework (Lim et al., 2024). Unfortunately, the state-transition matrix (the Jacobian of the NSSM) was not diagonal, which precluded scaling it up to very long sequences. Subsequent work however, succeeded to scale up NSSMs by simply taking the diagonal of the Jacobian matrix, and stabilizing the DEER updates with trust regions. They called this method ELK (evaluating Levenberg-Marquardt via Kalman) (Gonzalez et al., 2024).

In this paper, we propose an alternative approach to scaling up NSSMs. Instead of disregarding the non diagonal elements of the NSSM Jacobian, which might contain important information about the multistep interaction among neurons along feedback loops, we learn an NSSM whose state-transition matrix is constrained to be diagonal, and whose entries depend on both the current state and current input.

In summary, our main contributions in this paper are the following ones:

- To the best of our knowledge, we are the first to show how to scale up a bioinspired RNN to a competitive NSSM on long sequences, by using a diagonal nonlinear state-and-input dependent state-transition matrix and inherently a diagonal Jacobian matrix.
- We demonstrate that LrcSSMs can capture long-horizon tasks in a very competitive fashion on a set of standard benchmarks used to assess LSSMs accuracy and efficiency.
- We show that LrcSSMs consistently outperform many of the state-of-the-art LSSMs, including LRU, S5, S6, and Mamba, especially on the EthanolConcentration benchmark.

2. Background

Here we introduce the necessary background for understanding LrcSSMs: Firstly, the bioinspired nonlinear liquid networks LTCs, STCs, and LRCs, known for their dynamic expressivity. Secondly, the parallelization techniques enabling efficient training of traditionally sequential NSSMs.

[1] Technische Universität Wien (TU Wien) [2] MIT CSAIL [3] Liquid AI. Correspondence to: Mónika Farsang <monika.farsang@tuwien.ac.at>.

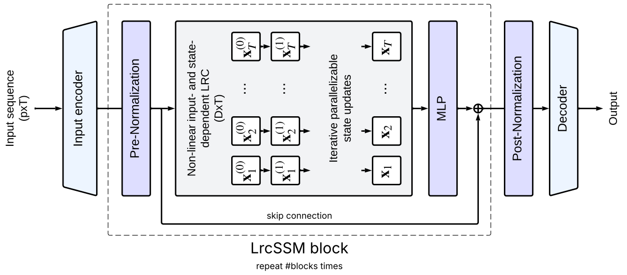

Figure 1. Liquid-Resistance Liquid-Capacitance NSSM (LrcSSM) architecture.

2.1. Bio-inspired Liquid Neural Networks

Electrical Equivalent Circuits (EECs) are simplified models defining the dynamic behavior of the membrane potential (MP) of a postsynaptic neuron, as a function of the MP of its presynaptic neurons and external input signals (Kandel et al., 2000; Wicks et al., 1996). In ML, EECs with chemical synapses are termed liquid time constant networks (LTCs) (Lechner et al., 2020; Hasani et al., 2021). For a neuron i with m presynaptic neurons of MPs x and n inputs of value u, the forget conductance $f_i(x, u)$ and update conductance $z_i(x, u)$ are defined as:

$$f_i(x, u) = \sum_{j=1}^{m+n} g_{ji}^{max} \sigma(a_{ji} y_j + b_{ji}) + g_i^{leak} \qquad (1)$$

$$z_i(x, u) = \sum_{j=1}^{m+n} k_{ji}^{max} \sigma(a_{ji} y_j + b_{ji}) + g_i^{leak}, \qquad (2)$$

where $y = [x, u]$ concatenates the MP (state) of all neurons and the inputs. In Equation (1), g_{ji}^{max} represents the maximum synaptic channel conductance, a_{ji} and b_{ji} parameterize the sigmoidal activation governing channel openness, and g_i^{leak} is the leaking conductance. In Equation (2), $k_{ji}^{max} = g_{ji}^{max} e_{ji}^{rev} / e_i^{leak}$, where e_{ji}^{rev} is the synaptic reversal potential (equilibrium membrane potential) and e_i^{leak} is the leaking potential. Since $g_{ji}^{max} \geq 0$, the sign of k_{ji}^{max} depends on $e_{ji}^{rev} / e_i^{leak}$. In biological neurons, the capacitance has a nonlinear dependence on the MP of presynaptic neurons and the external input as they both may cause the neuron to deform (Howell et al., 2015; Severin et al., 2022; Kumar et al., 2023). This behavior can be modeled by the elastance $\sigma(\epsilon_i(x, u)) = \sum_{j=1}^{m+n} w_{ji} y_j + v_j$.

LRC:

$$\dot{x}_i = (-\sigma(f_i(x, u))\, x_i + \tau(z_i(x, u))\, e_i^{leak})\, \sigma(\epsilon_i(x, u)) \qquad (3)$$

The states $\mathbf{x}$ of LRCs at time t can be computed using the explicit Euler integration scheme as:

$$\text{LRC:} \quad \mathbf{x}_t = \mathbf{x}_{t-1} + \Delta t\, \dot{\mathbf{x}}_{t-1} \qquad (4)$$

2.2. Parallelization Techniques

The DEER method (Lim et al., 2024) formulates next-state computation in NSSMs as a fixed-point problem and solves it using a parallel version of the Newton's method. At each iteration step, DEER linearizes the NSSM. This approximation is widely effective across many domains, and often yields accurate estimates and fast convergence. The main limitation of DEER is the use of a square Jacobian, which does not scale up to long sequences when included in the parallel scan. The second limitation is its numerical instability, which arises from the nature of Newton's method. In particular, the undamped version lacks global convergence guarantees and often diverges in practice (Wright, 2006; Gonzalez et al., 2024).

As an improvement, (Gonzalez et al., 2024) introduces Quasi-DEER, which scales DEER by using the diagonal of the Jacobian, only. This is shown to achieve convergence comparable to Newton's method while using less memory and running faster. Nevertheless, Quasi-DEER still suffers from limited stability.

To stabilize its convergence, Quasi-DEER leverages a connection between the Levenberg-Marquardt algorithm and Kalman smoothing in their ELK (Evaluating Levenberg-Marquardt with Kalman) algorithm (Gonzalez et al., 2024). This stabilization of the Newton iteration by constraining the step size within a trust region, prevents large and numerically unstable updates. As a result, updates are computed using a parallel Kalman smoother, with a running time that is logarithmic in the length of the sequence. Algorithm 1 below, presents this method (Gonzalez et al., 2024).

3. Scaling Up Non-linear LRCs

A scalable DEER or ELK approximation, first computes the dense Jacobian of the NSSM, as shown in Line 7 of Algorithm 1, and then extracts its diagonal as shown in Line 8. This results in a quasi approximation of the original DEER technique, called Quasi-DEER and Quasi-ELK (Gonzalez et al., 2024).

Our Parallelization. Instead of following this approach, we directly modify the underlying nonlinear LRC of Equation (3), such that its Jacobian is diagonal by the formulation itself. The main idea of this modification is that the state-connectivity submatrices a^x, w^x, $g^{max,x}$, and $k^{max,x}$ of the state-and-input-connectivity matrices a, w, g^{max}, and k^{max}, are constant parameter matrices that are theselves diagonalizable. Consequently, all cross terms are zeroed out in the LRC through diagonalization. Accordingly, we learn the complex diagonal matrices directly, instead.

As a result, our own algorithm is no longer a quasi-approximation, as we do not explicitly remove nondiagonal entries. Instead we learn their contribution to the dynamics, within the complex eigenvalues of the diagonal. Consequently, Line 8, $J_s \leftarrow \mathrm{Diag}(J_s)$ of Algorithm 1, is not needed anymore, and the update computations become more efficient. In this way, we retain the best of both approaches: A much more precise, more stable, and more scalable, parallelization technique.

3.1. Proposed Model

In order to achieve a diagonal Jacobian for the LRCs by default, we first modify the Equations (1), (2), by splitting their summation terms into a state-dependent and an input-dependent group, respectively. For the former, we only keep the self-loop synaptic parameters, and zero out all the cross-state synaptic parameters in the associated matrices. For the latter we keep the influence of all external inputs u through their cross-input synaptic parameters, as this part is zeroed out anyway in the Jacobian. To highlight the separation of the terms, we include an extra superscript x for the learnable parameters in the state-dependent part, and the superscript of u for the parameters in the input-dependent part. This separation results in Equations (5)-(7).

As a consequence, instead of keeping cross-synaptic activations, where each individual synapse between neuron j and i has its own g_{ji}^{max}, b_{ji} and k_{ji}^{max} as it was in Equation (1) and (2), we now only keep the self-loop neural activations, where the synaptic parameters from the same neuron are equal. Note that instead of the *ij* indices, we have only the j index in Equations (5) and (6).

We denote the modified equations of the LRCs with an asterisk. This gives us the following equations for the $f_i^*(x_i, u)$, $z_i^*(x_i, u)$, and $\epsilon_i^*(x_i, u)$ terms:

$$f_i^*(x_i,u) = \underbrace{g_i^{max,x}\sigma(a_i^x x_i + b_i^x)}_{x_i \text{ state-dependent}} + \underbrace{g_i^{max,u}\sigma(\sum_{j=1}^{n} a_{ji}^u u_j + b_j^u) + g_i^{leak}}_{u \text{ input-dependent}} \tag{5}$$

$$z_i^*(x_i,u) = \underbrace{k_i^{max,x}\sigma(a_i^x x_i + b_i^x)}_{x_i \text{ state-dependent}} + \underbrace{k_i^{max,u}\sigma(\sum_{j=1}^{n} a_{ji}^u u_j + b_j^u) + g_i^{leak}}_{u \text{ input-dependent}} \tag{6}$$

$$\epsilon_i^*(x_i,u) = \underbrace{w_i^x x_i + v_i^x}_{x_i \text{ state-dependent}} + \underbrace{\sum_{j=1}^{n} w_{ji}^u u_j + v_j^u}_{u \text{ input-dependent}} \tag{7}$$

$$\begin{aligned}\text{LrcSSM:}\quad \dot{x}_i = &-\sigma(f_i^*(x_i,u))\sigma(\epsilon_i^*(x_i,u))\, x_i \\ &+ \tau(z_i^*(x_i,u))\sigma(\epsilon_i^*(x_i,u))\, e_i^{leak}\end{aligned} \tag{8}$$

For the final form our proposed LRC model, Equation (8) can be formulated into the form of SSMs, by taking the vectorial form of the states $\mathbf{x}$ of size m and input vector $\mathbf{u}$ of size n:

$$\text{LrcSSM:}\quad \dot{\mathbf{x}} = \mathbf{A}(\mathbf{x},\mathbf{u})\mathbf{x} + \mathbf{b}(\mathbf{x},\mathbf{u}), \text{ where} \tag{9}$$

$$\mathbf{A}(\mathbf{x},\mathbf{u}) = \mathrm{diag}\begin{bmatrix} -\sigma(f_1^*(x_1,u))\sigma(\epsilon_1^*(x_1,u)) \\ \dots \\ -\sigma(f_i^*(x_i,u))\sigma(\epsilon_i^*(x_i,u)) \\ \dots \\ -\sigma(f_m^*(x_m,u))\sigma(\epsilon_m^*(x_m,u)) \end{bmatrix},$$

$$\mathbf{b}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \tau(z_1^*(x_1,u))\sigma(\epsilon_1^*(x_1,u))\, e_1^{leak} \\ \dots \\ \tau(z_i^*(x_i,u))\sigma(\epsilon_i^*(x_i,u))\, e_i^{leak} \\ \dots \\ \tau(z_m^*(x_m,u))\sigma(\epsilon_m^*(x_m,u))\, e_m^{leak} \end{bmatrix}.$$

This diagonal $\mathbf{A}(\mathbf{x},\mathbf{u})$ form and the reduced version of $\mathbf{b}(\mathbf{x},\mathbf{u})$ in Equation (9) results in a diagonal Jacobian matrix which makes the parallelizable iterative state updates exact and efficient, that is, this is not anymore a quasi-approximation of the Jacobian.

3.2. Theoretical Insights

The LrcSSM architecture enjoys three important theoretical properties. Firstly, by forcing the state-transition matrix of LrcSSMs to be diagonal and learned at every time step, the full sequence can be solved in parallel with a single prefix-scan, giving $\mathcal{O}(TD)$ time and memory and only $\mathcal{O}(\log T)$ sequential depth, where T is the input-sequence length, and D is the state dimension.

Secondly, LrcSSMs offer a formal gradient-stability guarantee that other input-varying systems such as Liquid-S4 and Mamba do not provide. Lastly, because LrcSSM forward and backward passes cost $\Theta(T\,D\,L)$ FLOPs, where L is the network depth of the LrcSSM architecture, for its low sequential depth and parameter count $\Theta(D\,L)$, the model follows the compute-optimal scaling law regime ($\beta \approx 0.42$) recently observed for Mamba, outperforming quadratic-attention Transformers at equal compute while avoiding the memory overhead of FFT-based long convolutions.

The full proof of all these properties is given in Appendix A, due to obvious space limitations. In particular we provide all details about LrcSSMs stability in A.1 and scalability in A.2.

Table 1. Test accuracy comparison of different models across *long-horizon* datasets ($> 1,500$). The performance of the models marked by † is reported from (Rusch & Rus, 2024). Results are averaged over 5 seeds.

	EthanolConcentration	MotorImagery	EigenWorms
Sequence length	1,751	3,000	17,984
Input size	2	63	6
#Classes	4	2	5
NRDE†	**31.4** ± **4.5**	54.0 ± 7.8	77.2 ± 7.1
NCDE†	22.0 ± 1.0	51.6 ± 6.2	62.2 ± 2.2
Log-NCDE†	**35.9** ± **6.1**	57.2 ± 5.6	82.8 ± 2.7
LRU†	23.8 ± 2.8	51.9 ± 8.6	**85.0** ± **6.2**
S5†	25.6 ± 3.5	53.0 ± 3.9	83.9 ± 4.1
Mamba†	27.9 ± 4.5	47.7 ± 4.5	70.9 ± 15.8
S6†	26.4 ± 6.4	51.3 ± 4.7	**85.0** ± **16.1**
LinOSS-IMEX†	29.9 ± 1.0	**57.9** ± **5.3**	80.0 ± 2.7
LinOSS-IM†	29.9 ± 0.6	**60.0** ± **7.5**	**95.0** ± **4.4**
LrcSSM (Ours)	**36.9** ± **5.3**	**58.6** ± **3.1**	**90.6** ± **1.4**

4. Experiments

We compare LrcSSMs against nine models representing the state of the art for a range of long-sequence tasks. These include the Neural Controlled Differential Equations (NCDE) (Kidger et al., 2020), Neural Rough Differential Equations (NRDE) (Morrill et al., 2021) and Log-NCDE (Walker et al., 2024), Linear Recurrent Unit (LRU) (Orvieto et al., 2023), S5 (Smith et al., 2023), MAMBA (Gu & Dao, 2023), S6 (Gu & Dao, 2023), Linear Oscillatory State-Space models with implicit-explicit time integration (LinOSS-IMEX (Rusch & Rus, 2024)) and with implicit time integration (LINOSS-IM (Rusch & Rus, 2024)).

Long-Horizon Sequence Tasks. We focus on the tasks from the UEA Multivariate Time Series Classification Archive that require learning long-range interactions, especially those with a sequence length above 1,500, up to 18,000. These include the EthanolConcentration dataset (Large et al., 2018), which contains spectroscopic recordings of solutions, the MotorImagery dataset, which captures data from the motor cortex, and the EigenWorms (Yemini et al., 2013) dataset of postural dynamics of the worm *C. elegans*.

As shown in Table 4, our LrcSSM model outperforms all other state-of-the-art methods on the EthanolConcentration task and achieves second-best performance on the MotorImagery and EigenWorms datasets.

Further experiments and details on them are presented in Appendix E and G. While ablation studies are reported in Appendix H.

5. Conclusion

In this work, we revisited the potential of nonlinear RNNs in the era of efficient, scalable LSSMs. While LSSMs have seen remarkable success due to their parallelizable structure and computational efficiency, nonlinear RNNs have largely been sidelined due to their inherently sequential nature. However, recent advances, particularly the DEER and ELK methods, have opened the door to parallelizing nonlinear RNNs, thus challenging their long-standing scalability limitation.

Building on these developments, we introduced the liquid-resistance liquid-capacitance nonlinear state-space model (LrcSSM), a novel NSSM architecture that combines the expressive power of bioinspired nonlinear RNNs with the scalability of modern LSSMs. By adapting the ELK method and carefully redesigning the internal structure of LRCs, we enable efficient parallel computation by inherently learning diagonal Jacobian matrices, while still preserving the dynamic richness of nonlinear state updates in biological neurons. Our design allows for exact ELK updates, rather than relying on quasi-approximations. Our results suggest that nonlinear-RNN-based SSMs are a promising direction for future research in sequence modeling.

Our experiments demonstrate that LrcSSM not only matches but often exceeds the performance of leading LSSMs such as LRU, S5, S6, and Mamba, particularly in long-horizon sequence modeling tasks. These results suggest that nonlinear-RNN-based SSMs are not only a feasible solution but can also be competitive, offering a promising direction for future research in sequence modeling.

Acknowledgements

M.F. has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101034277.

References

Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., and Flor, H. A spelling device for the paralysed. *Nature*, 398(6725):297–298, 1999.

Dao, T. and Gu, A. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Farsang, M., Neubauer, S. A., and Grosu, R. Liquid resistance liquid capacitance networks. In *The First Workshop on NeuroAI@ NeurIPS2024*, 2024.

Fu, D. Y., Kumbong, H., Nguyen, E., and Ré, C. Flashfftconv: Efficient convolutions for long sequences with tensor cores. *arXiv preprint arXiv:2311.05908*, 2023.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23): e215–e220, 2000.

Gonzalez, X., Warrington, A., Smith, J., and Linderman, S. Towards scalable and stable parallelization of nonlinear rnns. *Advances in Neural Information Processing Systems*, 37:5817–5849, 2024.

Grazzi, R., Siems, J., Zela, A., Franke, J. K. H., Hutter, F., and Pontil, M. Unlocking state-tracking in linear rnns through negative eigenvalues, 2025. URL https://arxiv.org/abs/2411.12537.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. doi:10.48550/arXiv.2312.00752.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35: 35971–35983, 2022a.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces, 2022b. URL https://arxiv.org/abs/2111.00396. doi:10.48550/arXiv.2111.00396.

Hasani, R., Lechner, M., Amini, A., Rus, D., and Grosu, R. Liquid time-constant networks. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 35(9), pp. 7657–7666, 2021. doi:10.1609/aaai.v35i9.16936.

Hasani, R., Lechner, M., Wang, T.-H., Chahine, M., Amini, A., and Rus, D. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*, 2022. doi:10.48550/arXiv.2209.12951.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Howell, B., Medina, L., and Grill, W. Effects of frequency-dependent membrane capacitance on neural excitability. *Neural Engineering*, 12(5):56015–56015, October 2015.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S., Hudspeth, A. J., Mack, S., et al. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020a.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling Laws for Neural Language Models, 2020b. URL https://arxiv.org/abs/2001.08361. Version Number: 1.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33: 6696–6707, 2020.

Kumar, J., Gupta, P. D., and Ghosh, S. Effects of nonlinear membrane capacitance in the hodgkin-huxley model of action potential on the spike train patterns of a single neuron. *Europhysics Letters*, 142(6):67002, jun 2023. doi: 10.1209/0295-5075/acd80c.

Large, J., Kemsley, E. K., Wellner, N., Goodall, I., and Bagnall, A. Detecting forged alcohol non-invasively through vibrational spectroscopy and machine learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 298–309. Springer, 2018.

Lechner, M., Hasani, R., Amini, A., Henzinger, T. A., Rus, D., and Grosu, R. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2(10): 642–652, 2020. doi:10.1038/s42256-020-00237-3.

Lim, Y. H., Zhu, Q., Selfridge, J., and Kasim, M. F. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations*, 2024.

Martin, E. and Cundy, C. Parallelizing linear recurrent neural nets over sequence length. *arXiv preprint arXiv:1709.04057*, 2017.

Morrill, J., Salvi, C., Kidger, P., and Foster, J. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, pp. 7829–7838. PMLR, 2021.

Movahedi, S., Sarnthein, F., Cirone, N. M., and Orvieto, A. Fixed-point rnns: From diagonal to dense in a few iterations. *arXiv preprint arXiv:2503.10799*, 2025.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Parnichkun, R. N., Massaroli, S., Moro, A., Smith, J. T., Hasani, R., Lechner, M., An, Q., Ré, C., Asama, H., Ermon, S., et al. State-free inference of state-space models: The transfer function approach. *arXiv preprint arXiv:2405.06147*, 2024.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Poli, M., Thomas, A. W., Nguyen, E., Ponnusamy, P., Deiseroth, B., Kersting, K., Suzuki, T., Hie, B., Ermon, S., Ré, C., et al. Mechanistic design and scaling of hybrid architectures. *arXiv preprint arXiv:2403.17844*, 2024.

Rusch, T. K. and Rus, D. Oscillatory state-space models. *arXiv preprint arXiv:2410.03943*, 2024.

Severin, D., Shirley, S., Kirkwood, A., and Golowasch, J. Daily and cell type-specific membrane capacitance changes in mouse cortical neurons. *bioRxiv*, 2022. doi: 10.1101/2022.12.09.519806.

Smith, J. T., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling. In *ICLR*, 2023.

Walker, B., McLeod, A. D., Qin, T., Cheng, Y., Li, H., and Lyons, T. Log neural controlled differential equations: The lie brackets make a difference. In *Forty-first International Conference on Machine Learning*, 2024.

Wicks, S. R., Roehrig, C. J., and Rankin, C. H. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

Wright, S. J. Numerical optimization, 2006.

Yemini, E., Jucikas, T., Grundy, L. J., Brown, A. E., and Schafer, W. R. A database of caenorhabditis elegans behavioral phenotypes. *Nature methods*, 10(9):877–879, 2013.

Technical Appendices and Supplementary Material

A. Theoretical Insights

A.1. Stability

We analyze a single hidden dimension— because every recurrence is diagonal, all dimensions behave independently and identically. Recall the discrete-time update:

$$x_{t+1} \;=\; \lambda_t\, x_t \;+\; b_t, \qquad 0<\lambda_t\le\rho<1, \tag{10}$$

where ρ is a user–chosen radius (typically $0.9-0.99$) enforced by either the *tanh-clamp* or the *negative–softplus–exponential* parametrisation.

One step is contractive.

Lemma A.1 (ρ–contraction)**.** *For any $x, y \in \mathbb{R}^D$ we have* $\|x_{t+1}-y_{t+1}\|_2 \;=\; \|\lambda_t(x-y)\|_2 \;\le\; \rho\,\|x-y\|_2$.

Proof. λ_t is diagonal with all entries $\le\rho$, hence its operator (spectral) norm is $\le\rho$; multiplying by it can only shrink Euclidean distances. □

Forward states stay bounded. Iterating Lemma A.1 t times yields

$$\|x_t\|_2 \;\le\; \rho^{\,t}\|x_0\|_2 \;+\; \frac{1-\rho^{\,t}}{1-\rho}\,B, \qquad B:=\max_{s\le t}\|b_s\|_2. \tag{11}$$

Therefore the hidden state can *never blow up*, irrespective of sequence length.

Back-propagated gradients never explode.

Theorem A.2 (Gradient stability)**.** *Let a loss L depend only on the final state x_T. Then for any $0\le\tau<T$*

$$\big\|\nabla_{x_\tau}L\big\|_2 \;\le\; \rho^{\,T-\tau}\,\big\|\nabla_{x_T}L\big\|_2,$$

hence the Jacobian product norm is ≤ 1 and cannot explode.

Proof. The Jacobian of one step is $J_t=\lambda_t$, so $\|J_t\|_2\le\rho$. Back-propagation multiplies $T-\tau$ such Jacobians: $\nabla_{x_\tau}L = J_\tau^\top\cdots J_{T-1}^\top\nabla_{x_T}L$. Sub-multiplicativity of the spectral norm gives the result. □

Controlled vanishing. Because ρ is *tunable*, gradients decay at most geometrically: choosing $\rho\approx0.99$ keeps long-range signals alive; smaller values add regularisation.

Deep stacks. For L stacked layers with radii ρ_ℓ the bound becomes $\big\|\nabla^{(\text{layer } L)}_{x_\tau}L\big\|_2 \le \big(\prod_{\ell=1}^{L}\rho_\ell^{\,T-\tau}\big)\|\nabla_{x_T}L\|_2$. Keeping every ρ_ℓ close to 1 therefore preserves stability in depth.

How other models handle forward/gradient stability. S4/S6 keep $\mathrm{Re}(A)<0$ and collapse the recurrence into a single convolution kernel. In this setting, forward activations are bounded and back-propagated Jacobians never appear. **Mamba** re-introduces recurrence via a gate $\sigma(\cdot)\in[0,1]$; if that gate is clipped the same ρ-Lipschitz bound as ours holds, but no proof is given. **LinOSS** discretizes a non-negative diagonal ODE with a symplectic IMEX step, proving both state and gradient norms stay ≤ 1. **Liquid-S4** adds an input term $B\,u_t$ without clamping the spectrum, so stability relies on empirical eigenvalue clipping. Thus, among truly recurrent models, only LrcSSM (and LinOSS under its specific integrator) enjoy a formal guarantee that *both* forward trajectories and full Jacobian chains remain inside the unit ball.

LrcSSM has a stronger guarantee than Liquid-S4 or Mamba, and—unlike S4-type convolutions, can propagate gradients through actual recurrent steps while remaining provably safe from explosion. This makes training deep, long-sequence stacks straightforward: set $\rho\approx1$, forget about gradient clipping, and tune ρ itself as a single parameter to trade off memory length versus regularization.

A.2. Scalability

Let T denote the input sequence length and D the state dimension. Sequential methods inherently cannot be parallelized, requiring $\mathcal{O}(D)$ memory complexity and $\mathcal{O}(TD^2)$ computational work. Compared to this, the DEER (Lim et al., 2024) method is parallel but it comes with a major drawback, it requires $\mathcal{O}(TD^2)$ memory complexity and $\mathcal{O}(TD^3)$ computational cost.

The ELK technique introduced in (Gonzalez et al., 2024) achieves fast and stable parallelization by incorporating diagonal Jacobian computation for scalability. This reduces both memory and computational complexity significantly to $\mathcal{O}(TD)$. Our approach achieves the same complexity — $\mathcal{O}(TD)$ for both memory and computation, thanks to the use of inherently diagonal Jacobians.

Now let's assess formal complexity and compute–optimal scaling laws for LrcSSM:

Compute, throughput, and memory. Let FLOPs $\approx c_f\,B\,T\,D\,L$, be the dominant training cost, where B is the batch size, T the sequence length, D the hidden width, L the network depth, and c_f an architecture–specific constant we define (lower for SSMs and higher for Transformers). The single-GPU throughput (tokens s^{-1} GPU^{-1}) is throughput $\approx\frac{TB}{\text{wall-clock time}}$ The *memory footprint* is the sum of peak activations and model parameters.

Scaling-law (Kaplan et al., 2020b; Hoffmann et al., 2022)**.** A *scaling law* is any asymptotic or empirical relation of the form

$$\mathrm{Loss}(C) \;=\; A\,C^{-\beta}+E, \qquad C = \text{compute (FLOPs)}, \quad \beta > 0, \tag{12}$$

or a closed-form complexity identity such as FLOPs $\propto$ $T\,D$.

Recent large-scale studies like (Poli et al., 2024) show that β depends on the operator's per-token cost: *Dense attention*: $\beta \approx 0.48$–0.50 (Kaplan et al., 2020b). *Linear-time RNN/SSM (Mamba, Hyena):* $\beta \approx 0.42$–0.45 in 70 M–7 B runs (Gu & Dao, 2023; Poli et al., 2023). *Hybrid (recurrence + sparse attention):* β can reach 0.41 (MAD pipeline) (Poli et al., 2024).

Table 2 summarizes the per–layer cost of the main long-sequence architectures in terms of forward/backward FLOPs, peak activation memory, and parallel depth over the sequence length T. Because LrcSSM shares the same $\mathcal{O}(TD)$ compute curve as Mamba but with a smaller constant c_f (no low-rank gate, no FFT), we expect it to sit at—or slightly below—the 0.42–0.45 band. The claim is compatible with existing data: Mamba-3B matches a 6-B Transformer at the same FLOPs (Gu & Dao, 2023), and LinOSS shows $2\times$ lower NLL than Mamba on 50 k-token sequences at equal compute (Rusch & Rus, 2024). Hence, $\beta \approx 0.42$ is a defensible prior for LrcSSM; a hybrid LrcSSM + local-attention block could plausibly move β toward 0.41.

Sequence-length scaling. For single–GPU throughput $K(T)$, LrcSSM inherits the near-perfect linear behaviour $K(T) \propto T$ of the scan primitive, with practical speed-ups obtainable through width-w windowing and double buffering that saturate L2 cache bandwidth. Liquid-S4 degrades linearly in *latency* because it remains sequential, whereas FFT-based S4/Hyena layers incur $\mathcal{O}(T \log T)$ compute and become memory-bound beyond $T \approx 64$k tokens. Hence, for contexts up to 64k, LrcSSM (and Mamba) are the *compute winners*; at larger T the FFT models may overtake them in raw FLOPs but pay a significant activation cost.

Sequence-length scaling. Let $K(T)$ be the wall-clock time for a single forward pass of length T on one GPU. LrcSSM: $K(T) \approx \dfrac{c}{\mathrm{SMs}}\,T$ (linear) but can drop to $\approx \dfrac{c}{\mathrm{SMs}}\,\dfrac{T}{w}$ with a width-w scan and double-buffering—near-perfect L2-cache reuse, where SM is the number of CUDA Streaming Multiprocessors on the GPU, and c a hardware-and-kernel–dependent constant (e.g., time per token per SM). Mamba (Gu & Dao, 2023): same asymptotic, but the fused CUDA kernel shows $\approx 5\times$ higher throughput than a Transformer on $4\,$k tokens; on shorter sequences the constant cost of its scan kernel dominates. S4/Hyena (FFT): $\mathcal{O}(T \log T)$; cross-over with linear methods occurs around $T \approx 8$–$16\,$k on A100s—FlashFFTConv reduces the constant $4\times$–$8\times$ (Fu et al., 2023). Liquid-S4 (Hasani et al., 2022): remains sequential; throughput degrades linearly without remedy.

Thus, for $T \leq 64\,$k, LrcSSM and Mamba are compute winners; beyond $64\,$k, Hyena/S4 win in pure flops but can be memory-bound.

B. Parallelizing Non-linear RNNs

Algorithm 1 ELK (Gonzalez et al., 2024)

```
procedure ELK($f$, $s_0$, init_guess, tol, method, quasi)
    diff $\leftarrow \infty$
    states $\leftarrow$ init_guess
    while diff $>$ tol do
        shifted_states $\leftarrow [s_0$, states$[:-1]]$
        $f_s \leftarrow f$(shifted_states)
        $J_s \leftarrow$ GetJacobians($f$, shifted_states)
        $J_s \leftarrow$ Diag($J_s$)
        $b_s \leftarrow f_s - J_s \cdot$ shifted_states
        new_states $\leftarrow$ ParallelKalmanFilter($J_s, b_s$, states, $s_0$)
        diff $\leftarrow \|$ states $-$ new_states$\|_\infty$
        states $\leftarrow$ new_states
    end while
    return states
end procedure
```

C. Comparison to Other Techniques

C.1. Comparison to Linear State Space Models

State-of-the-art time-invariant LSSMs typically take the following general form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{13}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{14}$$

The main differences between LrcSSM and time-invariant LSSMs are the following:

- There is no non-linearity in the recurrent and input update ($\mathbf{A}$ and $\mathbf{B}$, respectively) in time-invariant LSSMs, which allows them to be parallelized over the time dimension. Here, we investigate non-linear recurrent update and non-linear input update too.
- There are two key aspects of the matrices that state-of-the-art LSSMs usually follow:
    (1) First, matrix $\mathbf{A}$ is generally time-invariant (constant), although recent work has introduced an input-dependent variant $\mathbf{A}(\mathbf{u})$ (Hasani et al., 2022; Gu & Dao, 2023). In our model however, this matrix is both state-and-input dependent, $\mathbf{A}(\mathbf{x}, \mathbf{u})$. Second, instead of using a traditional

Table 2. Per–layer asymptotic complexity (sequence length T, width D).

Architecture	**F/B FLOPs**	**Memory**	**Parallel depth**
Mamba(Gu & Dao, 2023)	$\mathcal{O}(TD)$	$\mathcal{O}(D)$	$\mathcal{O}(\log T)$
LinOSS(Rusch & Rus, 2024)	$\mathcal{O}(TD)$	$\mathcal{O}(D)$	$\mathcal{O}(\log T)$
Liquid-S4(Hasani et al., 2022)	$\mathcal{O}(TD)$	$\mathcal{O}(D)$	$\mathcal{O}(T)$
S4/Hyena(Gu et al., 2022b; Poli et al., 2023)	$\mathcal{O}(T \log T\, D)$	$\mathcal{O}(T)$	$\mathcal{O}(\log T)$
Transformer(Kaplan et al., 2020a)	$\mathcal{O}(T^2 D)$	$\mathcal{O}(T^2 + TD)$	$\mathcal{O}(1)$
LrcSSM (ours)	$\mathcal{O}(TD)$	$\mathcal{O}(D)$	$\mathcal{O}(\log T)$

$\mathbf{B}$ matrix that is simply multiplied by the input $\mathbf{u}$, we adopt the form $\mathbf{b}(\mathbf{x}, \mathbf{u})$, allowing the input to have a more embedded influence on the state update.

(2) Modern LSSMs typically require a special initialization, such as diagonal plus-low rank parameterization of the transition matrix of the LSSMs via higher-order polynomial projection (HiPPO) matrix (Gu et al., 2020) or only diagonal transition matrices with specific parameterization (Gu et al., 2022a; Orvieto et al., 2023). In our case, we calculate the entries of $\mathbf{A}$ and $\mathbf{b}$ from biology-grounded equations of (9).

C.2. Comparison to Liquid-Resistance Liquid-Capacitance Networks (LRCs)

In summary, our approach of LrcSSM differs from LRCs (Farsang et al., 2024) in the following ways:

- Learning in LRCs, like in traditional NSSMs (or nonlinerar RNN models), is inherently sequential. In contrast, we aim for an efficient, parallelizable version in LrcSSMs.
- We modified entries of $\mathbf{A}$ and $\mathbf{b}$ to only depend on the self states, rather than on all other states, while still allowing them to depend on the full input. This change yields diagonal Jacobians, exact solutions, and improved efficiency in the update computations.
- While the original LRCs use a single computation layer (a single computation block), we have restructured the LRC architecture into a block-wise design in LrcSSMs, similar to the LSSM-styled models such as LRU and S5. This design is illustrated in Figure 1.

D. Related work

Linear Structural State-Space Models (LSSMs). Since the introduction of S4 (Gu et al., 2022b), LSSMs rapidly evolved in sequence modeling. S4 used FFT to efficiently solve linear recurrences, and inspired several variants, including S5 (Smith et al., 2023), which replaced FFT with parallel scans. Liquid-S4 (Hasani et al., 2022) introduced input-dependent state-transition matrices, moving beyond the static structure but relied on FFT. Recent work, such as S6 and Mamba (Gu & Dao, 2023), adapted the concept of input-dependency and continued to push LSSMs to more efficient computation with a hardware-aware parallel algorithm.

Parallelizing Non-linear State Space Models (NSSMs). While traditional nonlinear RNNs have been favored for their memory efficiency, their major limitation lies in the lack of parallelizability over the sequence length. This has led to the development of parallelizable alternatives, such as (Martin & Cundy, 2017; Orvieto et al., 2023; Movahedi et al., 2025; Grazzi et al., 2025). One notable example is the Linear Recurrent Unit (LRU) (Orvieto et al., 2023), which uses complex diagonal state-transition matrices with stable exponential parameterization, achieving comparable performance with LSSMs. While LRUs argue that linear recurrence is sufficient, in this work we show that incorporating non-linearity in the transition dynamics can offer significant advantages. Importantly, these approaches achieve parallelism through entirely new architectures, without addressing how to parallelize existing NSSMs. Techniques like DEER (Lim et al., 2024) and ELK (Gonzalez et al., 2024) fill this gap by enabling parallel training and inference for arbitrary non-linear recurrent models.

Positioning LrcSSM in Recent Advances. Our LrcSSM aligns with the structured state-space duality (SSD) framework introduced by (Dao & Gu, 2024), as its main focus is on designing an RNN that behaves almost like an SSM (diagonalizable and parallelizable). In addition, recent work on parallel state-free inference (Parnichkun et al., 2024) can be also combined with LrcSSMs to further enhance their efficiency.

E. Experiments

We follow the same classification evaluation benchmark proposed in (Walker et al., 2024) and then used by (Rusch & Rus, 2024). These tasks are part of the UEA Multivariate Time Series Classification Archive (UEA-MTSCA). All of these datasets consist of biologically or physiologically grounded time-series data, derived from real-world measurements of dynamic systems, which can be human, animal, or chemical. They capture continuous temporal signals such as neural activity, bodily movements, or spectroscopic readings, making them very well-suited for benchmarking models that need to learn complex temporal dependencies. We followed the exact same hyperparameter-tuning protocol, using a grid search over the validation accuracy. More details on these experiments are given in the Appendix G.2. After fixing the hyperparameters, we compare the average test set accuracy over five different random splits of the data. As we are reporting the results of the other models from Rusch et. al (Rusch & Rus, 2024), we also used the exact same seeds for the dataset splitting as well. When presenting the results, we highlight the top three performing models.

Short-Horizon Sequence Tasks. In Table 3, we report results on datasets with sequence lengths shorter than 1,500 elements. These datasets include the Heartbeat dataset (Goldberger et al., 2000), which contains heart sound recordings, as well as SelfRegulationSCP1 and SelfRegulationSCP2 (Birbaumer et al., 1999), which include data on cortical potentials. We report the test accuracy results. We found that our LrcSSM model performed average on these tasks, and we suspect that they lack interesing input correlations.

Long-Horizon Sequence Tasks. Here, we present the table again for better readability and flow purposes, with the sequence length, input size and number of classes included.

Average Performance Across Datasets. In Table 5, we report the average accuracy across all six datasets considered from the UEA-MTSCA archive. LrcSSM achieved an accuracy of 66.3%, placing it at the forefront alongside the LinOSS-IM model, outperforming all other state-of-the-art models, including LRU, S5, S6, Mamba, and LinOSS-IMEX. The implicit integration scheme of the LinOSS-IM model, seems to have played an important role, and we plan to investigate a similar integration scheme for LrcSSM, too. Our current scheme is just a simple explicit Euler.

F. Discussion

Competitive Long-Horizon Performance. Our experimental evaluations show that the LrcSSM model performs moderately well on short-horizon datasets, as seen in Table 3, while demonstrating highly competitive performance on datasets with long input sequences, as shown in Table 4. In those long-sequence tasks, LrcSSMs outperform LRUs, Mamba, and S6, and also achieve better average performance across all datasets, as presented in Table 5.

The only model that LrcSSMs generally does not outperform is the LinOSS-IM model, except on the EthanolConcentration dataset (for both LinOSS-IMEX and LinOSS-IM versions), and on MotorImagery and EigenWorms (in the case of LinOSS-IMEX). This may be attributed to the fact that LinOSS is based on forced linear second-order ODEs, whereas LrcSSMs are built upon LRCs, which are nonlinear first-order ODEs. Another possible reason lies in the integration technique: while we were able to outperform the implicit-explicit (IMEX) integration scheme, we did not surpass the fully implicit one (IM) in average test accuracy. This suggests that more sophisticated integration schemes for LrcSSMs (which currently use explicit Euler) may be worth investigating.

Biological Inspirations in Sequence Modeling. We find it particularly interesting that the LinOSS model also exhibits biological relevance, as it models cortical dynamics through harmonic oscillations. In contrast, our approach models information transmission through chemical synapses, which is a different biological phenomenon. The strong performance of both approaches, despite being grounded in different aspects of neuroscience, highlights the significant potential of biologically inspired models as a foundation for future research in sequence modeling.

Efficient Sequence Modeling with Diagonalized Jacobians. In this paper, we focused on the biologically inspired non-linear LRC model, and demonstrated how this model can be made more efficient for long-sequence modeling, by redesigning its underlying state-transition matrix $\mathbf{A}$ and its input-transition vector $\mathbf{b}$, such that the resulting Jacobian is a diagonal matrix, for the state-update iterations. This matrix can then be directly used in the parallelizable ELK method, which gives an exact ELK update, and not an approximation. We believe this approach can also be applied to many other non-linear RNNs of interest.

Limitations. As pointed out in Section A.2, this is parallelized version holds a good promise towards efficient non-linear RNNs compared to sequential computation costs. Linear SSMs have also the same costs. However, we also have to take into account that LRCs solved by ELK need more Newton steps to converge at each iteration, which linear SSMs do not require. The number of iterations depends on the convergence of the state updates, which stops once the difference between the consecutive state updates gets below a defined threshold (see Line 4 of Algorithm 1).

Table 3. Test accuracy comparison of different models across relatively *short-horizon* datasets ($< 1,500$). The performance of the models marked by † is reported from (Rusch & Rus, 2024). The same hyperparameter tuning protocol and dataset splitting over the same 5 seeds were used.

	Heartbeat	SelfRegulationSCP1	SelfRegulationSCP2
Sequence length	405	896	1,152
Input size	61	6	7
#Classes	2	2	2
NRDE†	73.9 ± 2.6	76.7 ± 5.6	48.1 ± 11.4
NCDE†	68.1 ± 5.8	80.0 ± 2.0	49.1 ± 6.2
Log-NCDE†	74.2 ± 2.0	82.1 ± 1.4	54.0 ± 2.6
LRU†	**78.1 ± 7.6**	84.5 ± 4.6	47.4 ± 4.0
S5†	73.9 ± 3.1	**87.1 ± 2.1**	**55.1 ± 3.3**
Mamba†	**76.2 ± 3.8**	80.7 ± 1.4	48.2 ± 3.9
S6†	**76.5 ± 8.3**	82.8 ± 2.7	49.9 ± 9.4
LinOSS-IMEX†	75.5 ± 4.3	**87.5 ± 4.0**	**58.9 ± 8.1**
LinOSS-IM†	75.8 ± 3.7	**87.8 ± 2.6**	**58.2 ± 6.9**
LrcSSM (Ours)	72.7 ± 5.7	85.2 ± 2.1	53.9 ± 7.2

G. Experimental Details

G.1. Training Setup

We used A100 GPUs with 80 GB of memory. Training time ranged from less than 1 up to 2-3 hours per data split, depending on the dataset and model. Early stopping was used to prevent overfitting, which varies the training time.

G.2. Hyperparameters

We performed a grid search over the following set of hyperparameters:

Using the grid shown in Table 6, we selected the best configuration for each dataset based on the average validation accuracy across five data splits. The splits were generated using the same random seeds as in (Rusch & Rus, 2024) to ensure full comparability. The final hyperparameters used to report the test accuracies are listed in Table 7.

We found that, in general, LrcSSMs benefit from higher learning rates and are not particularly sensitive to the hidden dimension of the encoded input. However, a lower state-space dimension and fewer layers tend to be advantageous.

G.3. Dataset sources

The datasets can be downloaded from the following links:

- Short-horizon tasks:
  - Heartbeat
  - SelfRegulationSCP1
  - SelfRegulationSCP2
- Long-horizon tasks:
  - EthanolConcentration
  - MotorImagery
  - EigenWorms

G.4. Additional Remarks on the Datasets

We used the datasets as they were publicly available (i.e., without an additional time dimension). However, this aspect is treated as an additional hyperparameter in the models reported by (Rusch & Rus, 2024). We hypothesize that incorporating this dimension could help our model learn even better dependencies, which might be worth investigating in the future.

G.5. Additional Remarks on the Model Design

Integration Scheme. As pointed out in Section F, we used the explicit Euler integration scheme. This is a simple and straightforward solution, but it might be worth investigating more sophisticated and computationally expensive integration methods. In fact, we conducted some preliminary experiments with a hybrid explicit-implicit solver but did not observe any performance improvement, although we did not explore it across the full hyperparameter grid.

Integration Timestep. For the integration step, we used a timestep of $\Delta t = 1$ in all our experiments. As (Rusch & Rus, 2024) investigated different Δt values across the datasets and observed no substantial gain in performance, they also continued with $\Delta t = 1$ for all their experiments. However, it might still be worth investigating this in our case as well.

Table 4. Test accuracy comparison of different models across *long-horizon* datasets ($> 1,500$). The performance of the models marked by † is reported from (Rusch & Rus, 2024). Results are averaged over 5 seeds.

	EthanolConcentration	MotorImagery	EigenWorms
Sequence length	1,751	3,000	17,984
Input size	2	63	6
#Classes	4	2	5
NRDE†	**31.4 ± 4.5**	54.0 ± 7.8	77.2 ± 7.1
NCDE†	22.0 ± 1.0	51.6 ± 6.2	62.2 ± 2.2
Log-NCDE†	**35.9 ± 6.1**	57.2 ± 5.6	82.8 ± 2.7
LRU†	23.8 ± 2.8	51.9 ± 8.6	**85.0 ± 6.2**
S5†	25.6 ± 3.5	53.0 ± 3.9	83.9 ± 4.1
Mamba†	27.9 ± 4.5	47.7 ± 4.5	70.9 ± 15.8
S6†	26.4 ± 6.4	51.3 ± 4.7	**85.0 ± 16.1**
LinOSS-IMEX†	29.9 ± 1.0	**57.9 ± 5.3**	80.0 ± 2.7
LinOSS-IM†	29.9 ± 0.6	**60.0 ± 7.5**	**95.0 ± 4.4**
LrcSSM (Ours)	**36.9 ± 5.3**	**58.6 ± 3.1**	**90.6 ± 1.4**

Table 5. Average test accuracy (%) across all datasets. As before, the performance of the models marked by † is reported from (Rusch & Rus, 2024). Results are averaged over 5 seeds.

	Average Test Accuracy
NRDE†	60.2 ± 17.1
NCDE†	55.5 ± 18.2
Log-NCDE†	64.4 ± 16.9
LRU†	61.8 ± 22.6
S5†	63.1 ± 21.2
Mamba†	58.6 ± 18.8
S6†	62.0 ± 21.2
LinOSS-IMEX†	**65.0 ± 19.0**
LinOSS-IM†	**67.8 ± 21.6**
LrcSSM (Ours)	**66.3 ± 18.6**

Table 6. Hyperparameter grid. Same values as in (Walker et al., 2024; Rusch & Rus, 2024).

Parameter name	Value
learning rate	$10^{-5}, 10^{-4}, 10^{-3}$
hidden dimension	$16, 64, 128$
state-space dimension	$16, 64, 256$
number of blocks (#blocks)	$2, 4, 6$

H. Ablation Studies

Input- and State-dependency. We conducted ablation studies to assess the importance of incorporating state-dependency in the state-transition matrix $\mathbf{A}$ and input-transition vector $\mathbf{b}$. Given the extensive hyperparameter search required, we fixed the architecture to 6 layers of SSM blocks, each with 64 states, an input encoding dimension of 64, and a learning rate of 10^{-4}. As shown in Table 8, the average results indicate that learning both input- and state-dependent transitions yields better performance. We also suggest that future work could treat these dependencies as tunable hyperparameters, as some datasets may benefit from both forms of dependency, while others may perform well with input-dependency alone.

Please note that results reported here for LrcSSM, do not match the results of the previous tables because we used a fix setup without hyperparameter tuning, to only focus on the importance of state-dependency and changed the underlying matrix $\mathbf{A}$ and $\mathbf{b}$ of $\dot{x} = A(x, u)x + b(x, u)$. This results in having even better test accuracies reported here for Heartbeat and SelfRegulationSCP2.

Complex-valued State-Transition Matrix and Input-Transition Vector. We also experimented with complex-valued learnable parameters, focusing on those interacting directly with the state x. In particular,we experimented with the parameters $g_i^{max,x}$ of $f_i^*(x_i, u)$ and $k_i^{max,x}$ of $z_i^*(x_i, u)$ as defined in Eq.(5) and (6), respectively, as well as their shared sigmoidal channel parameters a_i^x and b_i^x. These were gradually converted to complex values, and experiments were conducted using a fixed configuration of 6 SSM blocks, each with 64 state dimensions and 64-dimensional encoded input, and a learning rate of 10^{-4}. As shown in Table 9, we found no significant performance gains on average from using complex-valued parameters. As a result, we opted to use real-valued learnable parameters in our main experiments. Nevertheless, we also evaluated the tuned models with their complex-valued counterparts. The only notable improvement occurred on the MotorImagery dataset, where

Table 7. Hyperparameters used for LrcSSM per dataset.

	lr	hidden dim.	state-space dim.	#blocks
Heartbeat	10^{-3}	64	64	4
SelfRegulationSCP1	10^{-3}	64	16	2
SelfRegulationSCP2	10^{-3}	128	64	2
EthanolConcentration	10^{-4}	128	16	2
MotorImagery	10^{-4}	16	16	4
EigenWorms	10^{-4}	16	16	4

Table 8. Experimentation with different input and state-dependent matrices. Here, we use a fixed configuration with input encoding of 64 and 6 blocks of SSMs with 64 units. We found that excluding state dependency from $\mathbf{A}$ and then from $\mathbf{b}$ too, downgrades performance on average.

	LrcSSM (default)	LrcSSM	LrcSSM
$\mathbf{A}$ dependence	$\mathbf{A}(\mathbf{x}, \mathbf{u})$	$\mathbf{A}(\mathbf{u})$	$\mathbf{A}(\mathbf{u})$
$\mathbf{b}$ dependence	$\mathbf{b}(\mathbf{x}, \mathbf{u})$	$\mathbf{b}(\mathbf{x}, \mathbf{u})$	$\mathbf{b}(\mathbf{u})$
Heartbeat	**75.0** ± **2.6**	**75.0** ± **1.8**	73.0 ± 2.7
SelfRegulationSCP1	84.8 ± 2.8	**85.0** ± **2.9**	83.1 ± 1.4
SelfRegulationSCP2	**55.4** ± **7.7**	49.6 ± 5.5	51.4 ± 2.9
EthanolConcentration	36.1 ± 1.1	**37.6** ± **3.9**	34.2 ± 2.9
MotorImagery	55.7 ± 4.1	**57.9** ± **2.9**	54.3 ± 6.0
EigenWorms	85.6 ± 5.4	85.0 ± 5.5	**86.7** ± **5.4**
Average	**65.4** ± **17.9**	65.0 ± 18.0	63.8 ± 18.7

Table 9. Experimentation with complex valued parameters. Here, we use a fixed configuration with input encoding of 64 and 6 blocks of SSMs with 64 units. We found very similar average performance between real-valued and complex-valued parameters.

	LrcSSM (default)	LrcSSM with	LrcSSM with	LrcSSM with
$g_i^{max,x}$	$\in \mathbb{R}$	$\in \mathbb{C}$	$\in \mathbb{C}$	$\in \mathbb{C}$
$k_i^{max,x}$	$\in \mathbb{R}$	$\in \mathbb{C}$	$\in \mathbb{C}$	$\in \mathbb{C}$
a_i^x	$\in \mathbb{R}$	$\in \mathbb{R}$	$\in \mathbb{C}$	$\in \mathbb{C}$
b_i^x	$\in \mathbb{R}$	$\in \mathbb{R}$	$\in \mathbb{R}$	$\in \mathbb{C}$
Heartbeat	75.0 ± 2.6	74.3 ± 5.2	73.75 ± 3.2	73.0 ± 4.0
SelfRegulationSCP1	84.8 ± 2.8	82.9 ± 2.7	83.1 ± 4.2	84.8 ± 2.2
SelfRegulationSCP2	55.4 ± 7.7	50.4 ± 4.4	53.6 ± 3.6	58.6 ± 3.5
EthanolConcentration	36.1 ± 1.1	41.8 ± 2.1	40.0 ± 4.5	42.1 ± 3.6
MotorImagery	55.7 ± 4.1	53.2 ± 2.6	53.9 ± 3.5	52.5 ± 4.3
EigenWorms	85.6 ± 5.4	85.0 ± 6.5	88.3 ± 5.7	86.1 ± 6.3
Average	65.4 ± 17.9	64.6 ± 16.8	65.4 ± 17.4	66.2 ± 16.4

accuracy increased from 54.3 ± 3.1 to 58.6 ± 3.1. Substituting this result into the average accuracy reported in Table 5 would yield 65.6 ± 18.9, which still ranks our model as the second-best overall.