# PARALLEL MEMRISTIVE-FRIENDLY RESERVOIRS FOR GRADIENT-FREE LONG-SEQUENCE CLASSIFICATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Can a highly efficient sequence model remain competitive when none of its feature-producing weights are optimized? We introduce memristive-friendly parallelized reservoirs (MARS), a deep reservoir architecture whose dynamics are derived from a potentiation–depression model of memristive devices. MARS removes the dense recurrent matrix, reformulates the resulting input-dependent recurrence as an associative parallel scan, and stacks fixed, randomly initialized blocks through subtractive skip connections. All reservoir and encoder weights are sampled once and remain frozen; only a linear readout is trained with ridge regression. Training therefore requires neither gradient descent nor backpropagation through the reservoir. Despite this restriction, MARS is competitive with fully trained sequence models, including Mamba and LinOSS, on six long-sequence time-series classification datasets. It attains the highest mean accuracy on two datasets and a higher mean accuracy than Mamba on five of six. For fixed hyperparameters, MARS trains in $0.2$–$1.6$ seconds. Based on the reported times, this is approximately $54$–$390\times$ faster than the fastest gradient-trained baseline with reported timing on each dataset and $201$–$13{,}780\times$ faster than Mamba. MARS also provides an approximately $21\times$ forward-pass speedup over an already lightweight echo state network on long sequences. These results show that deep, partially random dynamical systems can achieve competitive accuracy with exceptionally fast model fitting. Because its local state update is derived from memristive potentiation and depression, MARS also offers a starting point for future implementations on memristive and in-memory hardware.

## 1 INTRODUCTION

Modern sequence models obtain their representations through end-to-end, gradient-based optimization. Reservoir computing (RC) offers a fundamentally different paradigm. A fixed, randomly initialized dynamical system transforms the input sequence, while learning is confined to a linear readout (Nakajima & Fischer, 2021; Tanaka et al., 2019). This removes the need to backpropagate through the recurrent dynamics, but raises a consequential question. Can features produced by an untrained dynamical system remain competitive with representations learned by modern sequence models, particularly on long sequences?

Echo state networks (ESNs) established this readout-only approach as a lightweight alternative to fully trained recurrent neural networks (Jaeger, 2001; Lukoševičius, 2012). The memristive-friendly ESN (MF-ESN) extends it with a local state update derived from the potentiation and depression dynamics of memristive devices (Pistolesi et al., 2025b; Miranda et al., 2020; Milano et al., 2022b). This connection is appealing for future neuromorphic and in-memory implementations, but the recurrent computation remains sequential, limiting its scalability on modern parallel hardware. Meanwhile, structured sequence models such as S4, LRU, and Mamba have demonstrated that suitably structured recurrences can be both expressive and efficiently parallelized (Gu et al., 2022a; Orvieto et al., 2023; Gu & Dao, 2024).

We introduce memristive-friendly parallelized reservoirs (MARS), which combine the training simplicity and physics-inspired dynamics of reservoir computing with the parallel execution and depth of modern structured recurrent architectures. MARS removes the dense recurrent matrix from MF-ESN and rewrites the resulting input-dependent affine recurrence as an associative parallel scan



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.

(Blelloch, 1990). Subtractive skip connections then allow these fixed, randomly initialized reservoir blocks to be composed into deep architectures. Neither the reservoir nor the encoder is trained by gradient descent; in MARS-TC, even the temporal-convolution weights remain random and fixed. Only the final linear readout is fitted by ridge regression.

Our contributions are:

- We derive a parallel-scan formulation of memristive-inspired reservoir dynamics and introduce subtractive skip connections for constructing deep reservoirs without backpropagation through their feature-producing layers.
- We show that parallelization provides an approximately $21\times$ forward-pass speedup over a classical ESN for sequence lengths from $1{,}000$ to $500{,}000$, despite the ESN itself already being a lightweight model.
- We demonstrate that this partially random architecture remains competitive with fully trained sequence models. MARS achieves the highest mean accuracy on six of seven short-sequence reservoir benchmarks; on six long-sequence datasets, MARS or MARS-TC attains the highest mean accuracy on two and exceeds Mamba's mean accuracy on five. For fixed hyperparameters, its $0.2$–$1.6$ second training times are approximately $54$–$390\times$ shorter than those of the fastest gradient-trained baseline with reported timing on each dataset.

Taken together, these results identify a striking regime in which deep sequence classification can be both competitive and extremely fast without gradient-based tuning of the feature-producing weights. Because the local dynamics remain derived from a memristive device model, MARS also provides a concrete algorithmic starting point for future implementations on memristive hardware.

## 2 Background

This section reviews the three foundations of MARS. These are reservoir computing, memristive-inspired dynamics, and parallel recurrent computation.

**Reservoir Computing.** Reservoir computing processes a sequence with a fixed recurrent dynamical system and learns only a linear readout. The reservoir projects the input history into a high-dimensional state without gradient-based tuning of its internal weights. Echo state networks are the most widely studied realization of this principle (Jaeger, 2001; Lukoševičius, 2012). Given an input $\mathbf{x}_t \in \mathbb{R}^{N_x}$, the state $\mathbf{h}_t \in \mathbb{R}^{N_h}$ of a leaky ESN is updated according to

$$\mathbf{h}_{t+1} = (1-\alpha)\mathbf{h}_t + \alpha \tanh(\mathbf{W}^x \mathbf{x}_{t+1} + \mathbf{W}^h \mathbf{h}_t + \mathbf{b}) \tag{1}$$

where $\alpha$ is the leaking rate, $\mathbf{W}^x$ is the input matrix, $\mathbf{W}^h$ is the recurrent matrix, and $\mathbf{b}$ is the bias vector. A linear output is obtained from

$$\mathbf{y}_{t+1} = \mathbf{W}^y \mathbf{h}_{t+1}. \tag{2}$$

The readout matrix $\mathbf{W}^y$ is the only trained component and is typically fitted by ridge regression. The input and recurrent weights are sampled once and remain fixed. They are commonly scaled to place the reservoir in a stable fading-memory regime that preserves information from previous inputs.

Because the gradients are not propagated through the recurrent dynamics, reservoir computing avoids several optimization difficulties associated with backpropagation through time, including vanishing and exploding gradients (Pascanu et al., 2013). Its separation between a fixed dynamical system and a fitted readout has also motivated implementations on neuromorphic and low-power hardware (Zhang & Vargas, 2023).

Although the recurrent weights remain fixed, reservoir architecture and hyperparameters can still be selected by an outer optimization loop. Prior work has used neuroevolution to optimize recurrent reservoirs (Otte et al., 2016).

**Memristive Dynamics.** A memristive device has a conductance that depends on the history of the applied voltage or current. Its nonlinear state evolution and fading memory have motivated neuromorphic and reservoir-computing models. Recent work translates the potentiation and depression

dynamics of memristive nanowires into recurrent neural state updates (Pistolesi et al., 2025a;b). In the present work, these equations serve as a device-inspired software model and provide the physical basis for a potential future hardware mapping.

The internal state is governed by competing potentiation and depression rates

$$K_p(\mathbf{x}) = K_{p_0} \cdot e^{\eta_p \mathbf{x}} \tag{3}$$

$$K_d(\mathbf{x}) = K_{d_0} \cdot e^{-\eta_d \mathbf{x}}. \tag{4}$$

We use the physically motivated parameterization $K_{p_0} = 0.0001$, $\eta_p = 10$, $K_{d_0} = 0.5$, and $\eta_d = 1$ (Milano et al., 2022a). The potentiation rate increases exponentially with the stimulus, while the depression rate decreases. Their balance determines the direction and magnitude of the state change. The larger exponent in $K_p$ means that potentiation can rapidly dominate for sufficiently large inputs.

The time-discretized sequential update is

$$\mathbf{z}_{t+1} = \mathrm{RESCALE}(\mathbf{W}^h \mathbf{h}_t + \mathbf{W}^x \mathbf{x}_{t+1} + \mathbf{b}) \tag{5}$$

$$\mathbf{q}_{t+1} = (K_p(\mathbf{z}_{t+1}) + K_d(\mathbf{z}_{t+1})) \odot \mathbf{h}_t \tag{6}$$

$$\mathbf{r}_{t+1} = K_p(\mathbf{z}_{t+1}) - \mathbf{q}_{t+1} \tag{7}$$

$$\mathbf{h}_{t+1} = \mathbf{r}_{t+1}\Delta + \gamma \mathbf{h}_t. \tag{8}$$

Here, $\mathbf{W}^h \in \mathbb{R}^{N_h \times N_h}$ is the reservoir recurrent matrix, $\mathbf{W}^x \in \mathbb{R}^{N_h \times N_x}$ is the input matrix, and $\mathbf{b} \in \mathbb{R}^{N_h}$ is the bias vector. The operator $\odot$ denotes elementwise multiplication. The parameter $\Delta$ controls the discretized state update, while $\gamma$ controls the contribution retained from the preceding state.

The pointwise $\mathrm{RESCALE}$ function bounds the input to the exponential rate equations and controls their numerical range. It is given by

$$\mathrm{RESCALE}(z) = \frac{b-a}{1+\exp(-zs)} + a. \tag{9}$$

The lower and upper bounds are fixed to $a = 0.35$ and $b = 1.15$ following Pistolesi et al. (2025b). The hyperparameter $s$ controls the steepness of the transition between these bounds. Larger values of $s$ produce a sharper nonlinearity.

**Parallel Linear Recurrences.** Recent sequence models have shown that structured linear recurrences can remain expressive when they are embedded within deep architectures containing nonlinear transformations (Gu et al., 2022a; Orvieto et al., 2023; Gu & Dao, 2024; Feng et al., 2024). A time-invariant linear recurrence of the form $\mathbf{h}_{t+1} = \mathbf{W}^h \mathbf{h}_t + \mathbf{W}^x \mathbf{x}_{t+1}$ can be evaluated as a global convolution using the fast Fourier transform (Gu et al., 2021). Affine recurrences can instead be evaluated with a parallel prefix scan (Martin & Cundy, 2018; Qin et al., 2023). These approaches are particularly efficient when the recurrent transformation is structured. A diagonal transformation allows the state channels to be processed independently while retaining strong sequence-modeling performance (Gu et al., 2022a;b).

We consider the elementwise affine recurrence

$$\mathbf{h}_{t+1} = \mathbf{a}_{t+1} \odot \mathbf{h}_t + \mathbf{b}_{t+1}. \tag{10}$$

The affine transformations represented by the pairs $(\mathbf{a}_t, \mathbf{b}_t)$ compose associatively. A parallel prefix scan can therefore evaluate all states with logarithmic parallel depth and linear total work in the sequence length. We use the log-space construction of Heinsen (2023), which expresses the required cumulative products through cumulative sums.

$$\log \mathbf{h}_{t+1} = \log \mathbf{a}_{t+1} + \log \mathbf{h}_t + \log\left(1 + \exp(\log \mathbf{b}_{t+1} - \log \mathbf{a}_{t+1} - \log \mathbf{h}_t)\right) \tag{11}$$

By computing cumulative sum in the log-domain, the entire sequence $\{\mathbf{h}_t\}_{t=1}^T$ can be computed in parallel. Let:

$$\boldsymbol{\alpha}_t := \sum_{i=1}^{t} \log \mathbf{a}_i \quad \text{and} \quad \boldsymbol{\beta}_t := \log \mathbf{b}_t. \tag{12}$$

Then the parallel scan computes

$$\log \mathbf{h}_t = \boldsymbol{\alpha}_t + \log\left(\mathbf{h}_0 + \sum_{i=1}^{t} \exp(\boldsymbol{\beta}_i - \boldsymbol{\alpha}_i)\right) \tag{13}$$

and we recover the final output as

$$\mathbf{h}_t = \exp\left(\boldsymbol{\alpha}_t + \log\left(\mathbf{h}_0 + \sum_{i=1}^{t} \exp(\boldsymbol{\beta}_i - \boldsymbol{\alpha}_i)\right)\right) \tag{14}$$

This log-space computation, however, comes with a caveat: the inputs of the logarithm (i.e. $\mathbf{a}$ and $\mathbf{b}$) must be positive if the computation must stay in the real domain. Nonetheless, this is not a problem in our case as $K_d$ and $K_p$ are always positive due to the exponentiation.

## 3 Memristive-Friendly Parallelized Reservoirs (MARS)

MARS begins with the sequential memristive dynamics introduced in section 2. In the original formulation, $\mathbf{z}_{t+1}$ depends on the preceding hidden state through the dense recurrent matrix $\mathbf{W}^h$. Because $\mathbf{z}_{t+1}$ subsequently determines the nonlinear potentiation and depression rates, this dependency forces the time steps to be evaluated sequentially.

MARS removes the dense state-mixing path by setting $\mathbf{W}^h = \mathbf{0}$. Temporal memory is not removed. It remains in the elementwise recurrence of the memristive state $\mathbf{h}_t$. The input-dependent activations can now be computed independently for every time step as

$$\mathbf{z}_{[1:T]} = \mathrm{RESCALE}(\mathbf{W}^x \mathbf{x}_{[1:T]} + \mathbf{b}) \tag{15}$$

In this form, it is possible to apply the weighted sum and RESCALE for the whole sequence (denoted by $[1:T]$) at once. After the redefinition of $\mathbf{z}$, the sequential equations can be reformulated as follows:

$$\begin{aligned}
\mathbf{h}_{t+1} &= \mathbf{r}_{t+1}\Delta + \gamma \mathbf{h}_t \\
&= (K_p(\mathbf{z}_{t+1}) - \mathbf{q}_{t+1})\,\Delta + \gamma \mathbf{h}_t \\
&= (K_p(\mathbf{z}_{t+1}) - [K_p(\mathbf{z}_{t+1}) + K_d(\mathbf{z}_{t+1})] \odot \mathbf{h}_t)\,\Delta + \gamma \mathbf{h}_t \\
&= \Delta K_p(\mathbf{z}_{t+1}) - \Delta\,[K_p(\mathbf{z}_{t+1}) + K_d(\mathbf{z}_{t+1})] \odot \mathbf{h}_t + \gamma \mathbf{h}_t \\
&= (\gamma - \Delta\,[K_p(\mathbf{z}_{t+1}) + K_d(\mathbf{z}_{t+1})]) \odot \mathbf{h}_t + \Delta K_p(\mathbf{z}_{t+1})
\end{aligned} \tag{16}$$

The final formulation has exactly the same form as Equation 10, which allows the employment of parallel scan algorithm. Consequently, $\mathbf{a}_{t+1} := (\gamma - \Delta\,[K_p(\mathbf{z}_{t+1}) + K_d(\mathbf{z}_{t+1})])$ and $\mathbf{b}_{t+1} := \Delta K_p(\mathbf{z}_{t+1})$. Note that it is possible to compute $\mathbf{a}_{[1:T]}$ and $\mathbf{b}_{[1:T]}$ at once as they encode no recurrent relations. Computing these two terms and mapping them into the log-space yields a formulation that is well suited for parallel scan, with $\mathbf{a}_{[1:T]}$ and $\mathbf{b}_{[1:T]}$ as inputs. MARS preserves the memristive dynamics of MF-ESN, differing only in the absence of recurrent weights.

Parallel execution makes each MARS layer efficient, but depth is equally important for representational richness. As illustrated in Figure 1, the input sequence is first mapped by a fixed linear encoder and then processed by $N$ stacked memristive blocks. Within each block, the affine transformation, RESCALE, and the potentiation and depression rates are evaluated pointwise over the complete sequence. The elementwise affine recurrence is then evaluated by parallel scan.

The equations above describe one memristive block with a generic discretization step $\Delta$. For block $i$ in the stacked long-sequence models, every occurrence of $\Delta$ is evaluated with an independently selected $\Delta_i$. A shared step would impose the same compromise between numerical stability and temporal integration at every depth, even though the subtractive skip connections alter the frequency content and amplitude statistics passed to later blocks. Optuna (Akiba et al., 2019) selects each $\Delta_i$ jointly with the remaining hyperparameters, after which it remains fixed. The per-layer time steps therefore provide multiple dynamical time scales without introducing learned reservoir weights.

MARS combines each block input with its memristive response through a *subtractive skip connection*. Additive residual connections are commonly used to facilitate gradient propagation (He et al., 2016).

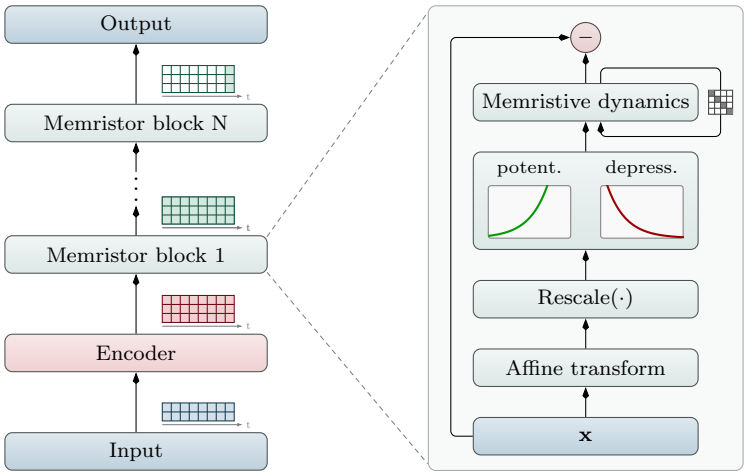


Figure 1: Overview of the MARS architecture. A fixed encoder is followed by stacked memristive blocks containing pointwise transformations, parallel recurrent computation, and subtractive skip connections. The final hidden state of the last block is mapped to the output by a ridge-trained linear readout. All other weights remain fixed after initialization.

Because the MARS feature-producing layers are not trained by backpropagation, the skip connection instead serves a signal-processing role. The memristive state can behave as a slowly varying, low-pass-like response because it accumulates and decays over time. Subtracting this response from the block input suppresses the slower component and produces a high-pass-like residual that emphasizes more rapid temporal variation. Repeating this operation across layers can progressively enrich the representation with higher-frequency components. A controlled illustration of this effect is provided in Appendix A.

## 4 Experiments

We evaluate MARS in two settings. First, we compare its forward-pass scaling and classification accuracy with classical ESN and sequential MF-ESN baselines. This places MARS within reservoir computing and tests whether its parallel formulation and deeper architecture preserve the efficiency of that paradigm. Second, we compare MARS with fully trained sequence models on six long-sequence datasets. This more demanding setting tests whether avoiding gradient-based feature learning imposes a substantial accuracy cost.

### 4.1 MARS within reservoir computing

**Runtime Analysis.** Removing the dense recurrent matrix $\mathbf{W}^h$ and evaluating the temporal recurrence with an associative scan allow MARS to exploit parallel hardware. We measure this advantage against a classical leaky ESN using randomly initialized input tensors sampled from $[0, 1]$, with sequence lengths ranging from $50$ to $500{,}000$. All measurements in this experiment were obtained on an NVIDIA GeForce RTX 4090 GPU. To isolate feature-extraction cost, we exclude readout fitting for both models. The three-layer MARS and single-layer ESN are configured to have roughly matched total parameter counts, making this a deliberately demanding comparison against an already lightweight baseline.

As shown in Figure 2 and Table 1, MARS is approximately $21\times$ faster than the ESN throughout the range from $1{,}000$ to $500{,}000$ time steps. At the longest sequence length, processing $100$ batches takes $79.6$ seconds with MARS and $1{,}669.4$ seconds with the ESN. The associative scan has logarithmic parallel depth and linear total work in the sequence length; the measured wall-clock results should therefore be interpreted as hardware-dependent empirical runtimes, rather than as evidence of logarithmic wall-clock scaling.

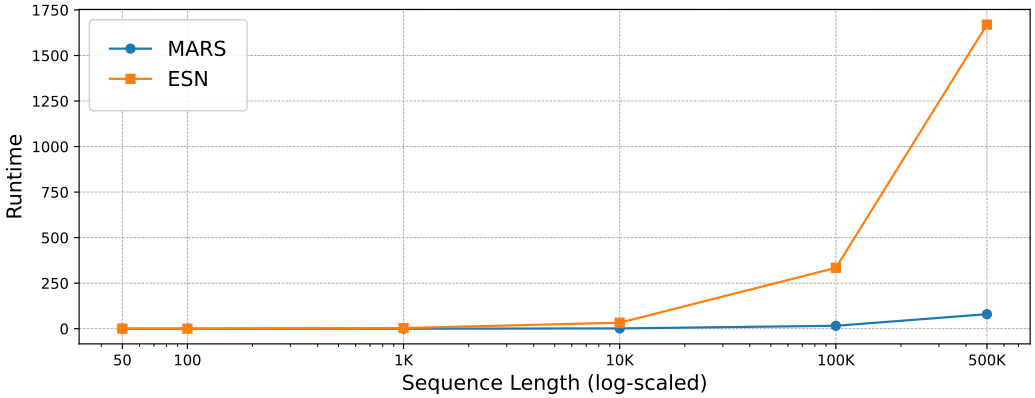


Figure 2: Wall-clock forward-pass runtime of MARS and ESN as a function of sequence length. Reported values are the total time for 100 forward passes with a batch size of 10; the sequence-length axis is logarithmically scaled. For lengths from 1,000 to 500,000, MARS is consistently approximately $21\times$ faster than the ESN.

Table 1: Total wall-clock time in seconds for 100 forward passes with a batch size of 10. At a sequence length of 500,000, MARS requires 79.6 seconds (1.3 minutes), compared with 1,669.4 seconds (27.8 minutes) for the ESN, corresponding to a $21.0\times$ speedup.

| **Model** | 50 | 100 | $1k$ | $10k$ | $100k$ | $500k$ |
|---|---|---|---|---|---|---|
| **MARS** | 0.0995 | 0.1012 | 0.1622 | 1.5079 | 15.9531 | 79.6439 |
| **ESN** | 0.2169 | 0.3835 | 3.4795 | 32.7758 | 334.0719 | 1,669.3938 |

**Time Series Classification.** We benchmark MARS against classical ESN and sequential MF-ESN models on seven short-sequence classification tasks from the UCR archive (Dau et al., 2019). These datasets cover several application domains and have sequence lengths of up to 286 time steps. Dataset statistics are provided in Table 6.

The models have roughly matched combined counts of fixed and trainable parameters. ESN and MF-ESN results are taken from Pistolesi et al. (2025b). We select the MARS input scaling, bias scaling, $\Delta$, $\gamma$, and $s$ using a procedure similar to that work. The selected values are reported in Table 7.

MARS achieves the highest mean accuracy on six of the seven tasks. On the remaining Epilepsy task, its mean accuracy is only 0.5 percentage points below that of MF-ESN. The largest improvements over the sequential MF-ESN occur on GunPoint and Coffee, where MARS gains 18.7 and 41.4 percentage points, respectively. The removal of the dense recurrent matrix therefore does not coincide with an accuracy penalty in this benchmark.

Each MARS recurrence is linear after its input-dependent coefficients have been computed. Depth, nonlinear rescaling, and subtractive skip connections nevertheless allow the model to construct a rich hierarchy of fixed features. This is consistent with the broader finding that deep compositions of structured linear recurrences and nonlinear transformations can be highly expressive (Gu et al., 2022a; Orvieto et al., 2023; Gu & Dao, 2024). The present comparison does not isolate the contribution of each architectural component, but it shows that their combination can outperform conventional recurrent reservoirs without gradient-based tuning of the feature-producing weights.

### 4.2 MARS AGAINST GRADIENT-TRAINED SEQUENCE MODELS

Comparing a reservoir network with fully trained sequence models is deliberately demanding. Gradient-based models adapt their feature-producing weights end-to-end, whereas MARS keeps all such weights fixed after random initialization and fits only the linear readout. Under this restriction,

Table 2: Test accuracy reported as mean and standard deviation on seven short-sequence classification tasks. ESN and sequential MF-ESN results are taken from Pistolesi et al. (2025b). MARS results are ours. **The highest mean accuracy** on each task is shown in bold.

| **Task** | ESN | MF-ESN | MARS |
|---|---|---|---|
| **Epilepsy** | $0.868 \pm 0.015$ | $\mathbf{0.961 \pm 0.007}$ | $0.956 \pm 0.008$ |
| **SyntheticControl** | $0.893 \pm 0.010$ | $0.950 \pm 0.007$ | $\mathbf{0.980 \pm 0.004}$ |
| **GunPoint** | $0.593 \pm 0.023$ | $0.769 \pm 0.070$ | $\mathbf{0.956 \pm 0.003}$ |
| **ECG5000** | $0.915 \pm 0.002$ | $0.924 \pm 0.002$ | $\mathbf{0.939 \pm 0.001}$ |
| **Coffee** | $0.586 \pm 0.029$ | $0.586 \pm 0.017$ | $\mathbf{1.000 \pm 0.000}$ |
| **JapaneseVowels** | $0.984 \pm 0.003$ | $0.974 \pm 0.004$ | $\mathbf{0.989 \pm 0.003}$ |
| **Wafer** | $0.986 \pm 0.002$ | $0.986 \pm 0.001$ | $\mathbf{0.996 \pm 0.001}$ |

Table 3: Test accuracy, reported as mean and standard deviation over five runs, on six long-sequence UEA classification datasets. Dataset abbreviations are EigenWorms (Worms), SelfRegulationSCP1 (SCP1), SelfRegulationSCP2 (SCP2), EthanolConcentration (Ethanol), Heartbeat, and MotorImagery (Motor). Baseline results are taken from Rusch & Rus (2025); MARS and MARS-TC results are ours. All results use the same data splits and random seeds. **The highest mean accuracy** on each dataset is shown in bold.

| | Worms | SCP1 | SCP2 | Ethanol | Heartbeat | Motor |
|---|---|---|---|---|---|---|
| Seq. length | 17,984 | 896 | 1,152 | 1,751 | 405 | 3,000 |
| #Classes | 5 | 2 | 2 | 4 | 2 | 2 |
| NRDE | $83.9 \pm 7.3$ | $80.9 \pm 2.5$ | $53.7 \pm 6.9$ | $25.3 \pm 1.8$ | $72.9 \pm 4.8$ | $47.0 \pm 5.7$ |
| NCDE | $75.0 \pm 3.9$ | $79.8 \pm 5.6$ | $53.0 \pm 2.8$ | $29.9 \pm 6.5$ | $73.9 \pm 2.6$ | $49.5 \pm 2.8$ |
| Log-NCDE | $85.6 \pm 5.1$ | $83.1 \pm 2.8$ | $53.7 \pm 4.1$ | $34.4 \pm 6.4$ | $75.2 \pm 4.6$ | $53.7 \pm 5.3$ |
| LRU | $87.8 \pm 2.8$ | $82.6 \pm 3.4$ | $51.2 \pm 3.6$ | $21.5 \pm 2.1$ | $\mathbf{78.4 \pm 6.7}$ | $48.4 \pm 5.0$ |
| S5 | $81.1 \pm 3.7$ | $89.9 \pm 4.6$ | $50.5 \pm 2.6$ | $24.1 \pm 4.3$ | $77.7 \pm 5.5$ | $47.7 \pm 5.5$ |
| S6 | $85.0 \pm 16.1$ | $82.8 \pm 2.7$ | $49.9 \pm 9.4$ | $26.4 \pm 6.4$ | $76.5 \pm 8.3$ | $51.3 \pm 4.7$ |
| Mamba | $70.9 \pm 15.8$ | $80.7 \pm 1.4$ | $48.2 \pm 3.9$ | $27.9 \pm 4.5$ | $76.2 \pm 3.8$ | $47.7 \pm 4.5$ |
| LinOSS-IMEX | $80.0 \pm 2.7$ | $87.5 \pm 4.0$ | $\mathbf{58.9 \pm 8.1}$ | $29.9 \pm 1.0$ | $75.5 \pm 4.3$ | $57.9 \pm 5.3$ |
| LinOSS-IM | $\mathbf{95.0 \pm 4.4}$ | $87.8 \pm 2.6$ | $58.2 \pm 6.9$ | $29.9 \pm 0.6$ | $75.8 \pm 3.7$ | $\mathbf{60.0 \pm 7.5}$ |
| **MARS** | $78.3 \pm 6.3$ | $89.2 \pm 3.3$ | $50.7 \pm 6.6$ | $\mathbf{36.7 \pm 4.0}$ | $67.4 \pm 8.0$ | $52.6 \pm 7.1$ |
| **MARS-TC** | $77.8 \pm 7.4$ | $\mathbf{90.1 \pm 2.8}$ | $57.9 \pm 6.7$ | $36.2 \pm 6.2$ | $73.2 \pm 11.2$ | $55.8 \pm 5.5$ |

one would not ordinarily expect a model to match strongly optimized sequence architectures, particularly on long sequences. What makes the comparison especially unusual is that MARS completes its final fitting in seconds rather than the minutes or hours required by the gradient-trained models. The experiment therefore asks how much gradient-based representation learning is actually necessary for competitive long-sequence classification.

We consider a recent line of work on recurrent and state-space sequence models. Walker et al. (2024) introduced Log-NCDE and compared it with architectures including S4, S5, Mamba, and LRU (Gu et al., 2022a; Smith et al., 2023; Gu & Dao, 2024; Orvieto et al., 2023). Their benchmark comprises six challenging long-sequence datasets from the UEA Multivariate Time Series Classification Archive (Bagnall et al., 2018). Rusch & Rus (2025) subsequently reused the same experimental design and added their linear oscillatory state-space models to the comparison.

We follow the experimental setting of Walker et al. (2024), including the data splits, removal of repeated sequences, and random seeds. The implementation framework is the only procedural difference. Our code uses PyTorch rather than JAX (Paszke et al., 2019; Bradbury et al., 2018).

We additionally evaluate MARS-TC, which prepends a one-dimensional temporal convolution to the encoder (Kalchbrenner et al., 2016). The finite convolutional kernels provide fixed local filters that can expose short-range and higher-frequency temporal variation, while the subsequent recurrent

Table 4: Training time in seconds for one selected model configuration. Baseline timings are reported by Walker et al. (2024); MARS timings are ours. MARS is approximately $54$–$390\times$ faster than the fastest gradient-trained baseline with reported timing on each dataset.

| **Model** | Worms | SCP1 | SCP2 | Ethanol | Heartbeat | Motor |
|---|---|---|---|---|---|---|
| **NRDE** | 8,402 | 1,947 | 2,331 | 2,932 | 13,927 | 14,166 |
| **NCDE** | 27,055 | 1,595 | 1,501 | 3,015 | 2,049 | 4,685 |
| **Log-NCDE** | 3,365 | 2,260 | 992 | 3,413 | 1,587 | 1,153 |
| **LRU** | 1,384 | 139 | 132 | 195 | 100 | 846 |
| **S5** | 478 | 224 | 227 | 114 | 208 | 271 |
| **Mamba** | 2,321 | 281 | 1,256 | 4,134 | 605 | 726 |
| **S6** | 2,465 | 76 | 168 | 102 | 78 | 580 |
| **MARS** | **1.6** | **1.4** | **0.4** | **0.3** | **0.2** | **0.8** |

reservoir can propagate these features through potentially long-ranging hidden-state dynamics. This combination expands the temporal representation without introducing gradient-based feature learning. The number of convolutional output channels, kernel size, and stride are selected independently for each dataset by Optuna. The convolutional and encoder weights are then randomly initialized and remain fixed. Only the final linear readout is trained.

The results in Table 3 place MARS in an unusual performance regime. MARS or MARS-TC attains the highest mean accuracy on SelfRegulationSCP1 and EthanolConcentration and exceeds Mamba's mean accuracy on five of the six datasets. The striking point is not uniform dominance, but that a deep, partially random model with only a ridge-trained readout occupies the same accuracy range as end-to-end optimized sequence models.

EigenWorms, with a sequence length of $17{,}984$, exposes the clearest limitation. MARS exceeds Mamba by $7.4$ percentage points in mean accuracy ($78.3\%$ versus $70.9\%$) but remains well below the strongest result of $95.0\%$. We hypothesize that, over extremely long sequences, the memristive state can settle early into a potentiation- or depression-dominated regime and consequently lose information from earlier inputs. Establishing this mechanism will require targeted analysis.

The computational contrast is clearest in the training times and trainable parameter counts reported in Table 4 and Table 5. Once a configuration has been selected, training MARS consists of evaluating the fixed feature-producing network over the training data and fitting the linear readout by ridge regression. None of the weights in its 2–6 reservoir layers is updated iteratively, and no gradient is propagated through these layers.

Model fitting must be distinguished from hyperparameter optimization. For each long-sequence dataset, we select the architecture and hyperparameters with Optuna using the Tree-structured Parzen Estimator sampler (Akiba et al., 2019). Each study runs for $300$ trials and maximizes the mean validation accuracy across five fixed seeds. Every trial evaluates a fresh model instantiation for each seed, but still fits only the ridge readout. Optuna therefore searches over architectural and dynamical choices without introducing gradient-based tuning of the feature-producing weights. It should be noted that the gains from Optuna over the initially selected manual configurations were generally moderate. Strong performance was already obtained with the manual settings, so the competitive results are not primarily attributable to the use of hyperparameter search. The full search space and evaluation protocol are reported in Appendix C. The selected dataset-specific configurations are reported in Appendix D.

As for the reported baselines, the times in Table 4 describe training one selected configuration and exclude hyperparameter-search cost. Under this protocol, MARS trains in $0.2$–$1.6$ seconds. This is approximately $54$–$390\times$ faster than the fastest gradient-trained baseline with reported timing on each dataset and $201$–$13{,}780\times$ faster than Mamba.

The trainable parameter count provides a second view of the optimization burden. Across the six datasets, MARS fits only $401$ to $1{,}604$ readout parameters. This represents a $3.6\times$ to $31\times$ reduction relative to the smallest gradient-trained model listed for each dataset and an $18\times$ to $2{,}579\times$ reduction relative to Mamba. These counts describe the number of fitted parameters rather than the total model

Table 5: Number of trainable parameters on the long-sequence datasets. Baseline counts are taken from Rusch & Rus (2025), while MARS counts are ours. For MARS and MARS-TC, only the fitted linear readout is counted. Their fixed encoder, reservoir, and temporal-convolution weights are excluded. **The smallest trainable parameter count** on each dataset is shown in bold.

| **Dataset** | Log-NCDE | LRU | S5 | Mamba | S6 | LinOSS-IMEX | MARS |
|---|---|---|---|---|---|---|---|
| **Worms** | 37,977 | 101,129 | 22,007 | 27,381 | 15,045 | 26,119 | **2,345** |
| **SCP1** | 91,557 | 25,892 | 226,328 | 184,194 | 24,898 | 447,944 | **463** |
| **SCP2** | 36,379 | 26,020 | 5,652 | 356,290 | 26,018 | 448,072 | **258** |
| **Ethanol** | 31,452 | 76,522 | 76,214 | 1,032,772 | 5,780 | 70,088 | **2,088** |
| **Heartbeat** | 168,320 | 338,820 | 158,310 | 1,034,242 | 6,674 | 29,444 | **138** |
| **Motor** | 81,391 | 107,544 | 17,496 | 228,226 | 52,802 | 106,024 | **541** |

size. The randomly initialized encoder, reservoir, and temporal-convolution weights remain fixed and are therefore not included.

## 5 Conclusion

We introduced MARS, a deep memristive-inspired reservoir architecture that removes the dense recurrent matrix and evaluates its input-dependent recurrence with an associative parallel scan. Fixed, randomly initialized blocks are composed through nonlinear transformations and subtractive skip connections. Only the final linear readout is fitted by ridge regression. The feature-producing weights require neither gradient-based tuning nor backpropagation through the reservoir.

This restriction does not prevent MARS from reaching the accuracy range of fully trained sequence models. MARS achieves the highest mean accuracy on six of seven short-sequence reservoir benchmarks. On the six long-sequence datasets, MARS or MARS-TC achieves the highest mean accuracy on two and exceeds Mamba's mean accuracy on five. For fixed hyperparameters, training takes only $0.2$ to $1.6$ seconds. This is approximately $54$ to $390\times$ faster than the fastest gradient-trained baseline with reported timing on each dataset and $201$ to $13{,}780\times$ faster than Mamba. MARS also provides an approximately $21\times$ forward-pass speedup over a classical ESN on long sequences while fitting only $401$ to $1{,}604$ parameters. Taken together, these results show that competitive long-sequence representations can emerge from deep, partially random dynamics without gradient-based optimization of the feature extractor.

Several questions remain open. The current evaluation is limited to classification, depends on problem-specific hyperparameter selection, and does not isolate the individual contributions of depth, nonlinear rescaling, subtractive skip connections, and temporal convolution. Performance on EigenWorms also indicates that the present dynamics can lose information over extremely long sequences. Future work should address these limitations through targeted ablations and broader sequence tasks.

The present results were obtained in software and do not demonstrate the energy or latency of a physical memristive implementation. Nevertheless, the local potentiation and depression update remains derived from a memristive device model, while the feature-producing weights remain fixed after initialization. If suitable hardware becomes available, these properties make MARS a concrete candidate for future mapping to memristor and in-memory chips. Evaluating such a mapping is an important next step.

### AI use statement

Generative AI tools were used only to assist with language editing and writing consistency. All AI-assisted text was reviewed and verified by the authors. Generative AI was not used to develop the research questions, methodology, experiments, or scientific conclusions. The authors take full responsibility for the manuscript.

### Ethics statement

This work develops computational methods for efficient time-series modeling using memristive-inspired reservoir computing and does not involve new human-subject research or the collection of personal or sensitive data by the authors. Our experiments use established, publicly available benchmark datasets. We are not aware of direct applications of the proposed method that raise concerns beyond those generally associated with machine-learning sequence models. The authors have read and adhered to the ICLR Code of Ethics.

### Reproducibility statement

The model definition and parallel recurrence underlying MARS are specified in Sections 2 and 3, including the update equations and architectural components. Section 4 describes the experimental settings, comparison protocols, runtime measurements, and implementation details. The runtime hardware is reported alongside Table 1, and the long-sequence classification results in Table 3 are averaged over five runs. For the long-sequence benchmarks, we follow the referenced data-splitting, preprocessing, and random-seed protocol, with the stated implementation difference that our experiments use PyTorch rather than JAX. Appendix B reports the characteristics of the short-sequence datasets and the selected MARS hyperparameters. Appendix C reports the Optuna search space, fixed parameters, five evaluation seeds, and final evaluation procedure for the long-sequence experiments. Appendix D reports the selected dataset-specific configurations. The tables identify results taken from prior work and report predictive performance, runtime, and parameter counts. Together, these details are intended to support independent reproduction of the reported results. An implementation and scripts for reproducing the experiments will be included upon acceptance.

## References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.

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. *arXiv preprint arXiv:1811.00075*, 2018.

Guy E Blelloch. Prefix sums and their applications. In *Sythesis of parallel algorithms*, pp. 35—60. Morgan Kaufmann Publishers Inc., 1990.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were rnns all we needed? *arXiv preprint arXiv:2410.01201*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers. In *NeurIPS*. Curran Associates Inc., 2021.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022a.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022b. Curran Associates Inc. ISBN 9781713871088.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Franz A. Heinsen. Efficient parallelization of a ubiquitous sequential computation, 2023.

H. Jaeger. The echo state approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade: Second Edition*, pp. 659–686. Springer, 2012.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations (ICLR)*, 2018.

Gianluca Milano, Enrique Miranda, and Carlo Ricciardi. Connectome of memristive nanowire networks through graph theory. *Neural Networks*, 150:137–148, 2022a.

Gianluca Milano, Giacomo Pedretti, Kevin Montano, Saverio Ricci, Shahin Hashemkhani, Luca Boarino, Daniele Ielmini, and Carlo Ricciardi. In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks. *Nature materials*, 21(2): 195–202, 2022b.

Enrique Miranda, Gianluca Milano, and Carlo Ricciardi. Modeling of short-term synaptic plasticity effects in zno nanowire-based memristors using a potentiation-depression rate balance equation. *IEEE Transactions on Nanotechnology*, 19:609–612, 2020.

Kohei Nakajima and Ingo Fischer. *Reservoir computing*. Springer, 2021.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023. URL https://arxiv.org/abs/2303.06349.

Sebastian Otte, Martin V. Butz, Danil Koryakin, Fabian Becker, Marcus Liwicki, and Andreas Zell. Optimizing recurrent reservoirs with neuro-evolution. *Neurocomputing*, 192:128–138, 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2016.01.088.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Veronica Pistolesi, Andrea Ceni, Gianluca Milano, Carlo Ricciardi, and Claudio Gallicchio. A model of memristive nanowire neuron for recurrent neural networks. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 479–484, Bruges, Belgium and online, 2025a. ESANN 2025, 23–25 April 2025.

Veronica Pistolesi, Andrea Ceni, Gianluca Milano, Carlo Ricciardi, and Claudio Gallicchio. A memristive computational neural network model for time-series processing. *APL Machine Learning*, 3 (1):016117, mar 2025b. ISSN 2770-9019. doi: 10.1063/5.0255168.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Neural Information Processing Systems (NeurIPS)*, 2023.

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

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

Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

Benjamin Walker, Andrew D. McLeod, Tiexin Qin, Yichuan Cheng, Haoliang Li, and Terry Lyons. Log neural controlled differential equations: The lie brackets make a difference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, Vienna, Austria, 2024. PMLR.

Heng Zhang and Danilo Vasconcellos Vargas. A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning. *IEEE Access*, 11:81033–81070, 2023.

## A ILLUSTRATION OF SUBTRACTIVE SKIP CONNECTIONS

Figure 3 illustrates the effect of subtractive skip connections in a controlled setting. A sinusoidal signal with weak noise is processed without affine transformations or rescaling. The slowly varying memristive response is subtracted from the input at each layer. The resulting residual contains increasingly pronounced higher-frequency components across three successive layers.

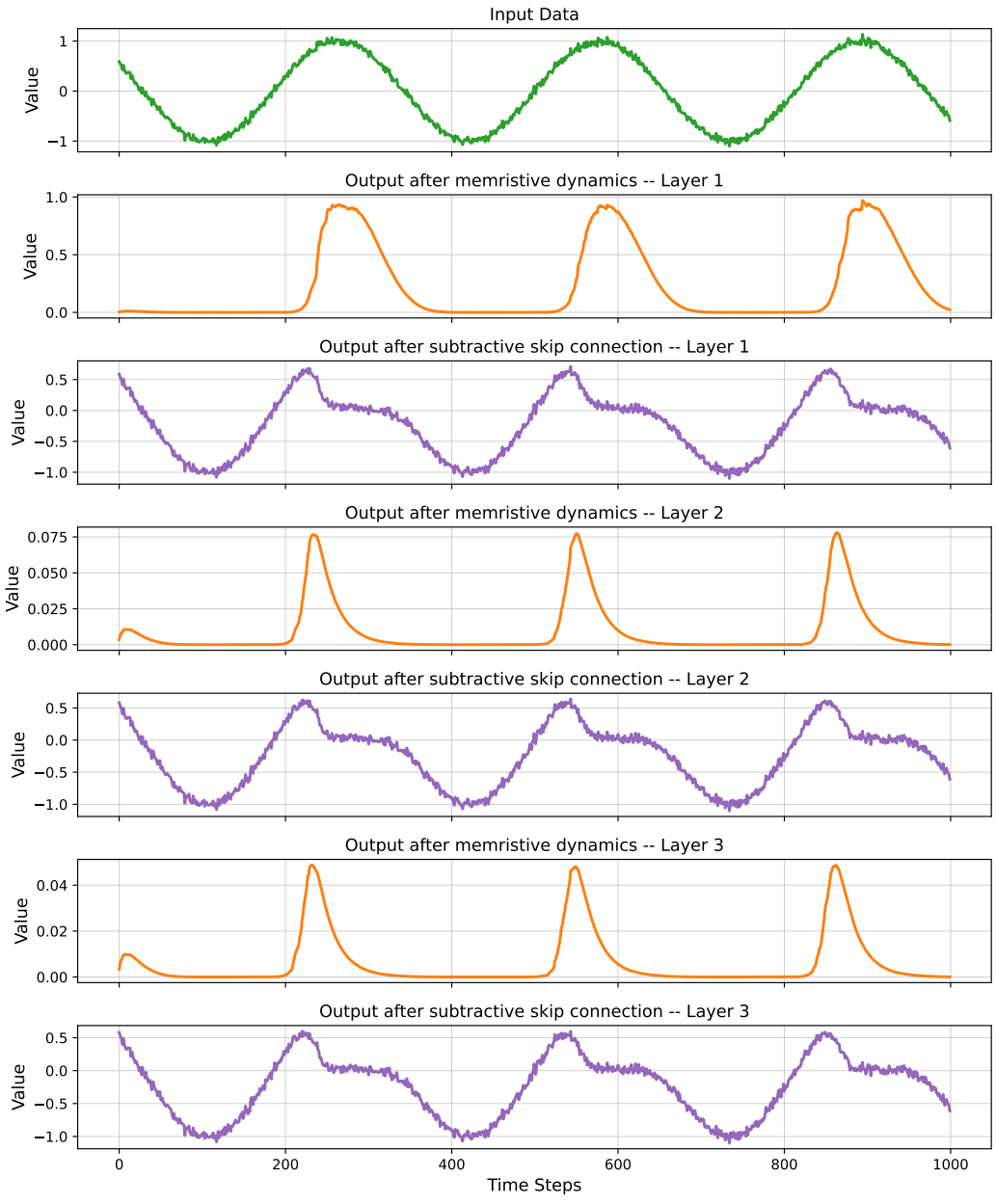


Figure 3: Toy illustration of the high-pass-like effect produced by subtractive skip connections. Subtracting the slowly varying memristive response leaves a residual with increasingly pronounced higher-frequency components in deeper layers.

## B SHORT-SEQUENCE DATASETS AND HYPERPARAMETERS

Table 6: Characteristics of the seven short-sequence classification datasets used in subsection 4.1. Train Size and Test Size give the number of sequences in each split. Length is the maximum sequence length, Classes is the number of output classes, Input Dim. is the number of input features, and Type describes the application domain.

| **Dataset** | Train Size | Test Size | Length | Classes | Input Dim. | Type |
|---|---|---|---|---|---|---|
| **Epilepsy** | 137 | 138 | 207 | 4 | 3 | HAR |
| **SyntheticControl** | 300 | 300 | 60 | 6 | 1 | Simulated |
| **GunPoint** | 50 | 150 | 150 | 2 | 1 | HAR |
| **ECG5000** | 500 | 4,500 | 140 | 5 | 1 | ECG |
| **Coffee** | 28 | 28 | 286 | 2 | 1 | Spectro |
| **JapaneseVowels** | 270 | 370 | 29 | 9 | 12 | Audio |
| **Wafer** | 1,000 | 6,164 | 152 | 2 | 1 | Sensor |

Table 7: Selected MARS hyperparameters for the seven short-sequence classification tasks in subsection 4.1. These values were chosen manually by trial and error and do not describe the long-sequence experiments. The parameters $\omega$ and $\beta$ denote input and bias scaling. ESN and MF-ESN hyperparameters are reported by Pistolesi et al. (2025b).

| **Dataset** | $\omega$ | $\beta$ | $\gamma$ | $\Delta$ | $s$ |
|---|---|---|---|---|---|
| **Epilepsy** | 0.1 | 0.1 | 1.0 | 0.05 | 5 |
| **SyntheticControl** | 0.1 | 0.5 | 1.0 | 0.1 | 6 |
| **GunPoint** | 0.1 | 0.1 | 1.0 | 0.1 | 6 |
| **ECG5000** | 0.1 | 0.2 | 1.0 | 0.08 | 6 |
| **Coffee** | 0.1 | 0.1 | 1.0 | 0.1 | 5 |
| **JapaneseVowels** | 0.1 | 0.1 | 1.0 | 0.5 | 1 |
| **Wafer** | 0.1 | 0.1 | 1.0 | 0.05 | 14 |

## C LONG-SEQUENCE HYPERPARAMETER OPTIMIZATION

**Optimization protocol.** Following Walker et al. (2024), the original training and test cases are pooled and randomly resplit into training, validation, and test subsets using a $70/15/15$ proportion. Repeated EigenWorms sequences are removed before splitting. We use the five fixed seeds $\{2345, 3456, 4567, 5678, 6789\}$. Each seed controls both the data split and the initialization of the fixed reservoir and encoder weights. For MARS-TC, it also controls the initialization of the temporal-convolution weights.

For each long-sequence dataset, we run 300 trials with Optuna's Tree-structured Parzen Estimator sampler (Akiba et al., 2019). Each trial instantiates the proposed configuration independently under all five seeds. For every seed, the linear readout is fitted on the $70\%$ training subset and evaluated on the $15\%$ validation subset. The trial objective is the arithmetic mean of the five validation accuracies. The test subsets remain untouched during hyperparameter selection. Every trial is evaluated under all five seeds before its objective value is returned.

The sparsity hyperparameter acts only on the fixed reservoir weight tensors, while all biases remain dense. For a configuration containing $N$ reservoir blocks, Optuna samples $N$ independent time steps $\Delta_1, \ldots, \Delta_N$. These values are selected jointly with the remaining hyperparameters and remain fixed during final model fitting.

**Fixed parameters.** The physical parameters remain fixed at $K_{p_0} = 0.0001$, $\eta_p = 10$, $K_{d_0} = 0.5$, and $\eta_d = 1$. The RESCALE bounds remain fixed at $a = 0.35$ and $b = 1.15$, and the stabilizer is fixed at $\gamma = 1$. None of these quantities is exposed to Optuna.

Table 8: Optuna search space for the long-sequence experiments. A separate study is run for each dataset. Integer ranges include both endpoints. The final three rows apply only to MARS-TC.

| **Hyperparameter** | **Range** | **Scale** |
|---|---|---|
| Reservoir size $N_h$ | $[100, 600]$ | Linear (int.) |
| Number of layers $N$ | $[2, 6]$ | Linear (int.) |
| Ridge regularization $\alpha$ | $[10^{-4}, 10]$ | Log |
| Input scaling $\omega$ | $[0.01, 2.0]$ | Log |
| Bias scaling $\beta$ | $[0.01, 2.0]$ | Log |
| Encoder weight scale | $[0.01, 2.0]$ | Log |
| Encoder bias scale | $[0.01, 2.0]$ | Log |
| Reservoir-weight sparsity | $[0.0, 0.99]$ | Linear |
| RESCALE slope $s$ | $[2, 15]$ | Linear (int.) |
| Per-layer time step $\Delta_i$ | $[10^{-7}, 10^{-1}]$ | Log |
| *Additional MARS-TC hyperparameters* | | |
| Temporal-convolution output channels | $[8, 64]$ | Linear (int.) |
| Temporal-convolution kernel size | $\{3, 5, 7, 9, 11\}$ | Categorical |
| Temporal-convolution stride | $[1, 5]$ | Linear (int.) |

**Final evaluation.** After the search terminates, the best-performing configuration is instantiated again under the same five seeds and evaluated on the held-out test subset. We report the mean and standard deviation across these five test runs.

## D SELECTED LONG-SEQUENCE HYPERPARAMETERS

The final hyperparameter values for the proposed models are reported below.

Table 9: Optimized reservoir and readout hyperparameters for MARS (no temporal-convolutional encoder) on each of the six benchmark datasets. Values are the best configurations returned by the respective Optuna study. $N_h$ is the reservoir size per layer and $N$ is the number of stacked memristive layers. The total number of frozen reservoir weights therefore scales as $N \cdot N_h$. $\alpha$ is the Tikhonov regularizer of the ridge readout. Per-layer time steps $\Delta_i$ are reported separately in Table 10.

| Dataset | $N_h$ | $N$ | $\alpha$ | $\omega$ | $\beta$ | enc$_W$ | enc$_b$ | sparsity | $s$ |
|---|---|---|---|---|---|---|---|---|---|
| EigenWorms | 468 | 2 | $1.93\times10^{-2}$ | 0.244 | 0.084 | 0.148 | 0.670 | 0.630 | 13 |
| SCP1 | 462 | 5 | $9.43\times10^{-2}$ | 0.038 | 1.073 | 0.072 | 0.217 | 0.306 | 14 |
| SCP2 | 257 | 6 | $1.62\times10^{-4}$ | 0.252 | 0.206 | 0.670 | 0.176 | 0.160 | 15 |
| Ethanol | 521 | 4 | 2.794 | 0.050 | 1.619 | 0.017 | 1.039 | 0.671 | 14 |
| Heartbeat | 137 | 2 | $4.71\times10^{-2}$ | 0.012 | 0.016 | 0.312 | 0.181 | 0.073 | 14 |
| Motor | 540 | 2 | $8.74\times10^{-4}$ | 0.135 | 0.048 | 0.129 | 0.119 | 0.290 | 7 |

Table 10: Per-layer time step $\Delta_i$ for MARS, as selected by Optuna. Column $\Delta_i$ gives the value used at the $i$-th memristive layer, indexed from the layer closest to the input. A dash (—) indicates that the dataset's optimal architecture uses fewer than $i$ layers (cf. column $N$ of Table 9) and therefore has no $\Delta_i$. Within a single dataset the selected $\Delta_i$ typically span several orders of magnitude, showing that different layers benefit from qualitatively different discretization steps and motivating the per-layer treatment of $\Delta$.

| Dataset | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ | $\Delta_4$ | $\Delta_5$ | $\Delta_6$ |
|---|---|---|---|---|---|---|
| EigenWorms | $4.14\times10^{-5}$ | $5.27\times10^{-7}$ | — | — | — | — |
| SCP1 | $1.51\times10^{-5}$ | $1.18\times10^{-3}$ | $1.10\times10^{-4}$ | $5.38\times10^{-4}$ | $9.63\times10^{-4}$ | — |
| SCP2 | $2.15\times10^{-4}$ | $6.59\times10^{-6}$ | $1.00\times10^{-4}$ | $1.57\times10^{-3}$ | $1.25\times10^{-4}$ | $5.14\times10^{-3}$ |
| Ethanol | $4.10\times10^{-3}$ | $1.24\times10^{-3}$ | $2.41\times10^{-7}$ | $2.59\times10^{-4}$ | — | — |
| Heartbeat | $6.53\times10^{-7}$ | $2.70\times10^{-7}$ | — | — | — | — |
| Motor | $2.29\times10^{-6}$ | $5.36\times10^{-3}$ | — | — | — | — |

Table 11: Optimized reservoir and readout hyperparameters for MARS-TC, which prepends a fixed random temporal-convolutional encoder to the memristive stack. Columns match those of Table 9. Because the TC layer compresses the input sequence temporally, the search generally selects shallower networks ($N \leq 5$) and, on four of the six datasets, wider reservoirs than in the no-TC configuration.

| Dataset | $N_h$ | $N$ | $\alpha$ | $\omega$ | $\beta$ | enc$_W$ | enc$_b$ | sparsity | $s$ |
|---|---|---|---|---|---|---|---|---|---|
| EigenWorms | 577 | 2 | $4.87\times10^{-3}$ | 0.206 | 0.059 | 0.093 | 0.021 | 0.505 | 11 |
| SCP1 | 600 | 2 | $6.90\times10^{-2}$ | 0.022 | 0.066 | 0.018 | 0.425 | 0.734 | 11 |
| SCP2 | 383 | 4 | 0.503 | 1.262 | 0.044 | 0.169 | 0.670 | 0.904 | 7 |
| Ethanol | 442 | 5 | 5.793 | 0.330 | 1.093 | 0.353 | 1.849 | 0.250 | 4 |
| Heartbeat | 485 | 2 | 1.179 | 0.555 | 0.660 | 1.182 | 1.122 | 0.130 | 4 |
| Motor | 362 | 4 | 1.483 | 0.474 | 0.205 | 0.413 | 0.193 | 0.291 | 10 |

Table 12: Per-layer time step $\Delta_i$ for MARS-TC, as selected by Optuna. Notation follows Table 10. A dash (—) indicates that the dataset's optimal architecture uses fewer than $i$ layers (cf. column $N$ of Table 11)

| Dataset | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ | $\Delta_4$ | $\Delta_5$ |
|---|---|---|---|---|---|
| EigenWorms | $1.64\times10^{-5}$ | $4.95\times10^{-7}$ | — | — | — |
| SCP1 | $7.83\times10^{-4}$ | $1.09\times10^{-4}$ | — | — | — |
| SCP2 | $3.12\times10^{-5}$ | $4.60\times10^{-5}$ | $4.28\times10^{-7}$ | $1.68\times10^{-7}$ | — |
| Ethanol | $2.82\times10^{-7}$ | $3.55\times10^{-6}$ | $1.68\times10^{-3}$ | $3.95\times10^{-6}$ | $1.55\times10^{-2}$ |
| Heartbeat | $9.54\times10^{-5}$ | $4.14\times10^{-4}$ | — | — | — |
| Motor | $8.74\times10^{-3}$ | $6.19\times10^{-3}$ | $5.48\times10^{-7}$ | $2.02\times10^{-3}$ | — |

Table 13: Optimized temporal-convolutional encoder hyperparameters for MARS-TC. The TC layer is a single 1-D convolution with randomly initialized, non-trainable weights applied to the raw input sequence before the linear encoder and the memristive stack. *Channels* is the number of output feature maps of the convolution, *Kernel* is its temporal receptive field in input time steps, and *Stride* is the temporal downsampling factor (a stride of $k$ divides the effective sequence length by $k$). The TC layer is the only per-dataset architectural choice that lets MARS-TC adapt to sequence lengths as different as 405 (Heartbeat) and 17,984 (EigenWorms).

| Dataset | Channels | Kernel | Stride |
|---|---|---|---|
| EigenWorms | 28 | 9 | 2 |
| SCP1 | 13 | 9 | 3 |
| SCP2 | 14 | 3 | 4 |
| Ethanol | 42 | 5 | 1 |
| Heartbeat | 63 | 5 | 1 |
| Motor | 35 | 7 | 1 |