Computational Efficiency of Multi-Step Learning Echo State Networks for Nonlinear Time Series Prediction

Takanori Akiyama¹ and Gouhei Tanaka¹,²,³, (Member, IEEE)

¹Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo 113-8656, Japan
²International Research Center for Neurointelligence, Institute for Advanced Study, The University of Tokyo, Tokyo 113-0033, Japan
³Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo, Tokyo 113-8656, Japan

Corresponding author: Gouhei Tanaka (gtanaka@g.ecc.u-tokyo.ac.jp)

The work Gouhei Tanaka was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 20K11882; in part by the Japan Science and Technology Agency (JST)-Mirai Program under Grant JPMJMI19B1, Japan; and in part by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP16007.

ABSTRACT The echo state network (ESN) is a representative model for reservoir computing, which has been mainly used for temporal pattern recognition. Recent studies have shown that multi-reservoir ESN models constructed with multiple reservoirs can enhance the potential of the ESN-based approach. In the present study, we investigate computational performance and efficiency of the multi-step learning ESN which is one of the multi-reservoir ESN models and characterized by step-by-step learning processes. We show that the time complexity of the training algorithm of the multi-step learning ESN is equal to or smaller than that of the standard ESN. Our numerical experiments demonstrate that the multi-step learning ESN can achieve better or comparable performance with much less computational time compared to the standard ESN in nonlinear time series prediction tasks. Moreover, we reveal how the model architecture of the multi-step learning ESN is effective in comparison with other possible variant models. The step-by-step learning is applicable to general multi-reservoir systems and hardware for enhancement of their computational ability and efficiency.

INDEX TERMS Reservoir computing, time series prediction, nonlinear dynamical systems, linear regression, computational cost.

I. INTRODUCTION

Recurrent neural networks (RNNs) and their variants have been widely used for pattern recognition with time series data, such as nonlinear time series prediction and classification. The echo state network (ESN) [1], [2], originally derived as a special type of RNN, is a representative reservoir computing (RC) model consisting of a reservoir and a readout [3]. In the ESN, the reservoir is typically given by a sparse random recurrent neural network and the readout is by a linear regressor or classifier. Since the reservoir is fixed and only the readout is adjusted with a simple learning algorithm, the ESN can be trained much faster than other RNN-based models with gradient-based learning algorithms. In addition to this merit in software computation, the RC approach has a high potential for developing machine-learning hardware by exploiting a variety of physical phenomena for dynamic reservoirs [4].

The ESN transforms a given input time series into a high-dimensional feature space using an RNN-based reservoir and subsequently extracts desired information from the high-dimensional reservoir state in the readout [3], [4]. The ESN has universal approximation properties as a discrete-time temporal filter under certain conditions [5]. In practice, its computational performance highly depends on the design of the unadaptable reservoir part [6]. In general, the computational ability of the ESN is enhanced with an increase in the reservoir size (i.e. the number of nodes in the reservoir), because it means an increase in the number of trainable parameters in the readout and therefore enables to approximate a wider class of dynamical systems. However, some studies empirically demonstrated that the performance of the ESN often peaks out when the reservoir size is increased in several computational tasks [7], [8]. In addition,
an ill-posed problem in the training of the readout with linear regression can occur when the number of reservoir nodes is much larger than the length of the input time series data [9].

A possible option to overcome the above-mentioned limitation of the standard single-reservoir ESN is to consider multi-reservoir ESN models consisting of multiple RNN-based reservoir modules. The reservoir modules can be connected serially as in the deep ESN models [10]–[12], hierarchically as in the hierarchical reservoir model [13], or parallelly as in the grouped ESN [14], [15]. In these existing models, the model output is given by a weighted sum of the states from all the reservoir modules, and the readout training is conducted in one step. As another approach, we have proposed the multi-step learning ESN for time series prediction [16], inspired by the hybrid model combining a knowledge-based model and an ESN [7]. The multi-step learning ESN consists of a series of multiple ESN modules with additional connections, which are trained step by step to reduce the output error. After the step-by-step training, the final output error can be much smaller than the initial output error at the first ESN module. The aim of the multi-step learning ESN is to keep the prediction error as small as possible in each step, which is particularly beneficial to prevent a rapid expansion of the error in autoregressive prediction for chaotic dynamical systems.

In our previous study, we have conducted the performance analysis only for the 2-step case with two ESN modules [16]. We also derived a method to compute Lyapunov exponents of the 2-step learning ESN and evaluated the model performance considering memory-nonlinearity tradeoff [17]. Remaining issues are to clarify how the computational performance and training cost of the N-step learning ESN changes with increasing N and to reveal its difference from other variant models. We tackle these issues in the present study. We evaluate the computational performance and the training cost of the N-step learning ESN in three benchmark tasks of nonlinear time series prediction. The first task is to predict chaotic time series data generated by the Lorenz system [2], [18], [19]. The second task is to predict nonlinear time series data generated by the nonlinear autoregressive moving average (NARMA) model. The third task is to predict the time series data in the Santa Fe Laser Dataset. First, we demonstrate that the multi-step learning ESN can achieve better or comparable performance with a much lower training cost compared to the standard ESN when the total reservoir size is sufficiently large. Then, we show that the model architecture of the multi-step learning ESN is more effective than that of other possible variants.

The rest of this paper is organized as follows. In Sec. II, we introduce the standard ESN and then describe the multi-step learning ESN. We also compare the time complexity of the training process between them. In Sec. III, we show the computational performance and efficiency of the multi-step learning ESN in three nonlinear time series prediction tasks. In Sec. IV, we analyze the effectiveness of the architecture of the multi-step learning ESN in comparison with other variants. In Sec. V, we conclude this work.

II. METHODS

A. ECHO STATE NETWORK

We briefly explain the standard ESN [1], [3], since it is a basis of the multi-step learning ESN introduced in Sec. II-B. The model architecture is illustrated in Fig. 1. The ESN consists of an input layer, a reservoir, and a readout. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation with a sparse random RNN. The network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of an input layer, a reservoir, and a readout. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation. Each reservoir node is generated from a uniform distribution in $[-\sigma, \sigma]$ where $\sigma > 0$. The reservoir network structure is given by a sparse random graph where all the elements are scaled so that the spectral radius of the reservoir is equal to a given value $\rho$. The reservoir is normally given by a sparse random RNN. The randomization approach for neural network models has merits in simplicity of implementation, fast learning, and applicability of linear learning algorithms [20], [21]. In time series prediction, the aim is to construct an ESN model that well approximates a given input-output relation.
By inserting the righthand side of Eq. (2) into Eq. (1), we can obtain an input layer, a reservoir, and a readout. In addition, the output of the (n − 1)th module is fed into both the input layer and the readout of the nth module. The training of the whole model is completed after iterating N training steps.

\[ y(t) = f_{\text{out}}(r(t), \hat{W}_{\text{out}}) = f_{\text{out}}(A_1 r_1(t) + W_{\text{in}}^{(1)} u(t)), \]

where \( f_{\text{out}} \) is an element-wise activation function for \( n = 1, \ldots, N \). The first module is equivalent to the standard ESN. With an input time series \( u(t) \), the state vector \( r_1(t) \in \mathbb{R}^{D_1} \) of the reservoir of the first module is updated as follows:

\[
\begin{aligned}
    r_1(t + \Delta t) &= \tanh \left[ A_1 r_1(t) + W_{\text{in}}^{(1)} u(t) \right],
    \\
    y(t) &= f_{\text{out}}(r_1(t), \hat{W}_{\text{out}}^{(1)}),
\end{aligned}
\]

where \( W_{\text{out}}^{(1)} \in \mathbb{R}^{N_o \times D_1} \) and \( A_1 \in \mathbb{R}^{D_1 \times D_1} \) denote the input and reservoir weight matrices of the first module, respectively. The output weight matrix of the first module is trained as follows:

\[
\hat{W}_{\text{out}}^{(1)} = \arg\min_{W_{\text{out}}^{(1)}} \left[ \sum_{-T \leq t < 0} \left\| f_{\text{out}}(r_1(t), W_{\text{out}}^{(1)} y(t)) - y(t) \right\|^2 + \beta \left\| W_{\text{out}}^{(1)} \right\|^2 \right].
\]

The reservoir states can be produced with iterations of the above equation in a free-running way only with a given initial state.

**B. MULTI-STEP LEARNING ECHO STATE NETWORK**

The multi-step learning ESN is one of the multi-reservoir ESN models, which consists of multiple ESN modules and additional connections as illustrated in Fig. 2 [16]. The N-step learning ESN contains N ESN modules, each including an input layer, a reservoir, and a readout. In addition, the output of the (n − 1)th module is fed into both the input layer and the readout of the nth ESN module for \( n = 2, \ldots, N \). The N readouts are trained step by step, unlike other multi-reservoir ESN models with one-step training. The size of the reservoir of the n-th module is denoted by \( D_n \). The procedure of time series prediction with the N-step learning ESN is divided into an initialization process, N training processes, and a prediction process.

In the initialization process, we set the input and reservoir weight matrices of all the modules randomly as described in Sec. II-A. For simplicity, we use the same values of the hyperparameters \( \sigma, (d), \) and \( \rho \) in all the modules in this study.

In the training processes for time period \(-T \leq t \leq 0\), the output weight matrix \( W_{\text{out}}^{(n)} \) is trained one by one for the reservoir state \( r_n \) of the \( \text{nth} \) module is obtained by Eq. (1) where the reservoir state \( \hat{y}_n(t) \) can be used as the input at time \( t \).

\[
\hat{y}_n(t) = f_{\text{out}}(r_n(t), \hat{W}_{\text{out}}^{(n)}).
\]

where \( f_{\text{out}} \) is an element-wise activation function for \( n = 1 \).

For the nth \((2 \leq n \leq N)\) module, the input, reservoir, and output weight matrices are denoted by \( W_{\text{in}}^{(n)} \in \mathbb{R}^{D_n \times (N_o + N_n)} \), \( A_n \in \mathbb{R}^{D_n \times D_n} \), and \( W_{\text{out}}^{(n)} \in \mathbb{R}^{N_o \times (D_{n+1} + N_n)} \) respectively. The reservoir state vector and the output vector of the nth module are denoted by \( r_n \) and \( \hat{y}_n \), respectively. In the nth training step,
the input \( u(t) \) and the output \( \hat{y}_{n-1}(t) \) of the \((n-1)\)th module are concatenated and then fed into the input layer of the \(n\)th module. The reservoir state vector \( r_n(t) \in \mathbb{R}^{D_n} \) for \( n \geq 2 \) is updated as follows:

\[
r_n(t + \Delta t) = \tanh \left( A_n r_n(t) + W_{\text{in}}^{(n)} \left[ \hat{y}_{n-1}(t + \Delta t) \right] u(t) \right).
\]

(11)

With the reservoir states \( r_n(t) \) computed for \(-T \leq t \leq 0\), the output weight matrix of the \(n\)th module is determined as follows:

\[
\hat{W}_{\text{out}}^{(n)} = \arg\min_{W_{\text{out}}^{(n)}} \sum_{-T \leq t' \leq 0} \left\| g_{\text{out}} \left( \left( \hat{y}_{n-1}(t') \right) r_n(t), W_{\text{out}}^{(n)} \right) - y(t') \right\|^2 + \beta_n \left\| W_{\text{out}}^{(n)} \right\|^2.
\]

(12)

where \( \beta_n \) is the regularization parameter. The reservoir state vector of the \(n\)th module and the output \( \hat{y}_{n-1}(t) \) of the \((n-1)\)th module are concatenated to produce the output of the \(n\)th module. The output of the \(n\)th module is obtained with the optimized output weight matrix as follows:

\[
\hat{y}_n(t) = g_{\text{out}} \left( \left( \hat{y}_{n-1}(t) \right) r_n(t), \hat{W}_{\text{out}}^{(n)} \right).
\]

(13)

where \( g_{\text{out}} \) represents an element-wise output function for \( n \geq 2 \).

In the prediction process for a certain time period \((t \geq 0)\), a forward propagation of an unknown input time series in the whole model with the \(N\) trained output weight matrices \( \hat{W}_{\text{out}}(n = 1, \ldots, N) \) is performed to determine the output at the last ESN module, which is regarded as the model output given by

\[
\hat{y}(t) = \hat{y}_N(t).
\]

(14)

Particularly in autoregressive prediction, the target output is set as \( y(t) = u(t + \Delta t) \). In this case, the model output \( \hat{y}(t) \sim u(t + \Delta t) \) at time \( t \) can be used as the input at time \( t + \Delta t \). Once the initial condition \( \hat{y}(0) = u(\Delta t) \) is given, the trained model can autonomously produce the predicted values of output time series. In this case, the model is called a closed-loop model. We repeat the following three steps:

1) Compute the output of the first module as follows:

\[
r_1(t + \Delta t) = \tanh \left( A_1 r_1(t) + W_{\text{in}}^{(1)} \hat{y}(t) \right).
\]

(15)

\[
\hat{y}_1(t + \Delta t) = g_{\text{out}} \left( r_1(t + \Delta t), W_{\text{out}}^{(1)} \right).
\]

(16)

2) Compute the output of the \(n\)th module iteratively for \( n = 2, \ldots, N \) as follows:

\[
r_n(t + \Delta t) = \tanh \left( A_n r_n(t) + W_{\text{in}}^{(n)} \left[ \hat{y}_{n-1}(t + \Delta t) \hat{y}(t) \right] \right).
\]

(17)

\[
\hat{y}_n(t + \Delta t) = g_{\text{out}} \left( \left( \hat{y}_{n-1}(t + \Delta t) \right) r_n(t + \Delta t), W_{\text{out}}^{(n)} \right).
\]

(18)

3) Obtain the output as follows:

\[
\hat{y}(t + \Delta t) = \hat{y}_N(t + \Delta t).
\]

(19)

C. TIME COMPLEXITY

We estimate the time complexity of the training process of the \(N\)-step learning ESN model. In most cases, we can assume \( D_n \gg N_n \) and \( D_n \gg N_n \) for \( n = 1, \ldots, N \) [3]. The time complexity for the training process of the first module is given by \( O(D_1^2(T + D_1)) \) as explained in Sec. II-A. Since we feed the output of the \((n-1)\)th module into the \(n\)th module for \( n \geq 2 \), it is computationally efficient to store the predicted output \( \hat{y}_{n-1} \) of the \((n-1)\)th module for training the \(n\)th module. The time complexity for training the \(n\)th module is given by \( O(D_n^2(T + D_n)) \). Consequently, the total time complexity is represented as \( O(T \sum_{n=1}^N D_n^2 + \sum_{n=1}^N D_n^3) \) for the \(N\)-step learning ESN. We compare this with the time complexity of the training process of the standard ESN, i.e. \( O(D^2(T + D)) \). When the total number of reservoir nodes is the same in both models, i.e. \( \sum_{n=1}^N D_n = D \), we can show

\[
O \left( T \sum_{n=1}^N D_n^2 + \sum_{n=1}^N D_n^3 \right) \leq O(D^2(T + D))
\]

(20)

from the following Cauchy-Schwarz inequalities:

\[
\sum_{n=1}^N D_n^2 \leq \left( \sum_{n=1}^N D_n \right)^2,
\]

(21)

\[
\sum_{n=1}^N D_n^3 \leq \left( \sum_{n=1}^N D_n \right)^3.
\]

(22)

This means that the learning cost of the \(N\)-step learning ESN is equal to or smaller than that of the standard single-reservoir ESN if the total reservoir size is the same. Therefore, the computational efficiency can be enhanced by dividing a large reservoir into multiple smaller reservoirs and applying the above-mentioned step-by-step training method.

III. NONLINEAR TIME SERIES PREDICTION

We apply the multi-step learning ESN to three nonlinear time series prediction tasks and compare its predictive performance and computational efficiency with those of the standard ESN. In Sec. III-A, we perform a prediction of the Lorenz system by using a trained closed-loop model. In Sec. III-B, we conduct a prediction of the NARMA system. In Sec. III-C, we deal with a prediction task using a real time series data related to laser dynamics. Numerical experiments were performed on a computer with CPU, Intel Xeon E5-2650, 2.20GHz.

A. THE LORENZ SYSTEM

We evaluate the predictive performance of the multi-step learning ESN in a chaotic time series prediction task with the Lorenz system [24] which is described as follows:

\[
\begin{align*}
\frac{dx}{dt} &= 10(y - x), \\
\frac{dy}{dt} &= x(28 - z) - y, \\
\frac{dz}{dt} &= xy - \frac{8}{3}z.
\end{align*}
\]

(23)
This three-dimensional autonomous dynamical system exhibits chaotic behavior, for which a long-term prediction is difficult. The practical aim of the task is to predict the future state of the system accurately for as long as possible.

We generated the time series data from the Lorenz system. The time lengths of initial transient, training, and testing data were set at $T_{\text{init}} = 2$, $T_{\text{train}} = 100$, and $T_{\text{test}} = 25$, respectively. With sampling interval $\Delta t = 0.02$, the lengths of the initial transient, training, and testing data are 100, 5000, and 1250, respectively.

The input data fed into the model is represented as a three-dimensional vector $\mathbf{u}(t) = [x(t), y(t), z(t)]^T$ and the target output is as $\mathbf{u}(t + \Delta t)$. Following the method in Pathak et al. [7], we set the output functions of the multi-step learning ESN as follows:

$$f_{\text{out}}(\mathbf{r}, \mathbf{W}_{\text{out}}) = \mathbf{W}_{\text{out}} \mathbf{r}^*,$$

$$g_{\text{out}}((\hat{\mathbf{r}}), \hat{\mathbf{W}}_{\text{out}}) = \hat{\mathbf{W}}_{\text{out}} (\hat{\mathbf{r}}^*),$$

$$r_j^* = \begin{cases} r_j & (j: \text{odd}) \\ r_j^2 & (j: \text{even}) \end{cases},$$

where $r_j$ and $r_j^*$ represent the $j$th component of $\mathbf{r}$ and $\mathbf{r}^*$, respectively. We fixed the mean degree of the reservoir network at $\beta_0 = 3$ [7]. We optimized the hyperparameters of the ESN models based on the 10-step walk forward validation [25] considering the candidate values in Table 1.

To measure the predictive performance, we used the Valid Time [7] which is defined as the elapsed time $t_{\text{valid}}$ until the normalized prediction error $E(t)$ first exceeds a given value $\epsilon$ ($0 < \epsilon < 1$), where

$$E(t) = \frac{||\mathbf{u}(t) - \hat{\mathbf{u}}(t)||}{(||\mathbf{u}(t)||^2)^{1/2}}.$$  

We set $\epsilon = 0.4$ in this study, following [7]. This measure provides the duration of sufficiently accurate prediction made by the trained model.

An example of chaotic time series prediction with the 10-step learning ESN is demonstrated in Fig. 3. The total number of reservoir nodes is 1000. The number of each reservoir is given by $D_n = 100$ for $n = 1, 2, \ldots, 10$. The prediction is successful until $t = t_{\text{valid}} \sim 8.76$, after which the difference from the target output expands by the nature of chaos.

In the multi-step learning ESN, we can choose different reservoir sizes under a fixed total size. To systematically examine the effect of each reservoir size on the predictive performance, we computed the Valid Time for the 2-step learning ESNs with combinations of $D_1 \in [50, 400]$ and $D_2 \in [50, 400]$. For each setting of the reservoir size, the Valid Time averaged over 10 trials is plotted in Fig. 4. When $D_1$ is relatively small, i.e. $D_1 \leq 100$, the predictive performance is drastically improved by increasing $D_2$. In contrast, if $D_2$ is relatively small, i.e. $D_2 \leq 100$, an increase in $D_1$ does not significantly improve the prediction accuracy. This result means that better predictive performance can be achieved by making the second reservoir larger than the first reservoir.

Next, we compared the $N$-step learning ESN with the standard ESN in terms of predictive performance and learning cost (i.e., the program execution time for the training process). The total reservoir size was fixed at $D$ in both models. For a given $D = 100N$, we used the $N$-step learning ESN where each reservoir size is $D_n = 100$ for $n = 1, \ldots, N$. We performed 10 trials with different network realizations for each parameter setting. The Valid Time averaged over the trials is plotted against the total reservoir size $D$ in Fig. 5(a). The predictive performance of the multi-step learning ESN

| Mark | Meaning | Candidate values |
|------|---------|------------------|
| $\sigma$ | Input scaling | $10^{-2}, 2 \cdot 10^{-2}, 5 \cdot 10^{-2}, 0.1, 0.2, 0.5, 1$ |
| $\rho$ | Spectral radius | $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5$ |
| $\beta$ | Regularization factor | $10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1$ |

This three-dimensional autonomous dynamical system exhibits chaotic behavior, for which a long-term prediction is difficult. The practical aim of the task is to predict the future state of the system accurately for as long as possible.

We generated the time series data from the Lorenz system. The time lengths of initial transient, training, and testing data were set at $T_{\text{init}} = 2$, $T_{\text{train}} = 100$, and $T_{\text{test}} = 25$, respectively. With sampling interval $\Delta t = 0.02$, the lengths of the initial transient, training, and testing data are 100, 5000, and 1250, respectively.

The input data fed into the model is represented as a three-dimensional vector $\mathbf{u}(t) = [x(t), y(t), z(t)]^T$ and the target output is as $\mathbf{u}(t + \Delta t)$. Following the method in Pathak et al. [7], we set the output functions of the multi-step learning ESN as follows:

$$f_{\text{out}}(\mathbf{r}, \mathbf{W}_{\text{out}}) = \mathbf{W}_{\text{out}} \mathbf{r}^*,$$

$$g_{\text{out}}((\hat{\mathbf{r}}), \hat{\mathbf{W}}_{\text{out}}) = \hat{\mathbf{W}}_{\text{out}} (\hat{\mathbf{r}}^*),$$

$$r_j^* = \begin{cases} r_j & (j: \text{odd}) \\ r_j^2 & (j: \text{even}) \end{cases},$$

where $r_j$ and $r_j^*$ represent the $j$th component of $\mathbf{r}$ and $\mathbf{r}^*$, respectively. We fixed the mean degree of the reservoir network at $\beta_0 = 3$ [7]. We optimized the hyperparameters of the ESN models based on the 10-step walk forward validation [25] considering the candidate values in Table 1.

To measure the predictive performance, we used the Valid Time [7] which is defined as the elapsed time $t_{\text{valid}}$ until the normalized prediction error $E(t)$ first exceeds a given value $\epsilon$ ($0 < \epsilon < 1$), where

$$E(t) = \frac{||\mathbf{u}(t) - \hat{\mathbf{u}}(t)||}{(||\mathbf{u}(t)||^2)^{1/2}}.$$  

We set $\epsilon = 0.4$ in this study, following [7]. This measure provides the duration of sufficiently accurate prediction made by the trained model.

An example of chaotic time series prediction with the 10-step learning ESN is demonstrated in Fig. 3. The total number of reservoir nodes is 1000. The number of each reservoir is given by $D_n = 100$ for $n = 1, 2, \ldots, 10$. The prediction is successful until $t = t_{\text{valid}} \sim 8.76$, after which the difference from the target output expands by the nature of chaos.

In the multi-step learning ESN, we can choose different reservoir sizes under a fixed total size. To systematically examine the effect of each reservoir size on the predictive performance, we computed the Valid Time for the 2-step learning ESNs with combinations of $D_1 \in [50, 400]$ and $D_2 \in [50, 400]$. For each setting of the reservoir size, the Valid Time averaged over 10 trials is plotted in Fig. 4. When $D_1$ is relatively small, i.e. $D_1 \leq 100$, the predictive performance is drastically improved by increasing $D_2$. In contrast, if $D_2$ is relatively small, i.e. $D_2 \leq 100$, an increase in $D_1$ does not significantly improve the prediction accuracy. This result means that better predictive performance can be achieved by making the second reservoir larger than the first reservoir.

Next, we compared the $N$-step learning ESN with the standard ESN in terms of predictive performance and learning cost (i.e., the program execution time for the training process). The total reservoir size was fixed at $D$ in both models. For a given $D = 100N$, we used the $N$-step learning ESN where each reservoir size is $D_n = 100$ for $n = 1, \ldots, N$.

We performed 10 trials with different network realizations for each parameter setting. The Valid Time averaged over the trials is plotted against the total reservoir size $D$ in Fig. 5(a). The predictive performance of the multi-step learning ESN

![FIGURE 3. An example of a prediction with the 10-step learning ESN, where $D_n = 100$ ($n = 1, \ldots, 10$), for the Lorenz system. The parameter values were set at $\sigma = 0.02$, $\rho = 10^{-5}$, and $\beta = 10^{-6}$.](image)

![FIGURE 4. Dependence of the Valid Time on the two reservoir sizes, $D_1$ and $D_2$, of the 2-step learning ESN in the prediction task with the Lorenz system.](image)
FIGURE 5. Comparison of the standard ESN and the $N$-step learning ESN in the time series prediction task with the Lorenz system. The total reservoir size was fixed at $D$ in all the models. For a given $D = 100N$, each reservoir size of the $N$-step learning ESN is $D_n = 100$ for $n = 1, \ldots, N$. Each plot corresponds to an average value over 10 trials with different network realizations and the error bar shows the standard errors. (a) The Valid Time. (b) The program execution time for the training process.

increases with the number $N$ of the ESN modules when $D$ is relatively small. The average Valid Time of the multi-step learning ESN is larger than that of the standard ESN. Fig. 5(b) shows that the program execution time for the training process of the multi-step learning ESN is much shorter than that of the standard ESN especially for large $D$ values. The computational time for training the multi-step learning ESN increases linearly with the total reservoir size, whereas that for the standard ESN increases superlinearly. This numerical result agrees with the result of the time complexity analysis in Sec. II-C. As observed in Fig. 4, there is a possibility that the predictive performance of the multi-step learning ESN is further improved by setting $D_n$ ($n \geq 2$) larger than $D_1$ under a fixed number of $D$.

B. THE NARMA MODEL

We consider another prediction task with the NARMA model, which has often been used as a benchmark for the performance evaluation of RC models [26]–[29]. The NARMA model with order-$m$ is described as follows:

$$y(t) = \alpha y(t-1) + \beta y(t-1) \sum_{i=1}^{m} y(t-i) + \gamma u(t-m)u(t) + \delta,$$

where $t$ is the discrete time, $u(t)$ is the input signal randomly generated from a uniform distribution in [0, 0.5]. The parameter values were set at $\alpha = 0.3$, $\beta = 0.05$, $\gamma = 1.5$, and $\delta = 0.1$ as in the previous studies. The order $m$ controls how distant past states and inputs influence the current state in Eq. (28). The aim of this task is to predict the output time series $y(t)$ from the input time series $u(t)$ for a certain time period. This is often called the NARMA-$m$ task. To predict a NARMA model with a larger $m$, an ESN-based model needs to have more memory capacity [6].

We performed the NARMA-$10$ task by generating the time series data from Eq. (28) with $m = 10$. The lengths of initial transient, training, and testing data were set at 100, 3000, 1000, respectively.

In the multi-step learning ESN, we set the output functions as follows:

$$f_{out}(r, W_{out}) = W_{out}r,$$  \hspace{1cm} (29)

$$g_{out}(\tilde{y}, \tilde{W}_{out}) = \tilde{W}_{out}(\tilde{y}).$$  \hspace{1cm} (30)

The mean degree of the reservoir network was fixed at $\langle d \rangle = 3$ [7]. The other hyperparameter values were optimized considering the candidate values in Table 1 based on the 10-step walk forward validation [25].

We evaluated the predictive performance for the testing data using the three error indices including the root mean square error (RMSE), the normalized RMSE (NRMSE), and the mean absolute error (MAE), which are defined as follows:

$$\text{RMSE} = \left( \langle ||y(t) - \tilde{y}(t)||^2 \rangle_t \right)^{1/2},$$  \hspace{1cm} (31)

$$\text{NRMSE} = \text{RMSE} / \left( \langle ||\tilde{y}(t)||^2 \rangle_t \right)^{1/2},$$  \hspace{1cm} (32)

$$\text{MAE} = \langle ||y(t) - \tilde{y}(t)||_1 \rangle_t,$$  \hspace{1cm} (33)

where $\tilde{y}(t)$ is the model output, $y(t)$ is the target output, $\tilde{y}$ is the temporal average of $y(t)$, and $\langle \cdot \rangle_t$ represents a temporal average.

Fig. 6 demonstrates an example of a successful prediction with the 10-step learning ESN where $D_n = 200$ for $n = 1, \ldots, 10$. In this case, the RMSE, NRMSE, and MAE.
FIGURE 7. Comparison between the \(N\)-step learning ESN and the standard ESN in the NARMA-10 task. The total reservoir size was fixed at \(D\) in both models. For a given \(D = 200\), we used the \(N\)-step learning ESN where each reservoir size is \(D_n = 200\) for \(n = 1, \ldots, N\). Each plot corresponds to an average NRMSE value over 10 trials and the error bar shows the standard error. (a) Predictive performance. (b) Program execution time for the training process.

values for 1000 testing steps are \(1.1 \times 10^{-2}, 1.0 \times 10^{-1}, \) and \(9.1 \times 10^{-3}\), respectively.

Next, we compare the predictive performance between the 10-step learning ESN and the standard ESN. The total number of reservoir nodes is given by \(D\). For a given \(D = 200\), we used the \(N\)-step learning ESN where \(D_n = 200\) for \(n = 1, \ldots, N\). The results are shown in Fig. 7. The advantage of the multi-step learning ESN over the standard ESN in the predictive performance is clearly observed in Fig. 7(a) (see Supplementary Figure 1 for the RMSE and MAE). This is because the multi-step learning ESN is suited for approximating dynamical systems with strong nonlinearity rather than those with long-term memory as suggested in our previous study [16]. The computational efficiency of the multi-step learning ESN is confirmed in Fig. 7(b).

Fig. 8 shows how the output error of the first ESN module is reduced by the subsequent modules in the 5-step learning ESN with \(D_n = 100\) for \(n = 1, \ldots, 5\). The prediction error gradually decreases as the training process proceeds, and thus, the final prediction error is significantly smaller than the output error of the first module. This result demonstrates that the successive training processes in the multi-step learning ESN model is effective for improving the prediction accuracy.

C. LASER DATASET

We deal with a prediction task with the Santa Fe Laser Dataset [30], which is one of the benchmark dataset for time series prediction [27]. The aim of this task is to make a one-step ahead prediction for an experimental recording of the output power of a far-infrared laser in a chaotic regime. We used a part of the laser dataset. The lengths of training and testing data were set at 4000 and 1000, respectively.

In the multi-step learning ESN, we set the output functions as in Eqs. (29) and (30). The mean degree of the reservoir network was fixed at \(\langle d \rangle = 3\) [7]. The other hyperparameter values were optimized considering the candidate values in Table 1 based on the 10-step walk forward validation [25].

Fig. 9 demonstrates an example of a successful prediction with the 10-step learning ESN where \(D_n = 100\) for \(n = 1, \ldots, 10\). In this case, the RMSE, NRMSE, and MAE values for 1000 testing steps are given by \(3.4 \times 10^{-3}, 6.3 \times 10^{-2}, \) and \(1.8 \times 10^{-3}\), respectively.

We compared the predictive performance between the 10-step learning ESN and the standard ESN. The total number of reservoir nodes is given by \(D\) in both models. For a given \(D = 100\), we used the \(N\)-step learning ESN where \(D_n = 100\) for \(n = 1, \ldots, N\). The results are shown in Fig. 10. As shown in Fig. 10(a), the performance of the multi-step...
learning ESN is significantly improved by increasing the system size (see Supplementary Figure 2 for the RMSE and MAE). The multi-step learning ESN yields better predictive performance than the standard ESN. In addition, Fig. 10(b) shows that the multi-step learning ESN is more computationally efficient than the standard ESN.

IV. ANALYSIS OF THE MULTI-STEP LEARNING ESN
We analyze the role of network architecture in the multi-step learning ESN model. Similar network architectures with multiple reservoirs are used in other models such as the DeepESN model [10], [11] and the hierarchical reservoir model [13]. In the former model, multiple reservoirs are stacked, and their states are collected to train a single readout. In the latter model, multiple ESN modules, including an input layer, a reservoir, and a readout, are stacked, and the training of multiple readouts are sequentially conducted. Therefore, our model is close to the hierarchical reservoir model, but it has additional network connections as illustrated in Fig. 11 for the 2-step case. One is Connection-1 via which the input data is fed into all the ESN modules. The other is Connection-2 via which the model output of an ESN module is fed into the output layer of the subsequent module.

To clarify the roles of Connection-1 and Connection-2 in nonlinear time series prediction tasks, we compared the predictive performance of the following five different variant models:
1) ESN: the standard ESN model [1] (see Sec. II-A).
2) Multi-step ESN: the 2-step learning ESN.
3) Multi-step ESN (wo1): the 2-step learning ESN without Connection-1.
4) Multi-step ESN (wo2): the 2-step learning ESN without Connection-2.
5) Multi-step ESN (wo1,2): the 2-step learning ESN without Connection-1 and Connection-2, which is equivalent to the hierarchical reservoir model [13].

The performance was evaluated in the three time series prediction tasks used in Sec. III. The total reservoir size was fixed at $D$ in all the variant models. In the 2-reservoir models, the sizes of reservoirs were set at $D_1 = D_2 = D/2$. For each parameter setting of each model, we performed 10 simulation trials. The other experimental conditions are the same as those in Sec. III.

Fig. 12(a) shows the results of the comparison among the five variant models in the Lorenz system prediction. The Valid Time was used to evaluate the predictive performance as in Sec. III-A. In Fig. 12(a), the best performance is obtained by the Multi-step ESN having Connection-1 and Connection-2 when the reservoir size is sufficiently large (i.e. $D \geq 400$). The Multi-step ESN (wo1) and Multi-step
FIGURE 12. Comparison among the five variant models for different total reservoir size $D$. In the Multi-step ESNs, the two reservoirs have the same size ($D_1 = D_2 = D/2$). Each plot shows the average value over 10 trials and the error bar indicates the standard error. (a) The Valid Time in the time series prediction task with the Lorenz system. (b) The NRMSE in the NARMA-10 task. (c) The NRMSE in the laser task.

**V. CONCLUSION**

We have demonstrated that the multi-step learning ESN is an effective and efficient multi-reservoir ESN model. In the training of the multi-step learning ESN, each ESN module is trained step by step with linear regression. We have analytically and numerically shown that this training method requires less computational cost than that of the standard ESN with the same total reservoir size. In addition, we have shown in the tested tasks that the computational performance of the multi-step learning ESN is better than that of the standard ESN when the total reservoir size is sufficiently large. In short, our results indicate that it is effective to separate a single large-scale reservoir into multiple reservoirs and apply the step-by-step learning method instead of the one-shot learning, for enhancement of computational performance and efficiency of the ESN-based approach. This strategy would be applicable to improving the performance of other reservoir computing systems and hardware [4], particularly when exhaustive efforts are needed for implementing a large-scale reservoir.

In nonlinear time series prediction, the task difficulty is involved with memory and nonlinearity of a dynamical system behind the time series data. The task is more difficult when the data has longer-term correlations and exhibits stronger nonlinear phenomena. Considering the results of the subsequent ESN module, contributing to enhancement of the prediction ability of the model.
our previous study [16] and the present study, we can
derive the conclusion that the multi-step learning ESN
is suited for predicting highly nonlinear behavior like chaotic
one. If time series data with long-term correlations are
handled, the reservoir size of each ESN module needs to be
enlarged so that the echo of input information is maintained
for a long term within the reservoir. A future work is to
apply the multi-step learning ESN to a prediction of
higher-dimensional chaotic systems [18] and other real-world
data. Another remaining work for practical applications is
performance comparison between the multi-step learning
ESN and other state-of-the-art machine learning models in
time series prediction.

CONFLICT OF INTEREST
The authors have no conflict of interest to mention.

REFERENCES
[1] H. Jaeger, “The ‘echo state’ approach to analysing and training recurrent
neural networks-with an Erratum note,” GMD-German Nat. Res. Inst.
Comput. Sci., Bonn, Germany, GMD Rep. 148, 2001.
[2] H. Jaeger and H. Haas, “Harnessing nonlinearity: Predicting chaotic
systems and saving energy in wireless communication,” Science, vol. 304,
no. 5667, pp. 78–80, Apr. 2004.
[3] M. Lukoševičius and H. Jaeger, “Reservoir computing approaches to
recurrent neural network training,” Comput. Sci. Res., vol. 3, no. 3,
pp. 127–149, 2009.
[4] 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 Netw., vol. 115, pp. 100–123,
Jul. 2019.
[5] L. Grigoryeva and J.-P. Ortega, “Echo state networks are universal,”
Neural Netw., vol. 108, pp. 495–508, Dec. 2018.
[6] H. Jaeger, “Short term memory in echo state networks,” GMD-German
Nat. Res. Inst. Comput. Sci., Bonn, Germany, GMD Rep. 152, 2002.
[7] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan,
and E. Ott, “Hybrid forecasting of chaotic processes: Using machine learning
in conjunction with a knowledge-based model,” Chaos, Interdiscipl.
Nonlinear Sci., vol. 28, no. 4, Apr. 2018, Art. no. 041101.
[8] A. Chattopadhyay, P. Hassanazadeh, and D. Subramanian, “Data-driven
predictions of a multiscale Lorenz 96 chaotic system using machine-
learning methods: Reservoir computing, artificial neural network, and long
short-term memory network,” Nonlinear Processes Geophys., vol. 27,
no. 3, pp. 373–389, Jul. 2020.
[9] M. Lukoševičius, “A practical guide to applying echo state networks,” in
Neural Networks: Tricks of the Trade. Berlin, Germany: Springer, 2012,
pp. 659–686.
[10] Q. Mu, L. Shen, and G. W. Cottrell, “Deep-ESN: A multiple
projection-encoding hierarchical reservoir computing framework,” 2017,
arXiv:1711.05255.
[11] C. Gallicchio, A. Micheli, and L. Silvestri, “Local Lyapunov exponents
of deep echo state networks,” Neurocomputing, vol. 298, pp. 34–45,
Jul. 2018.
[12] Z. Li and G. Tanaka, “Multi-reservoir echo state networks with
sequence resampling for nonlinear time-series prediction,” Neurocomputing,
vol. 467, pp. 115–129, Jan. 2022.
[13] F. Trifonbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, “Phoneme
recognition with large hierarchical reservoirs,” in Proc. Adv. Neural Inf.
Process. Syst., 2010, pp. 2307–2315.
[14] C. Gallicchio, A. Micheli, and L. Pedrelli, “Deep reservoir computing: A
critical experimental analysis,” Neurocomputing, vol. 268, pp. 87–99,
Dec. 2017.
[15] Z. Li and G. Tanaka, “HP-ESN: Echo state networks combined with
Hodrick-Prescott filter for nonlinear time-series prediction,” in Proc. Int.
Joint Conf. Neural Netw. (IJCNN), Jul. 2020, pp. 1–9.
[16] T. Akiyama and G. Tanaka, “Analysis on characteristics of multi-step
learning echo state networks for nonlinear time series prediction,” in Proc. Int.
Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.
[17] M. Inubushi and K. Yoshimura, “Reservoir computing beyond memory-
nonlinearity trade-off,” Sci. Rep., vol. 7, no. 1, p. 10199, Dec. 2017.
[18] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of
large spatiotemporally chaotic systems from data: A reservoir computing
approach,” Phys. Rev. Lett., vol. 120, no. 2, Jan. 2018, Art. no. 024102.
[19] P. Antonik, M. Galina, J. Pauwels, and S. Massar, “Using a reservoir
computer to learn chaotic attractors, with applications to chaos synchro-
nization and cryptography,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat.
Interdiscip. Top., vol. 98, no. 1, Jul. 2018, Art. no. 012215.
[20] S. Scardapane and D. Wang, “Randomness in neural networks: An
overview,” Wiley Interdiscip. Rev., Data Mining Knowl. Discovery,
vol. 7, no. 2, p. e1200, Mar. 2017.
[21] D. Wang and M. Li, “Stochastic configuration networks: Fundamentals
and algorithms,” IEEE Trans. Cybern., vol. 47, no. 10, pp. 3466–3479,
Oct. 2017.
[22] P. Erdos and A. Rényi, “On the evolution of random graphs,” Publ. Math.
Inst. Hung. Acad. Sci., vol. 5, no. 1, pp. 17–60, 1960.
[23] A. N. Tikhonov, “On the solution of ill-posed problems and the method of
regularization,” Dokl. Akad. Nauk, vol. 151, no. 3, pp. 501–504, 1963.
[24] E. N. Lorenz, “Deterministic nonperiodic flow,” J. Atmos. Sci., vol. 20,
no. 2, pp. 130–141, 1963.
[25] M. Lukoševičius and A. Uselis, “Efficient cross-validation of echo state
networks,” in Proc. Int. Conf. Artif. Neural Netw. Cham, Switzerland:
Springer, 2019, pp. 121–133.
[26] H. Jaeger, “Adaptive nonlinear system identification with echo state
networks,” in Proc. Adv. Neural Inf. Process. Syst., 2003, pp. 609–616.
[27] A. Rodan and P. Tino, “Minimum complexity echo state network,” IEEE
Trans. Neural Netw., vol. 22, no. 1, pp. 131–144, Jan. 2011.
[28] T. Strauss, W. Wustlich, and R. Labahn, “Design strategies for weight
matrices of echo state networks,” Neural Comput., vol. 24, no. 12,
pp. 3246–3276, Dec. 2012.
[29] A. Goulard, A. Shahani, and D. Stefanovic, “Exploring transfer function
nonlinearity in echo state networks,” in Proc. IEEE Symp. Comput. Intell.
Secur. Defense Appl. (CISDA), May 2015, pp. 1–8.
[30] A. S. Weigend and N. A. Gershenfeld, “Results of the time series
prediction competition at the Santa Fe Institute,” in Proc. IEEE Int. Conf.
Neural Netw., Mar./Apr. 1993, pp. 1786–1793.

TAKANORI AKIYAMA received the B.E. degree in
mathematical engineering and the M.S. degree in
mathematical informatics from The University of Tokyo,
Tokyo, Japan, in 2018 and 2020, respectively.
His research interests include machine learning
and neural networks.

GOUEI TANAKA (Member, IEEE) received the B.E.
degree in mathematical engineering and the M.S. and Ph.D.
degrees in complexity science from The University of Tokyo,
Tokyo, Japan, in 2000, 2002, and 2005, respectively.
He is currently a Project Associate Professor with the
International Research Center for Neurointelligence,
Graduate School of Engineering,
and the Graduate School of Information
Science and Technology, The University of Tokyo.
His research interests include mathematical engineering, nonlinear science,
network science, and their applications to real-world phenomena from
complex systems viewpoint.
Dr. Tanaka is also serving as a member of IEEE Task Force on Reservoir
Computing and a member of the IEEE Computational Intelligence Society.

* * *