A Tuning-Free Hardware Reservoir Based on MOSFET Crossbar Array for Practical Echo State Network Implementation

Yuki Kume Song Bian Takashi Sato

Graduate School of Informatics
Kyoto University
Yoshida-hon-machi, Sakyo, Kyoto, 606-8501, Japan
Tel: 075-753-4801
Fax: 075-753-4802
e-mail: paper@easter.kuee.kyoto-u.ac.jp

Abstract— Echo state network (ESN) is a class of recurrent neural network, and is known for drastically reducing the training time by the use of *reservoir*, a random and fixed network as the input and middle layers. In this paper, we propose a hardware implementation of ESN that uses practical MOSFET-based reservoir. As opposed to existing reservoirs that require additional tuning of network weights for improved stability, our ESN requires no post-training parameter tuning. To this end, we apply the circular law of random matrix to sparse reservoirs to determine a stable and fixed feedback gain. Through the evaluations using Mackey-Glass time-series dataset, the proposed ESN performs successful inference without post parameter tuning.

I. Introduction

Recurrent neural network (RNN), which contains loops in the network, is a class of neural network model that is suitable for analyzing time series data. Conventional RNN models include backpropagation through time (BPTT) [1], real time recurrent learning (RTRL) [2], and long short-term memory (LSTM) [3]. Most of these models are known to require long learning time until all the weights are trained. In addition, the gradient may vanish or explode due to the loop topology, making it difficult for the above models to attain stable training result [4].

More recently, echo state network (ESN) [5] and liquid state machine (LSM) [6] have been independently proposed in different research domains. They are collectively referred to as reservoir computing (RC) because both have the component called *reservoir* (or liquid). The reservoir serves as the input and the middle layers of the network. Under several constraints explained later, connectivity and the weights in the reservoir can be determined randomly, and they are fixed thereafter. The training of the network is thus applied only for the output layer, greatly reducing the computational cost of training.

The performances of various RNNs, including recently proposed DeepESN [7], are compared in [8]. The DeepESN uses series of multi-time scale reservoirs to enhance the dynamics of the network. In [8], ESN and DeepESN presented equal or higher accuracy than LSTM and gated recurrent units (GRU) [9]. Meanwhile, the learning time of ESN and DeepESN is considerably shorter than that of other models. ESN and DeepESN are reported 18x and 22x faster, respectively, than LSTM and GRU on some particular dataset.

Another advantage of ESN is the ease of hardware implementation. Since the structure and the weights of the reservoir are unchanged throughout the training as well as the inference phases, no training circuit is required, making ESN extremely area efficient. Moreover, hardware variations can be effectively used as the source of the random weights. Compared to software reservoir implementations, hardware reservoir is expected to achieve lower power consumption and faster operation in inference.

Due to its architectural simplicity and superior performance, hardware reservoir has become an active research topic [10]. For example, a hardware reservoir consisting of a memristor array is proposed in [11]. The advantage of using memristors is their programmability, and the weights are indeed written to the memristors in [11]. Here, the weights are calculated off-line, i.e., using a software. As a result, the memristor reservoir requires post tuning of the network parameters so that the constraints of the reservoirs are satisfied. In addition, due to the emerging nature of the memristor technology, precise programming of the analog values on memristers can sometimes be difficult [12] [13].

In this paper, we propose a tuning-free hardware reservoir consisting of a MOSFET crossbar array. The main contributions of this work are summarized as follows.

a) MOSFET-based Reservoir: The use of MOSFETs as the hardware reservoir achieves more stable and practical hardware implementation than the emerging devices. Unavoidable threshold-voltage variations in MOSFETs are used to determine the random weights. To the best of our knowledge, this work represents the first design exploration of hardware reservoir using intrinsic variations of MOSFET arrays.

b) Bounding the Spectral Radius: The connectivity and weights in the reservoir are represented by the elements in a weight matrix that are randomly determined by intrinsic MOSFET characteristics. However, as later explained in Section II.II-A, it becomes hard to control the spectral radius for MOSFET-based weight matrix, as the randomness are built-in to each MOSFET. In this work, by applying the circular law on sparse matrix, we derive a theoretical approach to calculate the spectral radius of the weight matrix in any MOSFET-based reservoir.

c) Tuning the Weights: The weights should be scaled following the calculated spectral radius. On software platforms, we can freely scale the weights. Unfortunately, on MOSFET-based reservoir, we cannot change the weights (or MOSFET characteristics). In this work, we propose a resistor-based technique in the crossbar array to scale the weight arbitrarily. The value of the resistance can be calculated before fabricating the hardware reservoir.

This paper is organized as follows. Section II reviews the constraints placed on the reservoir to construct an ESN. The existing work of reservoir design comprising of a memristor crossbar array is also reviewed. In Section III, we propose a new reservoir consisting of MOSFET crossbar array that autonomously satisfies the necessary constraints. The evaluation of the ESN that uses the proposed MOSFET reservoir will be shown in Section IV. Finally, Section V concludes this paper.

II. Echo State Network

Throughout this work, we use x, y for scalar variables, $\boldsymbol{x}, \boldsymbol{y}$ for vectors, and $\boldsymbol{X}, \boldsymbol{Y}$ for matrices.

A. Theory

Fig. 1 shows the general structure of ESN that consists of three layers: input, middle and output. The input and middle layers are collectively called *reservoir* and the output layer is called *readout*. Each layer has nodes represented by vectors $\boldsymbol{u}$, $\boldsymbol{x}$ and $\boldsymbol{y}$. Nodes are connected by edges with weights. $\boldsymbol{x}$ and $\boldsymbol{y}$ at time t are updated by

$$\boldsymbol{x}(t) = \tanh\left(\boldsymbol{W}_{\text{in}}\boldsymbol{u}(t) + \boldsymbol{W}\boldsymbol{x}(t-1)\right) \tag{1}$$

$$\boldsymbol{y}(t) = \boldsymbol{W}_{\text{out}}\boldsymbol{x}(t), \tag{2}$$

where $\boldsymbol{W}_{\text{in}}$, $\boldsymbol{W}$, and $\boldsymbol{W}_{\text{out}}$ are the weight matrices for the input, middle, output layers, respectively. According to the input $\boldsymbol{x}$, tracking or predicting time-series data is achieved at the output $\boldsymbol{y}$. Here, during the training, only $\boldsymbol{W}_{\text{out}}$ is trained, whereas $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}$ are fixed. Though both $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}$ can be randomly determined, there exist the following three constraints, referred to as *weight constraints*, on the matrix elements.

a) Connectivity: The matrix $\boldsymbol{W}_{\text{in}}$ is typically dense, whereas the matrix $\boldsymbol{W}$ should be sparse in order for each node to show modest interaction with other nodes. Connectivity measures how dense a matrix is, and is basically the ratio between the number of nonzero elements and the total number of matrix elements. Concretely, $1.25,\ 5,\ 15,\ 20\%$ are suggested in [5] [14].

b) Mean: The mean of the elements in $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}$ should be 0 to suppress increased offset in the input layer calculation and during the repetitive multiplication of $\boldsymbol{W}$ to the internal node vectors.

c) Spectral radius: Spectral radius, i.e., the largest absolute value of the eigenvalues of $\boldsymbol{W}$, should be bounded close to 1.0 [15] to avoid divergence or diminishment of $\boldsymbol{x}$ during the repetitive multiplication of $\boldsymbol{W}$ and $\boldsymbol{x}$ in Eq. (1).

B. Memristor Reservoir

Hardware reservoir using memristor crossbar array is proposed in [11]. Crossbar array has a form that can efficiently calculate the matrix-vector product, such as $\boldsymbol{W}_{\text{in}}\boldsymbol{u}(t)$ and

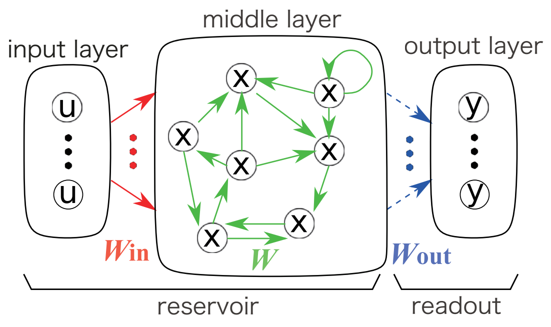


Fig. 1. General structure of ESN. The circles represent nodes and the arrows represent connections with weights. The input layer nodes are fully connected to the middle layer nodes. The middle layer nodes are randomly connected to other nodes or self in the middle layer and are fully connected to the output layer nodes.

$\boldsymbol{W}\boldsymbol{x}(t-1)$, in Eq. (1). Fig. 2 shows a reservoir architecture consisting of the memristor crossbar array [11]. In this circuit, the voltages $\boldsymbol{V}_u$ and $\boldsymbol{V}_x$ correspond to $\boldsymbol{u}$ and $\boldsymbol{x}$, respectively. $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}$ are respectively represented by the conductance of memristors, $\boldsymbol{G}_{in}$ and $\boldsymbol{G}$. Intersections with no memristor expresses no connection (zero weight). Here, the size of $\boldsymbol{W}_{\text{in}}$ is $1\times N$ and that of $\boldsymbol{W}$ is $N\times N$. These weights are calculated separately in a software, and programmed to the memristors before the reservoir starts performing actual computations.

In order to express both positive and negative weights by using positive conductance only, a pair of arrays, each of which represents positive and negative values, is used. The difference of these currents are calculated at the output of the middle layer using summation amplifier (SA) shown in Fig. 3(a). With an equal resistance value for R_0 and R_1, the output voltage V_o is given by $R_2(I^+ - I^-)$. Here, I^+ and I^- are the column currents of the positive and negative arrays, respectively. As shown in Fig. 3(b), the output range of V_o is limited by the supply voltages of the operational amplifiers. The transfer characteristic of the SA, f_{SA}, approximates $\tanh$. Hence, V_o follows:

$$V_\text{o} = f_{\text{SA}}\left[R_2(I^+ - I^-)\right] = f_{\text{SA}}\left[R_2(G^+ - G^-)V\right], \tag{3}$$

where the conductance difference $G^+ - G^-$ represents a negative value when $G^+ < G^-$.

In the memristor reservoir, Eq. (1) corresponds to the following equation

$$\boldsymbol{V}_x(t) = f_{\text{SA}}\left[R_2\boldsymbol{G}_{\text{in}}\boldsymbol{V}_u(t) + R_2\boldsymbol{G}\boldsymbol{V}_x(t-1)\right]. \tag{4}$$

The weights in the memristor reservoir can be programmed so that the weight constraints are satisfied. Therefore, post programming is required, where memristors are written with the designated values determined off-line.

III. MOSFET based reservoir

In this section, we propose a novel crossbar array reservoir that uses MOSFET current to express edge weights. Fig. 4 shows the overview of the ESN (MOS-ESN) that uses proposed MOSFET crossbar array. The proposed MOSFET crossbar array acts as the hardware reservoir in between the input signals and the software readout. In contrast to the memristor-based reservoir, the proposed MOSFET-based reservoir requires neither the separate calculation of the edge

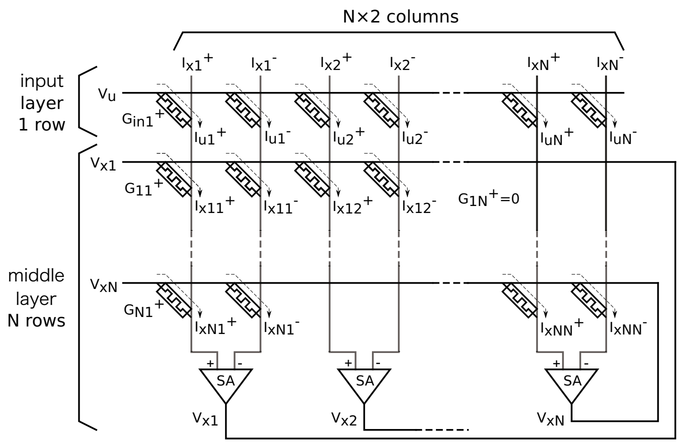


Fig. 2. Hardware reservoir using memristor crossbar array. Summation amplifier (SA) is shown in Fig. 3(a).

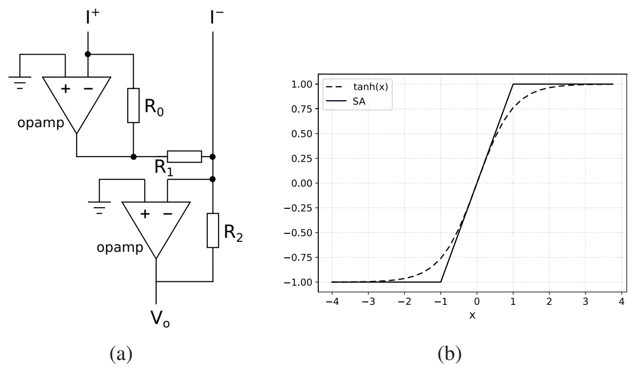


Fig. 3. Schematic and transfer function of summation amplifier (SA) [11]. (a) SA has two current inputs (I^+, I^-) and a voltage output (V_o). When R_0 and R_1 are set equal, SA outputs a voltage $V_\mathrm{o} = R_2(I^+ - I^-)$. (b) Transfer function of SA, which approximates $\tanh$ function.

weights nor the post programming of array elements. The key idea of the proposed reservoir is to utilize threshold-voltage variations in the MOSFETs to determine the random weights of W_{in} and W, and replaces memristors with MOSFETs in the array-structured reservoir. However, as-fabricated reservoirs using the MOSFET crossbar may not satisfy the weight constraints that are presented in the previous section. Hence, after quickly showing the structure of the proposed MOSFET reservoir in Section III.III-A, we will study the property of sparse random matrix to find an approximation formula for the estimation of spectral radius in Section III.III-B and Section III.III-C. Finally, utilizing the approximation formula, we will show how the proposed architecture of MOSFET based reservoir can satisfy the weight constraints without post tuning in Section III.III-D.

A. MOSFET Crossbar Array Reservoir

Fig. 5 shows the proposed MOSFET crossbar array reservoir. The basic structure is similar to the reservoir made of memristors. At the intersections of the crossbar, MOSFETs that are biased in the linear operation region are placed. Crossbar array is duplicated as was done in the memristor reservoir to represent negative conductances. In the MOSFET crossbar, control of the gate terminal is necessary as MOSFET is a three-terminal device. This is easily achieved by a 1-bit

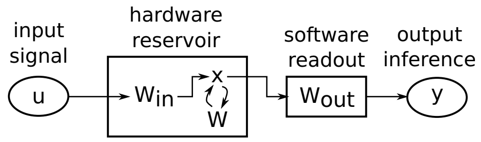


Fig. 4. Architecture of MOS-ESN that uses MOSFET reservoir.

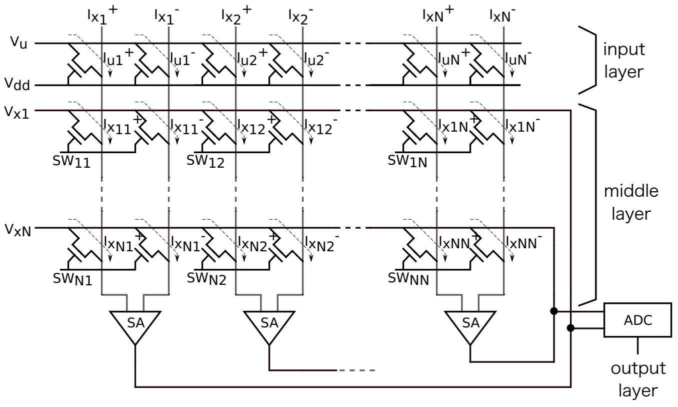


Fig. 5. Proposed hardware reservoir using MOSFET crossbar array. The structure is the same as that of memristor crossbar array except that the MOSFETs are used in place of memristors. ADC represents analog-to-digital converter for the interface to the output layer.

memory cell or a scan flip-flop that is connected to each of the MOSFET. The connections between the nodes in the reservoir can be determined through either writing connection patterns to the memory bits, or implemented by a layout design as mask patterns of connecting either VDD or VSS. Though all of the MOSFETs are biased under the same condition, conductances (weights) become random due to inherent device variations.

MOSFET based reservoir has several advantages over the memristor one. Below we summarize the major advantages to use MOSFETs to build a reservoir in ESN.

- Mature fabrication processes are available with low cost.
- MOSFET array can use compact periphery and power supply circuits as no memristor programming is necessary that requires high voltage and detailed timing control.
- Similarly to the memristor array, one instance of the reservoir can represent different node connections by altering the memory patterns that control the MOSFET states. The existence of the connection between nodes can be represented by turning on the corresponding MOSFET. There is no need to re-design mask layout nor re-fabrication of the chip.

While there exist a number of obvious advantages, MOSFET-based reservoir still has to satisfy the weight constraints in order to operate correctly in ESN. We investigate the characteristics of sparse random matrix generated by MOSFET crossbar arrays, and propose an automated weight tuning technique for MOSFET reservoirs.

B. Weights in MOSFET Reservoir

In general, the conductance of a MOSFET depends on the drain-source voltage. However, the conductance has to be fixed during the analysis. Hence, we firstly show that MOSFET conductance difference $G = G^{+} - G^{-}$ maintains a constant value regardless of the changes of drain-source voltage.

When threshold-voltage variations of MOSFETs are considered, the drain current $I_{\rm DS}$ is expressed as

$$I_{\rm DS} = {\rm A}\left[\left\{{\rm V_{GS}} - \left(\overline{\rm V_{th}} - \Delta_{\rm th}\right)\right\}{\rm V_{DS}} - \frac{1}{2}{\rm V_{DS}^2}\right], \quad (5)$$

where A is a constant gain factor, $V_{\rm GS}$ is gate-source voltage, $V_{\rm DS}$ is drain-source voltage, $\overline{V_{\rm th}}$ is a mean of threshold-voltage and $\Delta_{\rm th}$ is the deviation of the threshold-voltage. When a drain voltage is applied, conductance difference G is obtained as

$$G \;=\; (I_{\rm DS}^{+} - I_{\rm DS}^{-})/V_{\rm DS} = {\rm A}\left(\Delta_{\rm th}^{+} - \Delta_{\rm th}^{-}\right). \quad (6)$$

Here, G is constant and random as long as the variation of the gain factor is sufficiently small.

C. Weight Constrains

The weight matrix of the MOSFET array must also satisfy the weight constraints in Section II.II-A: *connectivity*, *mean* and *spectral radius*. The first two constraints are easy to satisfy.

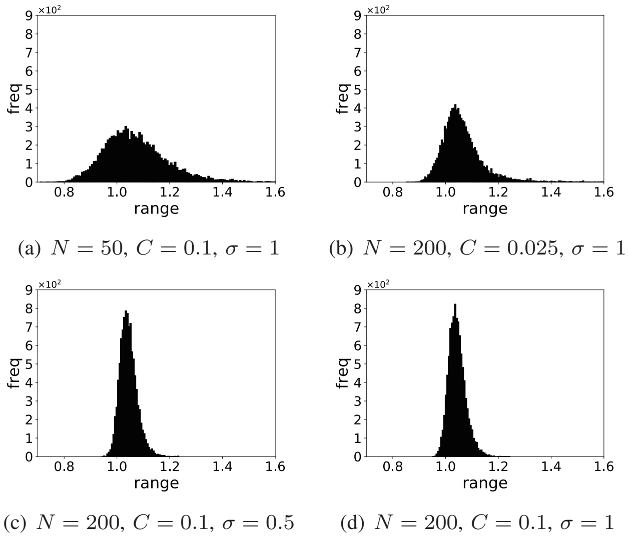
a) Connectivity constraint: The connectivity constraint requires a certain level of sparsity in the random weight matrix. This constraint can be satisfied as we can set arbitrarily connections by controlling the gate voltage of the MOSFETs.

b) Mean constraint: The mean constraint requires that the mean of the matrix elements is 0. This constraint is always satisfied since $V_{\rm th}$ variation follows normal distribution [16] and the active MOSFET elements are the samples from the normal distribution.

c) Spectral radius constraint: As opposed to the above two constraints, the spectral radius constraint, which requires the spectral radius is about 1.0, is nontrivial. In order to compute the spectral radius of a matrix, all the matrix elements should be known in advance. However, as the matrix elements come from random variations, it is impossible to know them in advance. Satisfying this constraint without trial and error is the salient feature of the proposed method.

Circular law [17] is about the distribution of eigenvalues of a large square random matrix. The circular law states that for a "dense" matrix of size $N \times N$ whose elements follow normal distribution with mean 0 and standard deviation σ, the spectral radius converges to $\sigma\sqrt{N}$ as N becomes large. Now we want to know the spectral radius of a "sparse" matrix $\boldsymbol{W}$, where the size and the connectivity are $N \times N$ and C, respectively. The elements in this matrix follow normal distribution with mean 0 and standard deviation σ as well.

As mentioned above, the derived spectral radius bound for a dense matrix is $\sigma\sqrt{N}$, where N is the dimension of the (square) dense matrix with $N \times N$ non-zero elements. Consequently, we can see that the spectral radius of a sparse matrix with dimension $N \times N$ and connectivity C cannot be larger than $\sigma\sqrt{NC}$. This fact is also experimentally confirmed, where the spectral radius does converge to $\sigma\sqrt{NC}$ as NC becomes large. Fig. 6 shows the results of Monte Carlo simulations on the spectral radii of sparse random matrices with different sets of parameters, N, C and σ. The histograms of the radii divided by the estimation $\sigma\sqrt{NC}$ concentrate about 1, indicating that $\sigma\sqrt{NC}$ serves as a good bound.



(a) $N = 50$, $C = 0.1$, $\sigma = 1$ (b) $N = 200$, $C = 0.025$, $\sigma = 1$

(c) $N = 200$, $C = 0.1$, $\sigma = 0.5$ (d) $N = 200$, $C = 0.1$, $\sigma = 1$

Fig. 6. Distribution of normalized spectral radii for different sets of N, C and σ. The peak of the distribution converge 1.0 as NC becomes large. The number of samples is 10,000.

Considering that the elements of conductance difference follow a normal distribution of mean 0 and standard deviation of $\sqrt{2}{\rm A}\sigma_{\rm th}$ (see Eq. (6)), the spectral radius of the proposed MOSFET reservoir, $\rho_{\boldsymbol{W}}$, can be estimated as:

$$\rho_{\boldsymbol{W}} = \sqrt{2}{\rm A}\sigma_{\rm th}\sqrt{\rm NC}. \quad (7)$$

Then, the next objective is to find a way to adjust the spectral radius to the target value α. One way to achieve this is to multiply a scalar $\alpha/\rho_{\boldsymbol{W}}$ for all the elements in the matrix, but it is impossible to control the conductance of fabricated MOSFETs. Instead, according to Eq. (4), we propose to use the value of R_2 to control the spectral radius. More specifically, spectral radius becomes α when R_2 in SA is set as:

$$R_2 = \frac{\alpha}{\rho_{\boldsymbol{W}}} = \frac{\alpha}{\sqrt{2}{\rm A}\sigma_{\rm th}\sqrt{\rm NC}}. \quad (8)$$

Note that the parameters A, $\sigma_{\rm th}$, N, and C are all available once the process technology to fabricate the MOSFET reservoir and the structure of the reservoir is determined.

D. Accuracy Improvement with Inverted Inference

As observed later in the next section (e.g., in Fig. 8), the inference accuracy suffers from the errors generated in the process of circuit simulation. The main source of this errors is that, instead of a perfect Gaussian, the simulated weight matrices $\boldsymbol{W}_{\rm in}$ and $\boldsymbol{W}$ are slightly biased (mean-shifted). In other words, the simulated weight matrices used in the calculations are $\Theta_{\rm in} = \boldsymbol{W}_{\rm in} + \boldsymbol{E}_{\rm in}$ and $\Theta = \boldsymbol{W} + \boldsymbol{E}$. Since the errors do not have a mean of 0, the output of Eq. (2) becomes biased, and the inference accuracy drops.

To solve the issue, we propose a double reservoir structure, illustrated in Fig. 7. The key insight here is that, the mean-shift induced by the simulation errors can be cancelled by an inverted inference. More concretely, let $\boldsymbol{u}_{\rm inv} = 2 \cdot \mu - \boldsymbol{u}$, where μ is the mean of the inputs (μ is fixed to be 0.35 V in

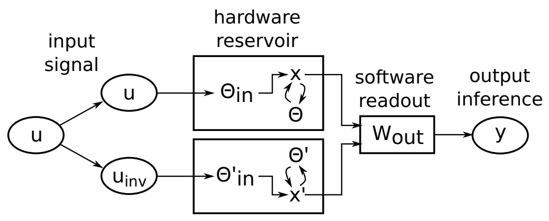


Fig. 7. The architecture of Dual-MOS-ESN that uses two reservoirs for normal and inverted inputs. The reservoir having counter input compensates the distortion observed when single reservoir is used.

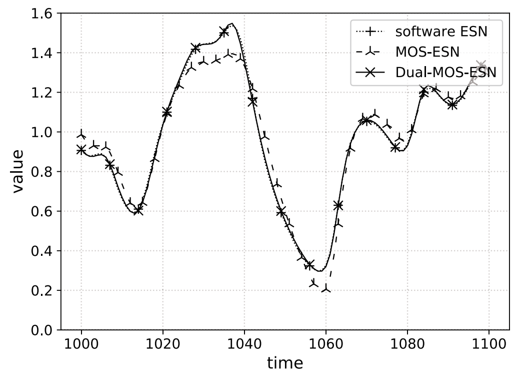


Fig. 8. Inference results of software ESN, MOS-ESN, and Dual-MOS-ESN.

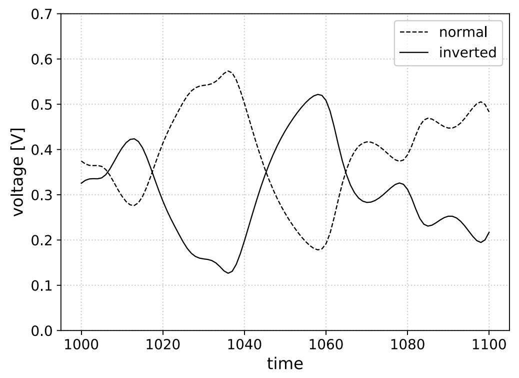


Fig. 9. Scaled normal and inverted Mackey-Glass series.

the experiment as shown in Fig. 9), instead of only computing Eq. 1 on $\boldsymbol{u}$, we also apply the same equation on $\boldsymbol{u}_{\mathrm{inv}}$, and the results are summed as

$$\begin{aligned}\boldsymbol{z} &= f_{\mathrm{SA}}(\Theta_{\mathrm{in}} \cdot \boldsymbol{u} + \Theta \cdot \boldsymbol{x}) + f_{\mathrm{SA}}(\Theta'_{\mathrm{in}} \cdot \boldsymbol{u}_{\mathrm{inv}} + \Theta' \cdot \boldsymbol{x}') \\ &= f_{\mathrm{SA}}(\boldsymbol{W}_{\mathrm{in}}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{u} + \boldsymbol{W}'_{\mathrm{in}}(2\mu - \boldsymbol{u}) + \boldsymbol{E}'\boldsymbol{u} + \Theta\boldsymbol{x} + \Theta'\boldsymbol{x}') \\ &= f_{\mathrm{SA}}((\boldsymbol{W}_{\mathrm{in}} - \boldsymbol{W}'_{\mathrm{in}})\boldsymbol{u} + (\boldsymbol{E} - \boldsymbol{E}')\boldsymbol{u} + c) \qquad (9)\end{aligned}$$

for some constant c, and the output can be calculated as $\boldsymbol{y} = \boldsymbol{W}_{\mathrm{out}} \cdot \boldsymbol{z}$. Here, f_{SA} is assumed to be a linear operator, and $(\Theta_{\mathrm{in}}, \Theta)$ and $(\Theta'_{\mathrm{in}}, \Theta')$ represent the two different hardware reservoirs in Fig. 7. By combining the results of the normal and inverted inferences, any mean shift induced by the simulation error $\boldsymbol{E}$ can be cancelled. Note that this technique also has practical significance, in the sense that real-world hardware also induces errors with non-zero means, and can still be cancelled by adopting the proposed double reservoir structure. The effectiveness of this technique is demonstrated in Section IV.IV-B.

IV. Numerical evaluation

A. MOS-ESN

In this section, the proposed ESN structures with single (MOS-ESN) and dual (Dual-MOS-ESN) hardware reservoirs are evaluated by the Mackey-Glass equation time-series dataset [18]. This is a nonlinear time-delayed differential equation written as:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \beta \frac{x(t-\tau)}{1 + (x(t-\tau))^n} - \gamma x(t), \qquad (10)$$

where β, γ are real numbers and τ is time delay of variable $x(t)$. A series data generated from this equation have a chaotic behavior. In addition, the data at time $t+1$ is generated depending on the present data and $t-\tau$. This time-delayed system is important for evaluating memory capacity.

In the evaluation, the proposed MOS-ESN is trained to estimate the next time value. When the input is the data at t, the target output value is set to the data at $t+1$. β, γ, τ, and n are set 0.25, 0.1, 17 and 10, respectively. These parameter values are taken from [11]. The beginning of the series, from $t = -\tau$ to -1, are 0 in both training and testing series. The data at $t = 0$ is set to 1.2 for the training and 0.2 for the testing. The different initial values lead to different series. This is to verify that the proposed MOS-ESN learns Eq. (10), not the training series. Note that the initial values are not restricted to these values.

The reservoir is simulated by a commercial circuit simulator [19] with a 65-nm device model. We first evaluate the MOS-ESN architecture based on Fig. 4. The size of input layer $K = 1$, the size of middle layer $N = 100$, and the connectivity of middle layer $C = 0.05$. R_0 is determined so that the voltage $I^{+}R_0$ fits within the output voltage range of the opamp. In this work, R_0 and R_1 are $1\,\mathrm{k}\Omega$. According to Eq. (8), R_2 is determined as $10\mathrm{k}\Omega$. The number of output layer L is 1, which is linear, and it is implemented using Python programming language. The loss function is mean square error, the optimizer is Adam [20], the length of training series and testing series are both 2000, batch size is 100, and epoch is 1000. All the simulations are executed on a workstation with Intel Core i9-7980XE processor running at 2.60 GHz with 128 GB of RAM.

The estimation result from the MOS-ESN and that from the software reservoir ESN written in Python are compared with the original series in Fig. 8. Inference by the software reservoir matches closely to the original series, while inferences by the MOS-ESN sees unignorable error particularly when the original series takes values at around the local maxima and local minima. Though the accuracy may be satisfactory depending on the application, the accuracy improvement is needed when higher accuracy is required.

B. Dual-MOS-ESN

In order to further improve the accuracy of the inference, Dual-MOS-ESN is then evaluated. The parameters of each layer are the same as that of MOS-ESN. By adding the inverted input (shown in Fig. 9) and a reservoir, inference error

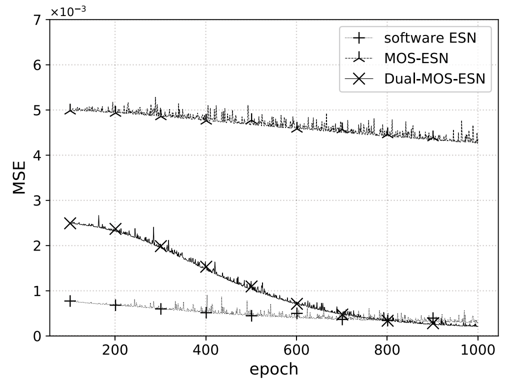


Fig. 10. Mean square errors of ESN simulations.

has been reduced to about 1/5 of MOS-ESN, from $-0.2 \sim 0.1$ to $-0.03 \sim 0.03$, also indicated in Fig. 8. Meanwhile, Fig. 10 shows the mean square error of reference software, MOS-ESN, and Dual-MOS-ESN. The Dual-MOS-ESN achieves better accuracy than MOS-ESN. After epoch 800, Dual-MOS-ESN fits best among three, which confirms the effectiveness of the proposed simple data augmentation.

V. Conclusion

In this paper, we proposed a new reservoir of the MOSFET crossbar array for building a weight-tuning-free ESN. The use of MOSFET enabled practical and stable implementation and its variations are effectively used as the source of the random weights. In order for the MOSFET reservoir to satisfy necessary constraints, we showed a method that autonomously controls the spectral radius of the random weight matrix without programming the weights represented by the MOSFET conductance. The evaluation using Mackey-Glass dataset showed that the proposed method successfully gave accurate inference. The Dual-MOS-ESN structure, consisting of two independent reservoirs by adding inverted input, further improved the inference accuracy.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant No. 17H01713. The authors also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[2] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.

[5] J. Herbert, "Adaptive Nonlinear System Identification with Echo State Networks," *Advances in Neural Information Processing Systems (NIPS)*, pp. 593–600, 2002.

[6] M. Wolfgang, M. Henry, and N. Thomas, "The "liquid computer": A novel strategy for real-time computing on time series," *Special Issue on Foundations of Information Processing of TELEMATIK*, vol. 8, no. 1, pp. 39–43, 2002.

[7] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017.

[8] ——, "Comparison between deepesns and gated rnns on multivariate time-series prediction," *CoRR*, vol. abs/1812.11527, 2018.

[9] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[10] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[11] A. M. Hassan, H. H. Li, and Y. Chen, "Hardware implementation of echo state networks using memristor double crossbar arrays," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2171–2177.

[12] P. Meuffels and R. Soni, "Fundamental issues and problems in the realization of memristors," 2012.

[13] S. Nafea, A. Dessouki, and E.-S. El-Rabaie, "Memristor overview up to 2015," vol. 24, pp. 79–106, 2015.

[14] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," *GMD-Report 148, German National Research Institute for Computer Science*, 01 2001.

[15] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural networks: the official journal of the International Neural Network Society*, vol. 20, pp. 391–403, 2007.

[16] K. Takeuchi, T. Tatsumi, and A. Furukawa, "Channel engineering for the reduction of random-dopant-placement-induced threshold voltage fluctuation," in *International Electron Devices Meeting Technical Digest*, 1997, pp. 841–844.

[17] T. Tao, V. Vu, and M. Krishnapur, "Random matrices: Universality of ESDs and the circular law," *Ann. Probab.*, vol. 38, no. 5, pp. 2023–2065, 2010.

[18] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[19] *HSPICE J-2014.09*, Synopsys, Inc., 2014.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conf. on Learning Representations*, 12 2014.