Supplementary Material to "Evolving Memristive Reservoir"

Xinming Shi, Leandro L. Minku *IEEE Senior Member*, and Xin Yao *IEEE Fellow*

***Abstract*—This supplementary material contains the pseduo codes of crossover and mutation operations, the evolved memristive reservoir circuits and their corresponding configuration, the parameter setting of the memristor model, the preliminary experiments to our proposed preprocessing steps, and the parameter setting of SOTA models. We also repeat some necessary contents here for easy reference and understanding.**

I. Pseudo code of crossover operation

Algorithm 1 displays the pseudocode of the crossover operation. As for the crossover of the reservoir's genome, the parent could be regarded as the product of W_{res} and W_{bool} of one individual. For the crossover operation of parents with different sizes, the size of the offspring's reservoir should be determined first. There are two alternatives for determining the size of the offspring's reservoir, which are to follow the parent with a larger size and to choose the size of the parent with better fitness, respectively. The crossover probability is P_c, which is to decide whether to take the crossover operation. Besides the determination of offspring's size, the weights value of the offspring's reservoir is determined by the following rule [1]:

$$w_{ij} = \begin{cases} \frac{w_{ij}^1 + w_{ij}^2}{2} & \text{if } w_{ij}^1, w_{ij}^2 \neq 0 \text{ and } random < 0.5, \\ w_{ij}^1 & \text{if } w_{ij}^1, w_{ij}^2 \neq 0 \text{ and } 0.5 \leq random < 0.75, \\ w_{ij}^2 & \text{if } w_{ij}^1, w_{ij}^2 \neq 0 \text{ and } 0.75 \leq random < 1.0, \\ w_{ij}^1 & \text{if } w_{ij}^2, = 0 \text{ and } w_{ij}^1 \neq 0, \\ w_{ij}^2 & \text{if } w_{ij}^1, = 0 \text{ and } w_{ij}^2 \neq 0. \end{cases} \tag{1}$$

Algorithm 1 Pseudo code of crossover `crossover()`

Input: $genomes$ in the current generation, P_c
Output: $genomes$ after crossover operations

$parent_1, parent_2$ selection by tournament strategy
if $random() < P_c$ **then**
  if $parent_1.size()! = parent_2.size()$ **then**
    if $parent_1.size() > parent_2.size()$ **then**
      $temp\_off \leftarrow parent_1$
    else
      $temp\_off \leftarrow parent_2$
    end if
  else
    if $parent_1.fitness > parent_2.fitness$ **then**
      $temp\_off \leftarrow parent_1$
    else
      $temp\_off \leftarrow parent_2$
    end if
  end if
  for each element w_{ij} in $temp\_off.size()$ **do**
    if w_{ij} is in overlapped area **then**
      Set w_{ij} based on Eq. (1), where w_{ij}^1 and w_{ij}^2 are the corresponding elements from $parent_1$ and $parent_2$, respectively
    else
      Discard w_{ij}
    end if
  end for
  Return $temp\_off$
end if
Return $parent_1$

II. Pseudo code of mutation operation

In order to encourage diversity of reservoirs during the evolution, five types of mutation operators are applied to the evolution.

- **Weight mutation**: For the values in W_{res} corresponding to the position where W_{bool} is not zero, there will be the probability P_m to mutate them to a new value taken uniformly at random within the allowable range. The pseudo code of weight mutation is given in Algorithm 2.
- **Add node:** To add a node to the reservoir and initialize its corresponding W_{bool} matrix to 0. The pseudo code of add node is given in Algorithm 3
- **Delete node:** To calculate the weight sum of W_{res} associated with each node, and delete the node with the smallest weight sum.
- **Jump mutation:** The jump step of CRJ structure will be mutated. The value range of the step is between 2 and N/2. According to the new jump step, W_{bool} will be

X. Shi and X. Yao are with Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, China, and School of Computer Science, University of Birmingham, UK. (e-mail: xxs972@cs.bham.ac.uk and xiny@sustech.edu.cn)

L. L. Minku is with School of Computer Science, University of Birmingham, UK. (e-mail:l.l.minku@bham.ac.uk)

updated, so that the CRJ structure could be rebuilt.

- **Input/GND node mutation:** The position of reservoir nodes connected to the input signal and GND will be mutated to increase circuit diversity picking a new position uniformly at random, since different terminals of the circuit connected to Input or GND could be a different circuit.

Algorithm 2 Pseudo code of weight mutation `mutate_weight()`

Input: $genomes$ that will be taken the `mutate_weight()`, P_{mem1}

Output: $genomes$ after `mutate_weight()`

```
for i in N do
    for j in N do
        if random() < P_mem1 then
            W_mem[i][j] ← random.uniform(0, 1)
        end if
    end for
end for
Return genomes with mutated W_res
```

Algorithm 3 Pseudo code of add node `mutate_add()`

Input: $genomes$ that will be taken the `mutate_add()`

Output: $genomes$ after `mutate_add ()`

```
Initializing temp_row_mem in [0, 1] with size 1 × N. Initializing temp_con_mem in [0, 1] with size (N+1) × 1
Stacking W_res with temp_row_mem
Stacking W_res with temp_con_mem;
Initializing temp_bool with 0
Stacking W_bool with temp_bool;
N+=1
Return genomes with added node
```

Algorithm 4 Pseudo code of delete node `mutate_delete()`

Input: $genomes$ that will be taken the `mutate_delete()`

Output: $genomes$ after `mutate_delete ()`

```
index ← a random index within N
Deleting one row with index of W_bool
Deleting one column with index of W_bool
Deleting one row with the same index of W_res
Deleting one column with the same index of W_res
N-=1
Return genomes with deleted node
```

III. The evolved memristive reservoir circuits and their corresponding configuration

This presents the evolved memristive reservoir circuits and their corresponding configuration, which is shown in Figure. 1.

Algorithm 5 Pseudo code of jump mutation `mutate_jump()`

Input: $genomes$ that will be taken the `mutate_jump()`

Output: $genomes$ after `mutate_jump ()`

```
step ← a random step between (2, N/2)
for i in (0, N − 1) do
    W_bool[i][i + 1] = 1
    start=0
    while (start + step) ≤ (N − 1) do
        W_bool[start][start + step] = 1
        W_bool[start + step][start] = 1
        start+=step
    end while
end for
Return genomes with jump mutation
```

Algorithm 6 Pseudo code of input or GND mutation `mutate_In/GND()`

```
index ← a random index of the terminal list IN/GND.
a ← a random number between (1, N)
while a in the terminal list of input or GND do
    a ← a random number between (1, N)
end while
The terminal in IN/GND with index ← a
Return genomes with mutated the terminals of connecting Input and GND.
```

Fig. 2 shows the superposition between the actual outputs of our proposed memristive reservoir vs corresponding targets. They show that the signal generated by our proposed memristive reservoir circuit is mimicking the desired signal well.

IV. Memristor model characteristics and parameter setting

This section provides the memristor model characteristics and corresponding parameter settings. Table I introduces the parameter meaning and setting of the applied memristor model [2]. They were adapted from the parameters of the bipolar model of the memristor model in [2] by being tuned further based on preliminary circuit simulation to ensure the memristors could generate the fading dynamics within our simulated time window (0.5s) one by one.

Regarding preliminary circuit simulation, it includes the following steps:

- Step 1: The range of the input signal has been limited to $(2.5V, 5V)$. Therefore, we first apply the pulse with the minimum voltage $2.5V$ as input to the embryo circuit.
- Step 2: Check if the fading state of the memristor current can be observed in the given time window, which is 0.5s.
- Step 3: For each parameter related to the fading effect (τ, σ, δ, θ), repeat steps 3.1 and 3.2:

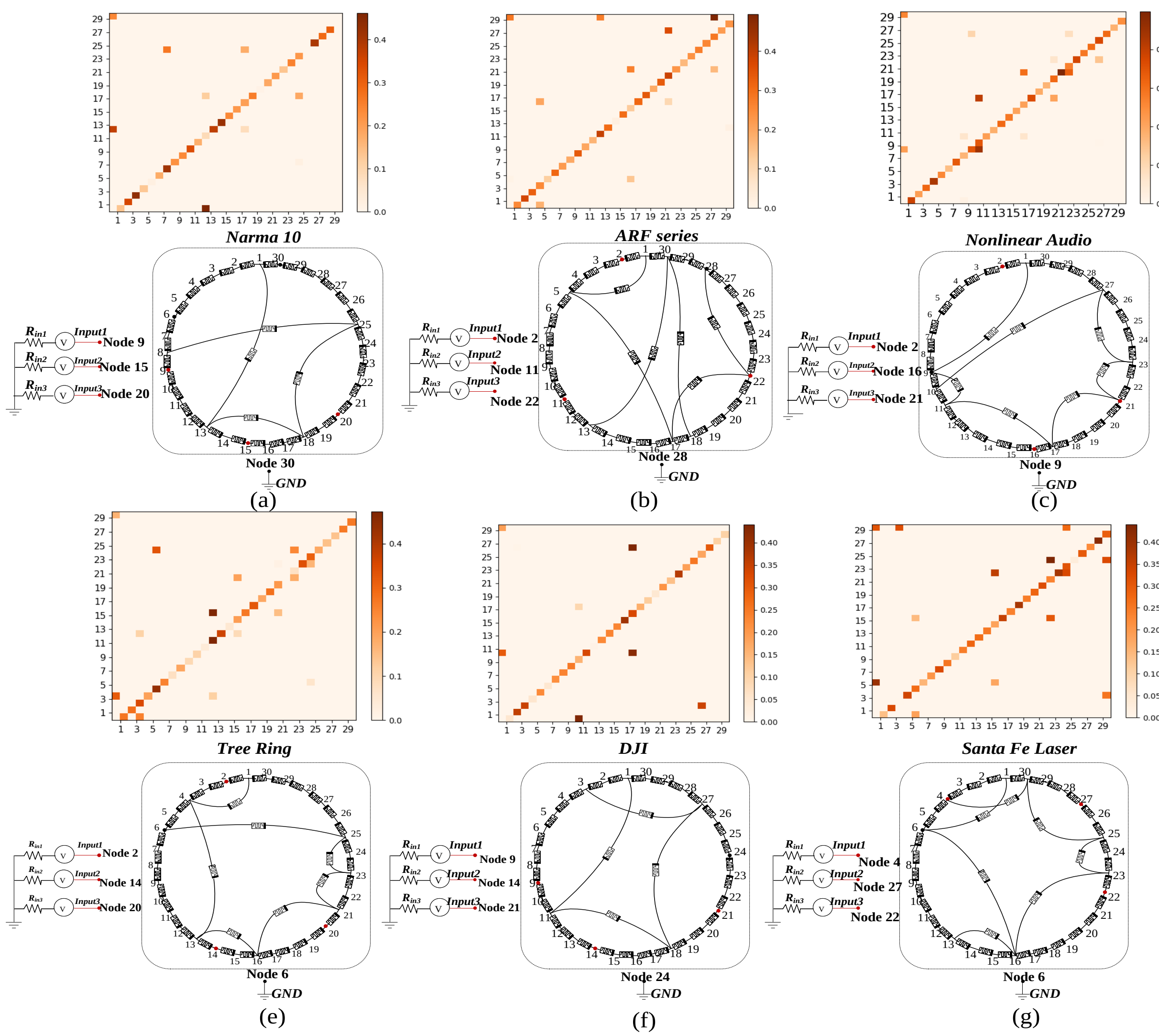

Fig. 1. Visualization of memristive reservoir topology and equivalent circuits. (a) Narma-10; (b) Nonlinear audio; (c) ARFIMA series; (d) Tree ring; (e) DJI (f) Santa Fe laser.

- Step 3.1: If no fading effect is observed, this means that the parameter is set too large, requiring more simulation time than 0.5s. Therefore, the parameter is tuned by decreasing it by 1% of its original value.
- Step 3.2: If the fading effect is observed sharply, this means that the parameter is set too large, requiring more simulation time than 0.5s. Therefore, the parameter is tuned by increasing it by 1% of its original value.
- Step 4: Based on these tuned parameters, we then apply the pulse with maximum voltage $5V$ as input to the embyro circuit. Steps 2 and 3 are repeated to further tune these parameters based on the $5V$ voltage.
- Step 5: $\alpha, \beta, \gamma, \lambda$ are related to the rate of state switching. Then, similar steps as Step 3 and 4 will be applied to tune these parameters.

The memristor model we have applied in this work has the forgetting effect, which will be applied to implement the short-term memory effect of the reservoir in this work. Its mathematical model is shown as follows:

$$i = (1-x)\alpha[1-e^{-\beta v}] + x\gamma sinh(\delta v), \tag{2}$$

$$\dot{x} = (\lambda[e^{\eta_1 v} - e^{\eta_2 v}] - \frac{x-\theta}{\tau})f(x), \tag{3}$$

$$\dot{\varepsilon} = \sigma(e^{\eta_1 v} - e^{\eta_2 v})f(x), \tag{4}$$

$$\dot{\tau} = \theta(e^{\eta_1 v} - e^{\eta_2 v}), \tag{5}$$

$$f(x) = \frac{(sign(v)+1)(sign(1-x)+1)+(sign(-v)+1)(sign(x)+1)}{4}, \tag{6}$$

Figure. 3 shows the circuit simulation result of the memristor state variable x under the pulse stimulus, where Figure. 3 (b) is our applied memristor. During the phase 2 and 4, when there is no pulse stimulus, the memristor state variable will fade toward the initial state.

This memristor model can also describe four different types of characteristics by setting different parameters, which are

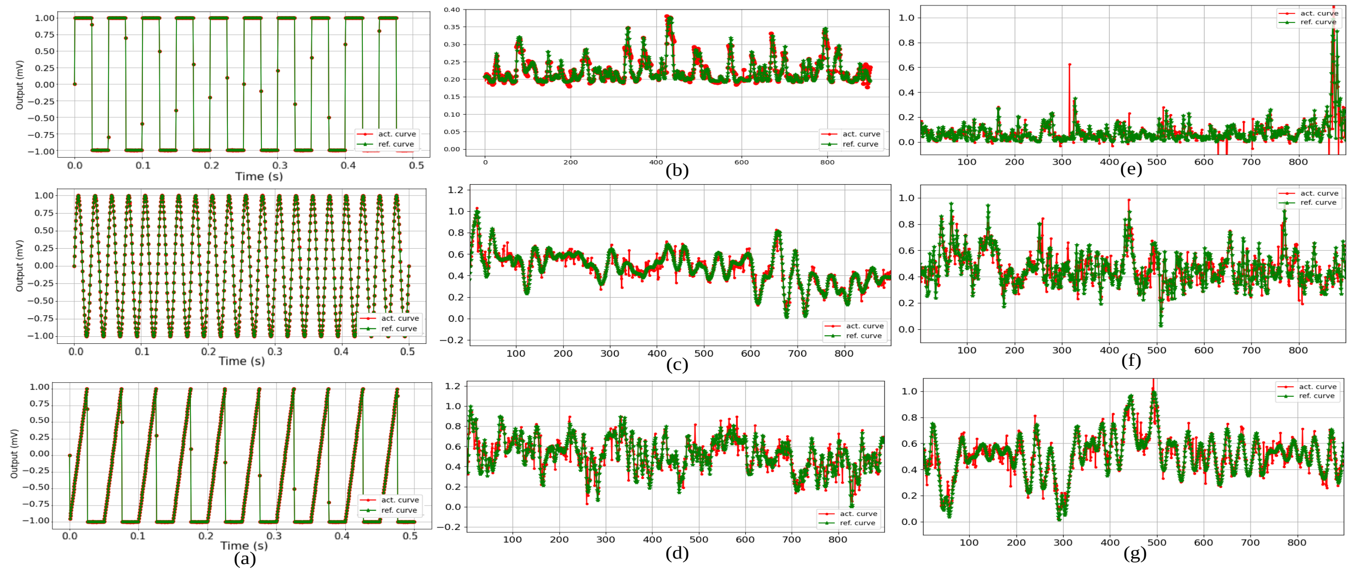

Fig. 2. Actual outputs of our proposed memristive reservoir vs corresponding targets. (a) Wave generation task; (b) Narma-10; (c) Nonlinear audio; (d) ARFIMA series; (e) Tree ring; (f) DJI (g) Santa Fe laser.

TABLE I
THE PARAMETERS OF APPLIED MEMRISTOR MODEL

Model parameters [2]	Meaning	Value
α	Prefactor corresponding to barrier height for Schottky barrier	1e-4
β	Exponent corresponding to depletion width for Schottky barrier	0.2
γ	Prefactor corresponding to barrier height for tunneling	1e-3
δ	Exponent corresponding to effective tunneling distance in the conducting region	1
ε	Retention of the Ohmic-like conducting channel	0.1
η_1	Interface effect with positive voltage	4
η_2	Interface effect with negative voltage	2
λ	Positive constant to control the change rate of x	0.005
τ	Diffusion time	0.5
θ	Positive-valued coefficient for τ	0.01
σ	Positive-valued coefficient for ε	0.0001

the bipolar, the unipolar, the bipolar with forgetting effect, the reversible bipolar and unipolar. The detailed definition of them are the following:

- The bipolar: The memristance increases and decreases by different polar voltages.
- The unipolar: The memristance can increase and decrease by the same polar voltage.
- The bipolar with forgetting effect: The memristance increases and decreases by different polarity of voltage, but the memristance will spontaneously decay at the mean time, even with no voltage.
- The reversible bipolar and unipolar: Memristor will behave as the bipolar memristor first, but after some iterations, it will turn to be a unipolar memristor.

The parameters setting for these four different types of the memristor are listed in the following Table II.

The window functions $f(x)$ used in different types of memrsitor are listed as follow:

TABLE II
THE PARAMETER SETTING AND CORRESPONDING DIFFERENT CHARACTERISTICS

Parameters	Bipolar	Bipolar with forgetting effect	Unipolar	Reversible
α	0.5e-5	0.5e-7	0.5e-4	0.5e-4
β	0.5	0.01	0.01	0.01
γ	25e-5	2e-7	3e-4	3e-4
δ	1	4	1	1
λ	1	0.13	0.5	0.05
η_1	1	4	3	1
η_2	2	2	3	1
θ	0.04	0.04	0.0001	0.03
σ	0.03	0.03	0.03	0.0001
ε	0.1	0.1	0.01	0.001
τ	10,000	0.15	0.05	0.5
Window function	Function (7)	Function (9)	Function (8)	Function (9)

$$f(x) = 1 - (2x - 1)^{2p} \tag{7}$$

$$f(x) = 1 - (x - stp(-i))^{2p} \quad stp(i) = \begin{cases} 1 & i \geq 0 \\ 0 & i < 0 \end{cases} \tag{8}$$

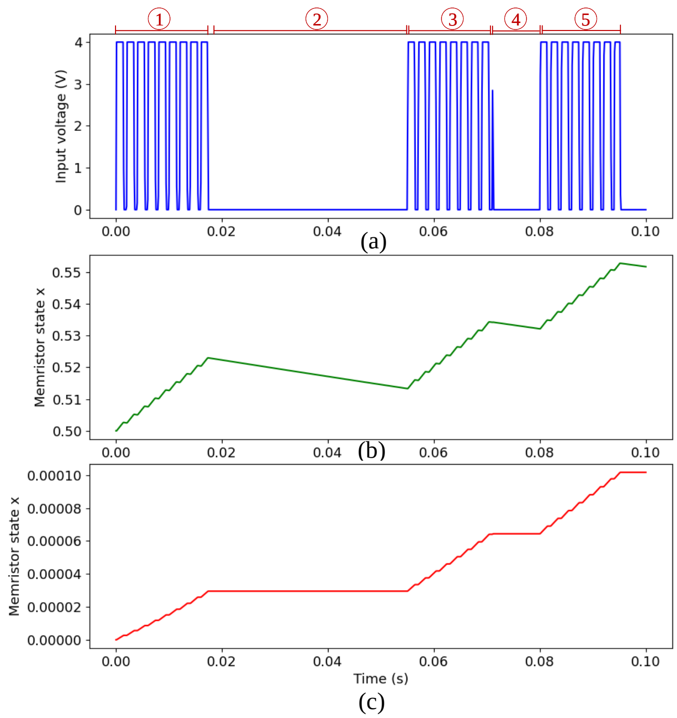

Fig. 3. The circuit simulation result of the memristor state variable x under the pulse stimulus. (a) Input pulse; (b) Forgetting memristor model state variable x; (c) HP memristor model state variable x.

$$f(x) = \frac{(sign(v)+1)(sign(1-x)+1)+(sign(-v)+1)(sign(x)+1)}{4} \tag{9}$$

V. Preliminary experiments to our proposed preprocessing steps

This section presents preliminary experiments to our proposed prepossessing steps, where the result comparisons of removing and applying our proposed preprocessing steps are listed in Table III. According to the results shown in Table III, our proposed preprocessing steps are required and play an important role to the reservoir performance.

When this preprocessing step is removed, a performance comparison table is listed in Table III. As we can see, if there is no preprocessing step that transforms from one signal of one dimension into multi-signal, the performance of the reservoir will degrade, which indicates that this preprocessing step plays an important role in the reservoir regarding as the input masking operation.

VI. The parameter setting of SOTA models

In order to make a fair comparison with SOTA models, we have applied the different optimization methods to the parameter selection of SOTA models, the specific parameter settings of the SOTA models are given in Table IV and Table V.

The global parameters of the baseline models will be optimized, such as the input scaling factor IS, reservoir scaling factor RS, leaky rate LR, spectral radius ρ, connectivity C for ESN with random topology, D and β for ESN with small-world topology, where C denotes the connection probability of a random reservoir, D denotes the number of the closest neighboring nodes, β denotes the probability of adding new connection to each connection. As for the vanilla RNN, there are also several parameters that will be optimized, including the sparseness of the connection p, the spectral scaling factor g, and the leaky rate LR. As for the vanilla LSTM, win_size will be optimized. Regarding the improved variants of RNN and LSTM, mRNN and mLSTM, there is one more memory parameter, k, which is needed to be optimized. Table IV gives the parameter setting of the ESN with random topology and ESN with small-world topology, and the corresponding comparison with our proposed method. Table V gives the parameter setting of vanilla RNN, vanilla LSTM, mRNN and mLSTM, and their corresponding comparison with our proposed method.

References

[1] K. C. Chatzidimitriou and P. A. Mitkas, “Adaptive reservoir computing through evolution and learning,” *Neurocomputing*, vol. 103, pp. 198–209, 2013.

[2] L. Chen, C. Li, T. Huang, X. Hu, and Y. Chen, “The bipolar and unipolar reversible behavior on the forgetting memristor model,” *Neurocomputing*, vol. 171, pp. 1637–1643, 2016.

TABLE III
PERFORMANCE (RMSE) COMPARISONS WITH AND WITHOUT PREPROCESSING

Our proposed memrsitve reservoir	Wave	Narma-10	DJI	Audio	Tree ring	ARF	Santa
Without preprocessing	0.1283	0.0925	0.1765	0.1298	0.0933	0.1981	0.1546
With preprocessing	0.0099	0.0239	0.0657	0.0493	0.0641	0.0743	0.0582

TABLE IV
COMPARISONS BETWEEN OUR PROPOSED METHOD AND OTHER OPTIMIZATION METHOD TO BASIC ESN

Optimization methods		Description	Task	Topology		RMSE with fixed evaluation number (4000)		Circuit feasibility & scalability
				ESN-random	ESN-small world	ESN-random	ESN-small world	
Software Reservoir Optimization	Grid search	Optimizing global parameters	Narma10	IS=0.0563, RS=0.5838, LR=0.4133, ρ=0.8039 C=0.6532	IS=0.7930, RS=0.6759, LR=0.3912, ρ=0.2495 D=57,β=0.3425	0.0778	0.0723	No
			Audio	IS=0.8498, RS=0.5283, LR=0.5247, ρ=0.3635 C=0.7365	IS=0.5380, RS=0.5453, LR=0.5849, ρ=0.4113 D=77,β=0.4535	0.1321	0.1321	
			DJI	IS=0.6382, RS=0.2706, LR=0.6522, ρ=0.4954 C=0.7242	IS=0.6405, RS=0.4043, LR=0.6197, ρ=0.1685 D=55,β=0.4323	0.1357	0.1353	
			DGR	IS=0.9543, RS=0.6312, LR=0.8320, ρ=0.7643 C=0.6452	IS=0.8974, RS=0.5439, LR=0.5898, ρ=0.2334 D=65,β=0.6115	0.8213	0.8300	
			ARF	IS=0.7925, RS=0.3953, LR=0.5048, ρ=0.7036 C=0.4406	IS=0.7538, RS=0.4588, LR=0.5165, ρ=0.7381 D=76,β=0.4234	1.5527	1.5120	
	Differential evolution	Optimizing global parameters	Narma10	IS=0.5050, RS=0.4058, LR=0.7927, ρ=0.9332 C=0.6873	IS=0.6239, RS=0.2648, LR=0.4944, ρ=0.53145 D=79,β=0.3950	0.0415	0.0413	No
			Audio	IS=0.8180, RS=0.5286, LR=0.5251, ρ=0.9997 C=0.1947	IS=0.5204, RS=0.5452, LR=0.5868, ρ=0.9996 D=46,β=0.5868	0.0725	0.0725	
			DJI	IS=0.8180, RS=0.5286, LR=0.5251, ρ=0.9997 C=0.1947	IS=0.7021, RS=0.1572, LR=0.3977, ρ=0.2834 D=53,β=0.4321	0.1234	0.1219	
			DGR	IS=0.8011 RS=0.3915, LR=0.9434, ρ=0.6732 C=0.8964	IS=0.7891, RS=0.5645, LR=0.7881, ρ=0.5433 D=71,β=0.8362	0.8420	0.8433	
			ARF	IS=0.9231, RS=0.4896, LR=0.4832, ρ=0.7856 C=0.5535	IS=0.3827, RS=0.5294, LR=0.5209, ρ=0.7290 D=58,β=0.8171	1.3950	1.3978	
Physical Reservoir Optimization	Manual optimization	Optimizing the physical reservoir by experts	The parameter selection and performance are highly relied on the experts' experience.					Yes
	Ours	Optimizing configuration signals	Narma10	Adaptive sparse topology based on the reconfigurable architecture		0.0239		Yes
			Audio			0.0493		
			DJI			0.0657		
			DGR			0.9892		
			ARF			0.0743		

TABLE V
COMPARISONS BETWEEN OUR PROPOSED METHOD AND OTHER OPTIMIZATION METHODS TO VANILLA RNN AND LSTM, MRNN AND MLSTM

Optimization methods		Description	Task	Methods		RMSE with fixed evaluation number (4000)		Circuit feasibility & scalability
				vanilla RNN	vanilla LSTM	vanilla RNN	vanilla LSTM	
Software Reservoir Optimization	Grid search	Optimizing global parameters	Narma10	p=0.8932, g=0.7124, LR=0.5892	win_size=14	0.0448	0.0415	No
			Audio	p=0.6291, g=0.8968, LR=0.9237	win_size=26	0.0277	0.0393	
			DJI	p=0.9347, g=0.7752, LR=0.8454	win_size=35	0.2605	0.2492	
			DGR	p=0.8966, g=0.5698, LR=0.7329	win_size=52	0.9494	0.8566	
			ARF	p=0.8123, g=0.9653, LR=0.5736	win_size=24	1.1620	1.1340	
	Differential evolution	Optimizing global parameters	Narma10	p=0.8212, g=0.7264, LR=0.3426	win_size=17	0.0325	0.0401	No
			Audio	p=0.6823, g=0.8234, LR=0.9246	win_size=33	0.0241	0.0373	
			DJI	p=0.6721, g=0.7642, LR=0.6764	win_size=24	0.2256	0.2033	
			DGR	p=0.7709, g=0.8521, LR=0.9813	win_size=35	0.9567	0.9066	
			ARF	p=0.3486, g=0.67544, LR=0.8867	win_size=32	1.0781	1.1270	
Optimization methods		Description	Task	Methods		RMSE with fixed evaluation number (4000)		Circuit feasibility & scalability
				mRNN	mLSTM	mRNN	mLSTM	
Software Reservoir Optimization	Grid search	Optimizing global parameters	Narma10	p=0.5632, g=0.6341, LR=0.8722, k=11	win_size=22, k=12	0.0219	0.0506	No
			Audio	p=0.9353, g=0.8764, LR=0.4655, k=25	win_size=27, k=33	0.0543	0.0231	
			DJI	p=0.9563, g=0.9443, LR=0.6901,k=77	win_size=36, k=25	0.2487	0.2531	
			DGR	p=0.8343, g=0.7862, LR=0.6841 k=55	win_size=75, k=65	0.9767	0.8466	
			ARF	p=0.7310, g=0.9891, LR=0.8702, k=58	win_size=45, k=47	1.0880	1.1490	
	Differential evolution	Optimizing global parameters	Narma10	p=0.8211, g=0.7409, LR=0.5560, k=28	win_size=14, k=29	0.0273	0.0432	No
			Audio	p=08810, g=0.8891, LR=0.7011, k=32	win_size=0.4321, k=25	0.0456	0.0211	
			DJI	p=0.7899, g=0.6096, LR=0.6711k=50	win_size=28, k=45	0.2234	0.2146	
			DGR	p=0.8012, g=0.5652, LR=0.7884,k=63	win_size=76, k=55	0.9833	0.8800	
			ARF	p=0.5021, g=0.8930, LR=0.7121, k=45	win_size=35, k=46	1.0249	1.0375	
Physical Reservoir Optimization	Manual optimization	Optimizing the physical reservoir by experts	The parameter selection and performance are highly relied on the experts' experience.					Yes
	Ours	Optimizing configuration signals	Narma10	Adaptive sparse topology based on the reconfigurable architecture		0.0239		Yes
			Audio			0.0493		
			DJI			0.0657		
			DGR			0.9892		
			ARF			0.0743		