Back-Propagation Learning of Infinite-Dimensional Dynamical Systems

Isao Tokuda

Department of Computer Science and Systems Engineering,

Muroran Institute of Technology, Muroran, Hokkaido 050-8585, Japan

Ryuji Tokunaga

Institute of Information Sciences and Electronics, University of Tsukuba, Ibaraki 305-8573, Japan

Kazuyuki Aihara

Department of Mathematical Engineering and Information Physics, Faculty of Engineering, The University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan;

CREST, Japan Science and Technology Corporation (JST), 4-1-8 Hon-cho, Kawaguchi, Saitama 332-0012, Japan

Requests for reprints should be sent to Dr. I. Tokuda, Dept. Comp. Sci. Sys. Engin., Muroran Institute of Technology, Muroran, Hokkaido 050-8585, Japan (E-mail: tokuda@csse.muroran-it.ac.jp).

Running title: Learning with Delays

Abstract

This paper presents numerical studies of applying back-propagation learning to a delayed recurrent neural network (DRNN). The DRNN is a continuous-time recurrent neural network having time delayed feedbacks and the back-propagation learning is to teach spatio-temporal dynamics to the DRNN. Since the time-delays make the dynamics of the DRNN infinite-dimensional, the learning algorithm and the learning capability of the DRNN are different from those of the ordinary recurrent neural network (ORNN) having no time-delays. First, two types of learning algorithms are developed for a class of DRNNs. Then, using chaotic signals generated from the Mackey-Glass equation and the Rössler equations, learning capability of the DRNN is examined. Comparing the learning algorithms, learning capability, and robustness against noise of the DRNN with those of the ORNN and time delay neural network (TDNN), advantages as well as disadvantages of the DRNN are investigated.

Key words: back-propagation learning, time-delay, recurrent neural network, retarded functional differential equations, infinite-dimensional dynamical system

1. Introduction

In biological neural networks, various types of *time delays* such as axonal propagation delays and synaptic transmission delays are experimentally observed. Natural direction of the neural network studies is to consider functions of such time delays in neural systems. So far, many neural network models with time delays have been introduced and possible functions of the delays have been discussed. For instance, delayed synaptic connections were introduced to neural networks to solve time-sequence recognition problem (Tank & Hopfield, 1987; Hopfield & Tank, 1989; Anderson & van Essen, 1987; Mozer, 1989; Lapedes & Farber, 1987). In the network, the delayed synapses function for concentrating input information in time and for recognizing input time-sequence patterns. Such time-delay neural networks (TDNNs) have been widely applied to practical engineering problems such as speech recognition (Tank & Hopfield, 1987; Hopfield & Tank, 1989; Anderson & van Essen, 1987; Mozer, 1989) and nonlinear predictions (Lapedes & Farber, 1987). Delayed feedback connections have been considered also in a Hebbian-type associative memory neural network (Sompolinsky & Kanter, 1986; Kleinfeld, 1986; Herz *et al.*, 1989). In this network, time delays destabilize memory states of the Hebbian-type neural network and enable the network to sequentially recall a set of the memories. Stability of time-delayed neural networks has been also analyzed extensively (Marcus & Westervelt, 1989; Baldi & Atiya, 1994; Cao & Wu, 1996; Pakdaman & Malta, 1998; Joy, 2000; Lu, 2000; Liao, Chen, & Sanchez, 2002; Peng, Qiao, & Xu, 2002).

Although the neural network models with time delays have been mainly studied in the above contexts, the focus of the present paper is rather different from these studies. Our interest here is in supervised learning of a delayed recurrent neural network (DRNN). The DRNN stands for a continuous-time recurrent neural network that has time delayed feedbacks. The supervised learning is to teach spatio-temporal dynamics to the DRNN by applying the back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986). Since the dynamics of the DRNN is described by retarded functional differential equations (RFDEs) (Hale, 1977), whose dynamical class is different from that of an ordinary differential equation model of recurrent neural network (ORNN), the learning algorithm and the learning capability of the DRNN are different from those of the conventional ORNN (Pineda, 1987; Doya & Yoshizawa, 1989; Pearlmuter, 1989; Sato, 1990; Gouhara *et al.*, 1992).

To our knowledge, the supervised learning of the DRNN has not yet been thoroughly investigated. So far, back-propagation learning has been introduced to DRNN and the network capability of learning complex dynamics such as chaos and speech has been examined (Tokuda, Hirai, & Tokunaga, 1993; Baldi & Atiya, 1994; Tokuda, Tokunaga, & Aihara, 1996; Tokuda & Aihara, 1997). Adaptive simulated annealing algorithm has been examined for efficient training of DRNN (Cohen, Saad, & Marom, 1997). A class of dynamical systems approximated by DRNNs has been also discussed (Tokuda, 1998).

The aim of the present paper is to consider possible functions of *time delays* in neural networks in the light of the supervised learning in DRNN. On the basis of the comparative studies with ORNN and TDNN, advantages as well as disadvantages of the DRNN are investigated. Robustness of the DRNN against noise is also studied.

This paper is organized as follows. In Section 2, using a standard mathematical model for DRNN, learning algorithms are introduced. The computational costs of the learning algorithms of DRNN are compared with those of ORNN. In Section 3, several numerical experiments are carried out to study learning capability of the DRNN. Based on comparative experiments with ORNN and TDNN, advantages and disadvantages of the DRNN are investigated. The final section is devoted for conclusions and discussions of the present work.

2. Learning Algorithms

2. 1 Recurrent Neural Network with Time Delays

As a standard model for DRNN, let us consider the Kleinfeld model (1986):

$$\begin{aligned}\frac{d}{dt}x^i(t) &= -\frac{1}{R}\cdot x^i(t) + \sum_{j=1}^{N} W_{ij}\cdot V_j(t) + \sum_{j=1}^{N} D_{ij}\cdot V_{\mathcal{D}_j}(t) + \sum_{j=1}^{m} I_{ij}\cdot z_j(t), \\ & V_j(t) = G(x^j(t)),\ \ V_{\mathcal{D}_j}(t) = \int_{-\tau}^{0} V_j(t+s)\cdot \mathcal{D}_j(s)\, ds, \qquad (1)\end{aligned}$$

which has been introduced as a network to generate a set of memory patterns sequentially in time. As is shown in Fig. 1, the network is composed of N-neurons, whose internal states, outputs, and delayed outputs are respectively denoted as $x_i(t)$, $V_i(t)$, and $V_{\mathcal{D}_i}(t)$ $(i = 1, \ldots, N)$. The network also receives m-external time-dependent inputs $z_i(t)$ $(i = 1, 2, \ldots, m)$. Synaptic connections from the j-th neuron, the j-th delayed neuron, and the j-th external input to the i-th neuron are described by the connection weights W_{ij},

D_{ij}, and I_{ij}, respectively. The delay function $\mathcal{D}_j(s)$ represents a response characteristic of the delayed synapse, which is integrated over a cut-off duration of τ. The delay function usually takes the form of delta function, $\mathcal{D}_j(s) = \delta(s - \tau_j)$, step function, $\mathcal{D}_j(s) = \Theta(\tau_j - s)/\tau_j$, or exponential decay function, $\mathcal{D}_j(s) = \exp(-s/\tau_j)/\tau_j$. The input-output function $G(x)$ is represented by a monotonously-increasing function such as the sigmoidal function $G(x) = 2/\{1+\exp(-x)\}-1$. In case when there is no time-delay, *i.e.*, $\tau = 0$, the network of Eq. (1) becomes an ORNN widely used for associative memory neural networks (Hopfield & Tank, 1985) and recurrent back-propagation neural networks (Pineda, 1987; Doya & Yoshizawa, 1989; Pearlmuter, 1989; Sato, 1990; Gouhara *et al.*, 1992).

Equation (1) belongs to a class of RFDEs (Hale, 1977) described as:

$$\frac{d}{dt}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega), \tag{2}$$

where $\boldsymbol{x} = \{x^1, \ldots, x^N\}$, $\boldsymbol{z} = \{z_1, \ldots, z_m\}$, $\boldsymbol{f} = \{f_1, \ldots, f_N\}$, $\Omega = \{W_{ij}, D_{ij}, I_{ij}\} \in R^K$, and $\boldsymbol{f} : C([-\tau, 0], R^N) \times R^m \times R^K \to R^N$. The dynamical system is infinite dimensional, since the state space $\boldsymbol{x}_t \in C([-\tau, 0], R^N)$ is represented by a continuous mapping of the interval $[-\tau, 0]$ into R^N according to $\boldsymbol{x}_t(\theta) = \boldsymbol{x}(t + \theta)$ for $\theta \in [-\tau, 0]$.

In the followings, learning algorithms for adjusting the weight parameters Ω are developed for a class of DRNNs described by Eq. (2).

2. 2 Problem Formulation

Let us consider a supervised learning to teach spatio-temporal dynamics to the DRNN of Eq. (2). The inverse problem for learning spatio-temporal dynamics can be stated as follows:

> "Classify the units of the DRNN into visible units $\{x^i | i \in \mathcal{V}\}$ and hidden units $\{x^i | i \notin \mathcal{V}\}$. Given an initial condition $\boldsymbol{x}_0$ and external inputs $\boldsymbol{z}(t)$, the DRNN of Eq. (2) gives rise to a unique solution $\boldsymbol{x}(t)$ satisfying $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega)$ for $t \in [0, T]$. Then, find the weight parameters Ω that give rise to a solution $\boldsymbol{x}(t)$ approximately following a teacher signal $\boldsymbol{\xi}(t) = \{\xi_i(t) | i \in \mathcal{V}\}$ as $x^i(t) \approx \xi_i(t)$ for $i \in \mathcal{V}$ and $t \in [-\tau, T]$."

The back-propagation learning algorithm for the inverse problem can be formulated as follows. First, the cost function is defined for the weight parameters Ω as

$$E(\Omega) = \int_0^T \frac{1}{2} \cdot \sum_{i \in \mathcal{V}} \{x^i(t) - \xi_i(t)\}^2 \; dt. \tag{3}$$

Then, the cost function (3) is minimized by the steepest descent method

$$\omega^{\text{new}} = \omega^{\text{old}} - \eta \cdot \frac{\partial E}{\partial \omega}(\Omega^{\text{old}}), \tag{4}$$

where ω ($\in \Omega$) stands for an element of the weight parameters and η is a learning speed.

The main part of the back-propagation learning is the computation of the first derivatives $\partial E / \partial \omega$ in Eq. (4). To compute the first derivatives for recurrent neural networks, there are mainly two algorithms: (1) the real time recurrent learning (RTRL) algorithm and (2) the time-dependent recurrent learning (TDRL) algorithm. In the following subsections, the two algorithms are introduced to DRNN.

2. 3 Real Time Recurrent Learning

In the RTRL algorithm, the first derivatives $\partial E/\partial\omega$ are computed as

$$\frac{\partial E}{\partial \omega} = \int_0^T \sum_{i\in\mathcal{V}} \{x^i(t) - \xi_i(t)\} \cdot \frac{\partial x^i}{\partial \omega}(t) \ \ dt, \tag{5}$$

where the partial derivatives $\partial x^i/\partial\omega$ in the right hand side are calculated by solving the first variational equations of Eq. (2) as

$$\begin{aligned} \frac{d}{dt}(\frac{\partial x^i}{\partial \omega})(t) \quad &= \quad \sum_{j=1}^{N} [\int_{-\tau}^{0} \frac{\partial f_i}{\partial x_t^j(s)}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega) \cdot \frac{\partial x^j}{\partial \omega}(t+s) \ \ ds \ ] \\ &\quad + \frac{\partial f_i}{\partial \omega}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega) \qquad (i = 1, 2, \ldots, N) \end{aligned} \tag{6}$$

with an initial condition $\partial x^i/\partial\omega = 0$ for $t \in [-\tau, 0]$. In Appendix B, explicit forms of the variational equations (6) are provided for the Kleinfeld model (1).

In the study of training temporal pattern generator networks, generalized learning algorithm has been derived for recurrent networks with arbitrary dynamic operators that can be applied to both continuous-time and discrete-time networks (Doya, 1991; Doya & Yoshizawa, 1991; Doya, 1993). We note that straightforward application of the generalized algorithm gives the same variational equations (6). We also note that, in case when there is no time delay, *i.e.*, $\tau = 0$, the present algorithm is equivalent to the RTRL algorithm for ORNN (Doya & Yoshizawa, 1989; Williams & Zipser, 1989).

2. 4 Time-Dependent Recurrent Learning

In the TDRL algorithm, the first derivatives $\partial E/\partial\omega$ are computed by using the Lagrange multipliers $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ as follows (see Appendix A for detailed derivation of the algorithm).

[**TDRL Algorithm**]

(i) For a given initial condition $\boldsymbol{x}_0$ and external inputs $\boldsymbol{z}(t)$, solve Eq. (2) forward in time. The solution curve $\boldsymbol{x}(t)$ is stored in the computer memory for a time interval of $t \in [0, T]$.

(ii) For a boundary condition $\boldsymbol{\lambda}(T) = 0$ and for a teacher signal $\boldsymbol{\xi}(t)$, Lagrange multipliers $\boldsymbol{\lambda}(t)$ are calculated for $t \in [0, T]$ by solving the following equations backward in time:

$$\begin{aligned} \frac{d}{dt}\lambda_i(t) \quad = \quad & \delta_{i \in \mathcal{V}} \cdot \{\xi_i(t) - x^i(t)\} - \sum_{j=1}^{N} \int_{-\tau}^{0} \delta_{t \in [0, T+s]} \cdot \lambda_j(t-s) \cdot \\ & \frac{\partial f_j}{\partial x^i_{t-s}(s)} (\boldsymbol{x}_{t-s}, \boldsymbol{z}(t-s), \Omega) \;\; ds. \end{aligned} \tag{7}$$

(iii) Using the Lagrange multipliers $\boldsymbol{\lambda}(t)$, the first derivatives are integrated as

$$\frac{\partial L}{\partial \omega} = \int_0^T \sum_{i=1}^{N} \lambda_i(t) \cdot \frac{\partial f_i}{\partial \omega} (\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega) \;\; dt. \tag{8}$$

In case when there is no time delay, *i.e.*, $\tau = 0$, the above algorithm is equivalent to the TDRL algorithm for ORNN (Pearlmuter, 1989; Sato, 1990). In Appendix B, explicit forms of the TDRL algorithm are provided for the Kleinfeld model (1).

2. 5 Computational Costs

Let us compare the computational costs of the learning algorithms of the DRNN with those of the ORNN.

(A) In the RTRL algorithm, the main computational part is the numerical integration of the variational equations (6). In case of DRNN, the variational equations (6) are $N{\cdot}K$ sets of RFDEs, whereas the variational equations (6) are $N{\cdot}K$ sets of ordinary differential equations in case of ORNN. The RTRL algorithm for the DRNN is computationally costly compared to that of the ORNN, since the RFDEs (6) are numerically treated as follows.

In the numerical integration of RFDEs of the form $\dot{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{y}_t)$ $(\boldsymbol{y} = \{\partial x^i/\partial\omega\} \in R^{N\times K})$, its state space is represented as

$$\{\boldsymbol{y}_t(\theta) \mid \theta = 0, \frac{\tau}{J_1 - 1}, \frac{2{\cdot}\tau}{J_1 - 1}, \ldots, \tau\} \tag{9}$$

using J_1 sets of dynamical variables $\boldsymbol{y}_t(\theta)$ divided with a time interval of $\tau/(J_1 - 1)$. In the numerical integration of the $N{\cdot}K$ sets of RFDEs, the memory size in the order of $N{\cdot}K{\cdot}J_1$ is required. This memory requirement is much larger than that of the ORNN, which needs a memory size in the order of only $N{\cdot}K$.

(B) In the TDRL algorithm, the main computational part is the computation of the Lagrange multipliers by numerical integration of the backward equations (7). This is computationally less expensive than the RTRL algorithm because the backward equations are merely N sets of RFDEs.

In the forward integration of the system equations (2), the solution curve $\boldsymbol{x}(t)$ $(t \in [0, T])$ is stored in the computer memory. Supposing that J_2 time steps between 0 and T of the dynamical states $\boldsymbol{x}(t)$ are stored, the memory size in the order of $N{\cdot}J_2$ is required. The memory requirement for the DRNN is comparable with that for the ORNN, since the ORNN needs the same amount of computer memory for storing its trajectory. As the time interval T is set to be very long, this memory requirement may become huge. However, if we set the interval T short enough as described in subsection 3. 2, the memory can be set to be within a practical computational size. Taking into account the fact that the TDRL algorithm is computationally much less expensive than the RTRL algorithm, the TDRL algorithm is more practical than the RTRL algorithm, except for the case when on-line learning is required.

In Table 1, learning algorithms and their computational costs are summarized for DRNN and ORNN.

2. 6 Training of Hidden Units

In the previous subsections, learning algorithms for DRNN have been discussed mainly for the optimization of the weight parameters. It is in principle possible to include other parameters to be optimized by the learning algorithms such as an initial condition of the hidden units. In fact, it has been reported that optimization of the initial condition of the hidden units is crucial especially to learn complex nonlinear dynamics such as chaos by ORNN. Since the chaotic property of sensitive dependence on initial conditions implies that a slight error in the initial condition of the hidden units diverges in time and strongly perturbs the trajectory of the visible units, the initial condition of the hidden units should be carefully determined. In the numerical studies (Sato, Murakami, & Joe, 1990; Sato, Joe, & Hirahara, 1990), the initial condition of the hidden units has been adjusted as the learning parameters. This adjustment seems to complicate the learning algorithm as well as the learning process of the ORNN. The situation becomes much more difficult in case of optimizing DRNN, because the initial condition of the hidden units of the DRNN is represented by $\{x_t^i(\theta) | i \notin \mathcal{V}\}$, which is continuous mapping on a time interval $\theta \in [-\tau, 0]$. Optimization of such continuous mapping is quite difficult. Hence, we should avoid using hidden units in the learning of the DRNN.

For an ORNN to learn complex high-dimensional dynamics, it is indispensable to use hidden units, because increasing a number of the hidden units is the only way to increase the dimension of the ORNN. In contrast, in order to enhance a capability of DRNN to learn complex high-dimensional dynamics, we can increase a number of the delay units, because the delays make the system infinite dimensional. It is therefore advised that, for the learning of the DRNN, using the time delays is much more efficient and practical than using the hidden units.

3. Numerical Experiments

This section presents experimental studies of applying the learning algorithm developed in Section 2 for the learning of complex nonlinear dynamics. As teacher signals, chaotic signals generated from the Rössler equations and the Mackey-Glass equation are exploited. By comparing the learning capability of the DRNN with those of other neural networks, advantages as well as disadvantages of the DRNN are studied.

3. 1 Network Model

In the learning of complex nonlinear dynamics, the following form of DRNN is utilized:

$$\begin{aligned}\frac{d}{dt}x(t) &= F(\Omega, x(t), x(t-\tau_1), \ldots, x(t-\tau_d)) \\ &= \sum_{k=1}^{h} \omega_k^{(1)} \cdot G(\omega_k^{(2)} \cdot x(t) + \sum_{j=1}^{d} \omega_{kj}^{(3)} \cdot x(t-\tau_j) + \omega_k^{(4)}). \qquad (10)\end{aligned}$$

This network has a single dynamical unit $x(t)$, multiple delta function-type delays $x(t-\tau_1)$,. . .,$x(t-\tau_d)$, and weight parameters $\Omega = \{\omega_k^{(1)}, \omega_k^{(2)}, \omega_{kj}^{(3)}, \omega_k^{(4)}\} \in R^K$. For the single visible unit $x(t)$, a teacher signal $\{\xi(t) | t \in [-\tau, T]\}$ $(\tau = \max_{j=1,\ldots,d}\tau_j)$ is given. As discussed in the previous section, no hidden units are used for the DRNN. Note that the function $F : R^K \times R^d \to R^1$ is based on a multi-layer-perceptron (MLP) that has d units in the input layer, h units in the middle layer, and one unit in the output layer. This type of neural network is used in the present study because:

(a) As is show in Appendix C, DRNN of Eq. (10) can be transformed to the Kleinfeld model (1). Hence, the learning capability of this network can be discussed as that of a standard DRNN.

(b) There is a variety of nonlinear dynamics described by the following form of a delay-differential equation:

$$\frac{d}{dt}\xi(t) = \tilde{F}(\xi(t), \xi(t-\tau_1), \ldots, \xi(t-\tau_d)) \qquad (11)$$

such as lasers (Ikeda & Matsumoto, 1987; Aida & Davis, 1992), a blood pressure model (Mackey & Glass, 1977), a growth model of a single species (Hale, 1977), and delay coordinate embedding of nonlinear dynamics (Takens, 1981). Such dynamics $\tilde{F}$ can be well approximated by MLP F, because the MLP has a universal approximation capability (Funahashi, 1989; Hornik, 1989; Cybenko, 1989).

3. 2 Practical Setting of Algorithm

In the studies of ORNN, learning algorithms often become unstable when numerical integration of the network equation gives rise to a divergence solution, *i.e.*, $x \rightarrow \infty$ as $t \rightarrow \infty$ (Doya, 1991). Similar problems may also happen in the learning of the DRNN. In order to avoid such numerical instability problems, which are typically caused by integration of the network dynamics for a long time interval T, the cost function of Eq. (3) is redefined for a teacher signal $\xi(t)$ divided into S-windows as

$$E(\Omega) = \sum_{n=1}^{S} \int_{(n-1)\cdot T/S}^{n\cdot T/S} \frac{1}{2}\cdot\{x(t) - \xi(t)\}^2 \ dt. \qquad (12)$$

At every starting point of the windowed interval, $t = 0, T/S, 2\cdot T/S, \ldots, (S-1)\cdot T/S$, the initial condition of the DRNN (10) is reset by using the teacher signal as $x_t(\theta) = \xi(t-\theta) \quad (\theta \in [-\tau, 0])$. The modified cost function (12) is then minimized by the quasi-Newton method

$$\Omega^{\mathrm{new}} = \Omega^{\mathrm{old}} - H(\Omega^{\mathrm{old}})\cdot\nabla E(\Omega^{\mathrm{old}}), \qquad (13)$$

where ∇E and H stand for a gradient vector and an approximate [*] of the inverse Hessian of E, respectively. The first derivatives $\partial E/\partial \omega$ are computed by using the TDRL algorithm of subsection 2. 4 and every differential equation is integrated by the Euler method with a small enough integration step.

[*] There is a variety of update formulas for approximating a series of H using E and ∇E. In our numerical experiments, the Broyden-Fletcher-Goldfarb-Shanno formula with Luenberger's self-scaling formula (Luenberger, 1973) is exploited.

The Euler's integration algorithm for the RFDE has been described in detail by Farmer (1982). The termination condition of the learning process is set as $\max_\omega |\partial E/\partial \omega| < 10^{-4}$.

3. 3 Delay Constant Parameters

In the learning of the DRNN of Eq. (10), we should carefully choose the delay parameters $\tau_1, \ldots, \tau_d$, because the learning capability strongly depends upon the delay parameters. In principle, it is possible to consider the delay parameters as learning parameters and adjust them by the learning algorithm. To our experience, however, the learning process of the delay parameters is quite sensitive to initial setting of the delay parameters. In order not to complicate the learning process of the DRNN, in this study, we fix the delay parameters prior to the learning. The delay parameters are chosen according to the following criteria.

(a) In case of learning a teacher signal $\xi(t)$ generated from the delay-differential equation of the form $\dot{\xi}(t) = \tilde{F}(\xi(t), \xi(t-\tau_1), \ldots, \xi(t-\tau_d))$, it is best to utilize the corresponding delay parameters $\tau_1, \ldots, \tau_d$ for the DRNN of Eq. (10). In chaotic data analysis, methods have been developed for identifying the time delays hidden behind the data signal $\xi(t)$. A maximal correlation function technique (Voss & Kurths, 1997) provides a reliable way to detect multiple time delays in the data. Using this technique, optimal delay parameters are determined (see Appendix D for details of the technique).

(b) In case that the signal $\xi(t)$ is not generated from a delay-differential equation as (a), the maximal correlation function technique fails to detect optimal time delays. Then, we use a delay-coordinate embedding technique (Takens, 1981; Sauer, York, & Casdagli, 1991). In this technique, the signal $\xi(t)$ is transformed to a $(d+1)$-dimensional delay coordinate space $\{\xi(t), \xi(t-\tau_1), \ldots, \xi(t-\tau_d)\}$. It has been proven that qualitative dynamics of the original system that generates the signal can be reconstructed in the delay coordinate space if the dimension $d+1$ is large enough. This implies that there exits an implicit dynamics of the form $\dot{\xi}(t) = \tilde{F}(\xi(t), \xi(t-\tau_1), \ldots, \xi(t-\tau_d))$. We try to approximate this dynamics $\tilde{F}$ by using the MLP F. The embedding dimension $d+1$ can be determined by using a false nearest neighbor algorithm (Abarbanel, 1996). About the delay parameters, it is mathematically guaranteed that for a generic choice of delay parameters the delay coordinate transformation provides an embedding (Takens, 1981; Sauer, York, & Casdagli, 1991). There are, however, empirical methods for choosing good delay parameters that work efficiently for analyzing real data (Abarbanel, 1996; Kantz & Schreiber, 1997). Among such techniques, we use an auto-correlation function technique that is to choose such delay τ that first intersects the zero line of the auto-correlation function of the signal $\xi(t)$.

3. 4 Mackey-Glass Equation

As a first learning example, a chaotic signal generated from the following Mackey-Glass equation (Mackey & Glass, 1977) is utilized:

$$\frac{d}{dt}\xi(t) = 0.2\cdot\frac{\xi(t-17)}{1+\xi^{10}(t-17)} - 0.1\cdot\xi(t). \tag{14}$$

According to the Lyapunov spectrum analysis (Farmer, 1982), this system has a first positive Lyapunov exponent of $\lambda_1 = 0.0052$ with the Lyapunov dimension (Kaplan & York, 1987) of $D_L = 2.14$.

Using the Mackey-Glass data $\{\xi(t)|0 < t < 200\}$ as a teacher signal, let us apply the learning algorithm to the DRNN (10). First, we identify the delay parameter τ by the maximal correlation function technique (see Appendix D for details). Figure 2 shows the maximal correlation function $R_{\max}(\tau)$ computed for the teacher signal $\{\xi(t)\}$ with a time delay varied as $\tau\in[4, 20]$. We see that the maximal correlation of $R_{\max}\approx 1$ is realized at $\tau = 17$. The value of $R_{\max}\approx 1$ gives the strongest evidence for an existence of time delayed feedback in the teacher signal. We therefore use this time delay for the learning of the teacher signal. Namely, for DRNN of Eq. (10) that has one delayed feedback ($d = 1$) with $\tau = 17$, the TDRL algorithm is applied. The number of the middle units is set as $h = 5$ and the time interval $T = 200$ is divided by $S = 100$ in the cost function (12).

Figure 3 shows a change in the cost function E with increasing the learning steps. After 1068-iterative learning, the DRNN achieved qualitatively similar chaotic dynamics as the original (see Figs. 4 (a) and (b)). According to the Lyapunov spectrum analysis, the network has a first positive Lyapunov exponent of $\lambda_1 = 0.0057$ with the Lyapunov dimension of $D_L = 2.15$.

These values are quite similar to those of the original Mackey-Glass equation. Hence, not only visually but also quantitatively similar dynamics has been achieved by the DRNN.

Let us consider the case when learning the Mackey-Glass equation by ORNN. In order to learn the Mackey-Glass equation that has the Lyapunov dimension of $D_L = 2.14$, one visible unit and at least two hidden units are required for the ORNN. More hidden units are required as the time delay of the Mackey-Glass equation is increased, since larger time delay gives rise to much higher-dimensional complex dynamics in the Mackey-Glass system. The present experiment therefore demonstrates that the DRNN can avoid introducing such a large number of hidden units by making good use of time-delayed feedback.

3. 5 Rössler Equation

As a second example, a chaotic signal generated from the following Rössler equations (Rössler, 1979) is studied:

$$\begin{aligned} \frac{d\xi_1}{dt} &= \xi_2 - \xi_3, \\ \frac{d\xi_2}{dt} &= \xi_1 + 0.4 \cdot \xi_2, \\ \frac{d\xi_3}{dt} &= 2 - (4 - \xi_1) \cdot \xi_3. \end{aligned} \tag{15}$$

Among the three dynamical variables, the first variable $\xi_1(t)$ is used as the teacher signal. According to the Lyapunov spectrum analysis (Shimada & Nagashima, 1979), this Rössler system has a first positive Lyapunov exponent of $\lambda_1 = 0.089$ with the Lyapunov dimension of $D_L = 2.022$.

Using the Rössler data $\{\xi_1(t)|0 < t < 80\}$ as a teacher signal, the learning algorithm is applied to the DRNN (10). Since the data are not generated from a delay-differential equation, the maximal correlation function technique fails to identify an optimal time delay parameter for the teacher signal. In stead of the maximal correlation function technique, by using the auto-correlation function technique with the false nearest neighbor algorithm, two time delays ($d = 2$) are set as $(\tau_1, \tau_2) = (0.8, 1.6)$ for the DRNN (10) [†] . The number of the middle units is set as $h = 8$ and the time interval $T = 80$ is divided by $S = 100$ in the cost function (12).

After 2250-iterative learning, the DRNN achieved qualitatively similar chaotic dynamics as the original (see Figs. 5 (a) and (b)). According to the Lyapunov spectrum analysis, the network has a first positive Lyapunov exponent of $\lambda_1 = 0.072$ with the Lyapunov dimension of $D_L = 2.035$. These values are again quite similar to those of the original Rössler equations.

3. 6 Robustness Against Noise

Let us study robustness of the learning of the DRNN against observational noise. By adding *Gaussian* noise $\{\nu_i\}$ ($E[\nu_i] = 0$, $E[\nu_i^2] = \sigma^2$) to the teacher signal as $\{\xi(i{\cdot}\Delta t) + \nu_i\}$ (Δt: sampling time), we investigate influence of the noise on the accuracy of learning the original signal $\{\xi(i{\cdot}\Delta t)\}$ by DRNN. The learning accuracy is measured by a nonlinear prediction error as follows.

[†] By the false nearest neighbor algorithm (Abarbanel, 1996), the minimum embedding dimension required for the teacher signal was found to be three. Two time delayed feedbacks are utilized in order to realize 3-dimensional delay coordinate reconstruction of the teacher signal.

First, the teacher signal $\{\xi(t)\}$ is divided into the first and the second halves. From the first half data with additive *Gaussian* noise, DRNN is optimized by the learning algorithm. Then, for the second half data with no noise added, nonlinear prediction is carried out by using the optimized DRNN. In the nonlinear prediction, for a given initial state $x_t(\theta) = \xi(t+\theta)$ with $\theta\in[-\tau, 0]$, the κ-time interval future state $\xi(t+\kappa)$ is predicted as $x(t+\kappa)$ $= \Phi^{\kappa}(x_t)$, where $\Phi^{\kappa} : C([-\tau, 0], R^1) \rightarrow R^1$ stands for a time evolution of the DRNN with κ-time interval. The nonlinear prediction error NPE is finally computed as the following normalized root-mean-square error

$$NPE = \frac{E[\{\xi(t+\kappa) - x(t+\kappa)\}^2]^{\frac{1}{2}}}{E[\{\xi(t) - E[\xi(t)]\}^2]^{\frac{1}{2}}}, \tag{16}$$

where $E[\cdot]$ stands for an averaging over time series.

As the teacher signals, chaotic solutions of the Rössler equations and the Mackey-Glass equation introduced in subsections 3. 4 and 3. 5 are utilized, where conditions of the learning algorithm are set to be the same as those of the previous subsections. In order to compare with the DRNN, nonlinear prediction errors are also computed for two types of other neural networks: (1) ORNNs and (2) TDNNs (see Appendix E for detailed description of the neural models).

Figures 6 show dependence of the nonlinear prediction errors of DRNN (solid line with squares), ORNN (dotted line with crosses), and TDNN (dashed line with circles) on the noise level σ. While the Mackey-Glass data are predicted with a prediction interval of $\kappa = 4.1$ in Fig. 6 (a), the Rössler data are predicted with a prediction interval of $\kappa = 5.2$ in Fig. 6 (b). Each prediction error is drawn by averaging the NPE over 10 realizations of the neural networks obtained by the learning algorithm started from different initial conditions of the weight parameters. As the noise level σ is increased, the prediction error also increases for all three neural networks. Compared to ORNN and TDNN, DRNN shows better prediction accuracy especially for the learning of the Mackey-Glass equation. This might be due to the following reasons.

(a) For the learning of a signal $\xi(t)$ generated from the Mackey-Glass delay-differential equation, DRNN that belong to the same equational class should provide better prediction models than other neural networks.

(b) TDNN can be considered as the Euler's discretization of the continuous-time DRNN. As the Euler's discretization step Δt is increased, nonlinearity of the time evolution of the teacher signal, that is, $\xi(t) \rightarrow \xi(t+\Delta t)$, gets much stronger. Learning the time evolution with such stronger nonlinearity by TDNN should get worse for a large discretization step Δt.

(c) For the learning of the Rössler dynamics, DRNN and ORNN give rise to similar prediction curves especially in a small noise regime. This implies that both networks are capable of learning dynamics generated from ordinary differential equations with good accuracy. DRNN is slightly better than ORNN, because more dynamical information with continuous mapping $\xi_t([-\tau, 0])$ is given as an initial condition to DRNN.

(d) Increase in prediction errors of TDNN with increasing the noise intensity is not so significant as that of ORNN. This might be due to the fact that, in the learning of the ORNN, influence of noise is amplified by a time discretization procedure of $\dot{\xi}(t){\approx}\frac{\xi(t+\Delta t)-\xi(t)}{\Delta t}+\frac{\nu(t+\Delta t)-\nu(t)}{\Delta t}$. This noise amplification makes the learning of the ORNN much more sensitive to noise. The DRNN, on the other hand, is robust against such amplified noise.

3. 7 Effect of Noise on Initial Condition

One of the drawbacks of utilizing DRNN for learning complex signals is the continuum of data required to set the initial condition $x_0([-\tau, 0])$. If noisy signals are input to the initial condition, dynamical behavior of the DRNN may become very unstable. In this subsection, we examine an effect of noise in the initial condition of the DRNN by computing the nonlinear prediction error as follows.

First, teacher signal $\{\xi(t)\}$ is divided into the first and the second halves. From the first half data with no noise added, DRNN (10) is optimized by the learning algorithm. Then, for the second half data, nonlinear prediction is carried out by using the optimized DRNN. As discussed in subsection 2. 5, in numerical experiments, the initial condition of the DRNN is represented by J_1 sets of dynamical variables $\{x_t(-i{\cdot}\Delta t)|i=0,1,\ldots,J_1{-}1\}$ divided with a time interval of $\Delta t=\tau/(J_1{-}1)$. In the nonlinear prediction, the initial condition is set using a noisy signal as $x_t(-i{\cdot}\Delta t)=\xi(t-i{\cdot}\Delta t)+\nu_i \;\; (i=0,1,\ldots,J_1-1)$, where $\{\nu_i\}$ is *Gaussian* noise with $E[\nu_i]=0$ and $E[\nu_i^2]=\sigma^2$. From the noisy initial condition, the κ-time interval future state of the noise-free signal $\xi(t+\kappa)$ is predicted as $\widetilde{\xi}(t+\kappa)=\Phi^{\kappa}(\xi_t)$. The nonlinear prediction error is finally computed by Eq. (16).

Except for the noisy initial condition explained above, experimental conditions are set to be the same as those of subsection 3. 6. Namely, chaotic solutions of the Rössler equations and the Mackey-Glass equation are used as the teacher signals and the prediction accuracy of DRNN is compared with those of ORNN and TDNN.

Figures 7 show dependence of the nonlinear prediction errors of DRNN (solid line with squares), ORNN (dotted line with crosses), and TDNN (dashed line with circles) on the noise level σ. As the noise level σ is increased, the prediction error also increases for all three neural networks. Compared to ORNN and TDNN, DRNN gives the lowest prediction errors for both chaotic signals. In particular, excellence of the DRNN is clearly shown in the nonlinear prediction of the Mackey-Glass data. This experiment therefore implies that the effect of noise on the initial condition of the DRNN is not at all in a level of destabilizing the network behavior.

4. Conclusions and Discussions

In this paper, the learning algorithms and the learning capability of the DRNN have been investigated based on the comparative studies with ORNN and TDNN. Advantages ((A1)-(A4)) as well as disadvantages ((D1)-(D3)) of the DRNN are summarized as follows.

(A1) When the TDRL algorithm is utilized for DRNN, the computational cost can be largely reduced from the RTRL algorithm. In the TDRL algorithm, the memory requirement for the DRNN is the same as that for the ORNN.

(A2) For learning chaotic signals, DRNN was better than both ORNN and TDNN especially when learning a teacher signal generated from a delay-differential equation.

(A3) The learning process of the DRNN was robust against observational noise in the same level as that of the TDNN. DRNN was also robust against additive noise on the initial condition.

(A4) Instead of using hidden units that require a delicate optimization process of the initial condition, the learning capability of the DRNN can be controlled by using time delays.

(D1) The state space of the DRNN is represented by a continuous mapping of finite time interval to all dynamical variables. This makes numerical integration of the dynamical equations and the learning equations very costly.

(D2) In the RTRL algorithm, much more computational cost with larger memory is required for DRNN than ORNN.

(D3) It is not always easy to utilize DRNN, because a continuum of data are required to set the initial condition.

According to the summary above, we may conclude that, to learn complex chaotic signals that might be contaminated with observational noise, DRNN provides a good network model when the teacher signal is generated from a delay-differential equation. This conclusion might sound as a matter of course, because it should be easier to learn a delay-differential equation by using a DRNN that belongs to the same equational class. There is, however, a variety of high-dimensional complex signals in nature that are generated from delay-differential equations (Hale, 1977). Modelling such signals by ORNN or TDNN does not always provide a bright strategy to deal with the problem, because some of the important dynamical information such as the time delayed feedback structure are lost by such network models. The DRNN that gives a more precise model should be utilized for learning the high-dimensional complex signals.

In the present experiments, learning capability of recurrent neural networks with only delta-function type delays has been investigated. As is described in Section 2, various types of delays can be considered for neural network models. It is an interesting future problem to investigate learning capability of DRNN with other types of time delays.

Finally, delayed feedback systems can be found also in many engineering systems such as the passive optical resonator (Ikeda & Matsumoto, 1987). This laser system is known to exhibit rich dynamical phenomena such as higher-harmonic bifurcations that give rise to multi-stability of infinitely many periodic attractors. This multi-stability has been actually used as a memory device for complex information coding (Aida & Davis, 1992). This implies that the neural network with time-delays may also give rise to this type of multi-stability and might be capable of learning and embedding many attractors in the network. Learning multiple dynamics may also provide us with an interesting new application of the DRNN.

REFERENCES

Abarbanel, H. D. I. (1996). *Analysis of observed chaotic data.* New York: Springer.

Aida, T., & Davis, P. (1992). Oscillation modes of laser diode pumped hybrid bistable system with large delay and application to dynamical memory. *IEEE Journal of Quantum Electronics*, **28**, 686–699.

Anderson, C. H., & van Essen, D. C. (1987). Shifter circuits. *Proceedings of The National Academy of Science, USA*, **84**, 6297–6301.

Baldi, P., & Atiya, A. F. (1994). How delays affect neural dynamics and learning. *IEEE Transactions on Neural Networks*, **5**, 612–621.

Cao, Y. J., & Wu, Q. H. (1996). A note on stability of analog neural networks with time delays. *IEEE Transactions on Neural Networks*, **7**, 1533–1535.

Cohen, B., Saad, D., & Marom, E (1997). Efficient training of recurrent neural network with time delays. *Neural Networks*, **10-1**, 51–59.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, **2**, 303–314.

Doya, K., & Yoshizawa, S. (1989). Adaptive neural oscillator using continuous-time back-propagation algorithm. *Neural Networks*, **2**, 375–385.

Doya, K. (1991). A study of learning algorithms for continuous-time recurrent neural networks. *Ph.D Thesis*, University of Tokyo.

Doya, K., & Yoshizawa, S. (1991). Neural network model of temporal pattern memory. *Systems and Computers in Japan*, **22**, 61–69.

Doya, K. (1993). Basics of neural networks. In Aihara, K. (Ed.), *Neuro, Fuzzy, and Chaos - Introduction to Next Generation Analog Computing* (pp.13–39). Tokyo: Ohmsha. (in Japanese).

Farmer, J. D. (1982). Chaotic attractors of an infinite-dimensional dynamical system. *Physica D*, **4**, 366–393.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, 183–192.

Funahashi, K., & Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, **6**, 801–806.

Gouhara, K., Takase, H., Uchikawa, Y., & Iwata, K. (1992). Attractor learning of recurrent neural networks. *Artificial Neural Networks*, **2**, 371–374.

Hale, J. K. (1977). *Theory of functional differential equations*. New York: Springer.

Herz, A., Sulzer, B., Kühn, R., & van Hemmen, J. L. (1989). Hebbian learning reconsidered. *Biological Cybernetics*, **60**, 457–467.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141–152.

Hopfield, J. J., & Tank, D. W. (1989). Neural architecture and biophysics for sequence recognition. In Byrne, J. H., & Berry, W. O. (Eds.), *Neural Models of Plasticity* (pp.363–377). San Diego: Academic Press.

Hornik, K. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366.

Ikeda, K., & Matsumoto, K. (1987). High-dimensional chaotic behavior in systems with time delayed feedback. *Physica D*, **29**, 223–235.

Joy, M. (2000). Results concerning the absolute stability of delayed neural networks. *Neural Networks*, **13-6**, 613–616.

Kantz, H, & Schreiber, T. (1997). *Nonlinear time series analysis.* Cambridge: Cambridge University Press.

Kaplan, J. L., & York, J. A. (1987). Chaotic behaviour of multi-dimensional difference equations in functional differential equations and approximations of fixed points. In *Lecture Notes in Mathematics*, Vol. 730 (pp.204–227). Berlin: Springer.

Kleinfeld, D. (1986). Sequential state generation by model neural networks. *Proceedings of The National Academy of Science, USA*, **83**, 9469–9473.

Lapedes, A., & Farber, R. (1987). Nonlinear signal processing using neural networks. *Technical Report of Los Alamos National Laboratory*, LA-UR-87-2662.

Liao, X., Chen, G., & Sanchez, E. N. (2002). Delay-dependent exponential stability analysis of delayed neural networks: an LMI approach. *Neural*

Networks, **15-7**, 855–866.

Lu, H. (2000). On stability of nonlinear continuous-time neural networks with delays. *Neural Networks*, **13-10**, 1135–1143.

Luenberger, D. G. (1973). *Linear and nonlinear programming* Addison-Wesley.

Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, **197**, 287–295.

Marcus, C. M., & Westervelt, R. M. (1989). Stability of analog neural networks with delay. *Physical Review A*, **39-2**, 347–359.

Mozer, M. C. (1989). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, **3**, 349–381.

Pakdaman, K., & Malta, C. P. (1998). A note on convergence under dynamical threshold with delays. *IEEE Transactions on Neural Networks*, **9-1**, 231–233.

Pearlmuter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, **1**, 263–269.

Peng, J., Qiao, H., & Xu, Z. (2002). A new approach to stability of neural networks with time-varying delays. *Neural Networks*, **15-1**, 95–103.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, **59**, 2229–2232.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Rössler, O. E. (1979). Continuous chaos. *Annals of New York Academy of Science*, **31**, 376–392.

Sato, M. (1990). A Learning algorithm to teach spatio-temporal patterns to recurrent neural networks. *Biological Cybernetics*, **62**, 259–263.

Sato, M., Joe, K., & Hirahara, T. (1990). APOLONN brings us to the real world. *Proceedings of the International Joint Conference on Neural Networks*, **1** 581–587.

Sato, M., Murakami, Y., & Joe, K. (1990). Learning chaotic dynamics by recurrent neural networks. *Proceeding of The International Conference on Fuzzy Logic and Neural Nets, Iizuka*, 601–604.

Shimada, I., & Nagashima, T. (1979). A numerical approach to ergodic problem of dissipative dynamical systems. *Progressive Theory of Physics*, **61**, 1605–1616.

Sompolinsky, H., & Kanter, I. (1986). Temporal association in asymmetric neural networks. *Physical Review Letters*, **57**, 2861–2864.

Tank, D. W., & Hopfield, J. J. (1987). Neural computation by concentrating information in time. *Proceedings of The National Academy of Science, USA*, **84**, 1896–1900.

Tokuda, I. (1998). Approximation capability of neural networks with time-delayed feedbacks. *Technical Report of IEICE*, NC97-75, NLP97-123, 1–5.

Tokuda, I., & Aihara, K. (1997). Back-propagation learning of infinite-dimensional dynamical systems. *Technical Report of IEICE*, NC96-92, NLP96-138, 109–114.

Tokuda, I., Hirai, Y., & Tokunaga, R. (1993). Back-propagation learning of an infinite-dimensional dynamical system. *Proceedings of the International Joint Conference on Neural Networks, Nagoya*, **3**, 2271–2275.

Tokuda, I., Tokunaga, R., & Aihara, K. (1996). A simple geometrical structure underlying speech signals of the Japanese vowel /a/. *International Journal of Bifurcation and Chaos*, **6**, 149–160.

Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, **1**, 270–280.

Sauer, T., York, J. A., & Casdagli, M. (1991). Embedology. *Journal of Statistical Physics*, **65(3)**, 579–616.

Takens, F. (1981). Detecting strange attractors in turbulence. In *Lecture Notes in Mathematics*, Vol. 898 (pp.366–381). Berlin: Springer.

Voss, H., & Kurths, J. (1997). Reconstruction of non-linear time delay models from data by the use of optimal transformations. *Physics Letter A*, **234**, 336–344.

Zhang, J., & Jin, X. (2000). Global stability analysis in delayed Hopfield neural network models. *Neural Networks*, **13-7**, 745–753.

ABBREBATIONS

DRNN: Delayed Recurrent Neural Network

ORNN: Ordinary differential equation model of Recurrent Neural Network

ODE: Ordinary Differential Equation

RFDE: Retarded Functional Differential Equation

RTRL: Real Time Recurrent Learning

TDNN: Time-Delay Neural Network

TDRL: Time Dependent Recurrent Learning

MLP: Multi-Layer Perceptron

Appendix A

In this appendix, we show how the first derivatives $\partial E/\partial\omega$ are computed by the TDRL algorithm using the Lagrange multipliers $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$.

First, we rewrite the cost function of Eq. (3) as

$$\begin{aligned} L(\Omega) = \int_0^T \Big[\sum_{i=1}^{N} \Big\{ \frac{1}{2}\cdot\delta_{i\in\mathcal{V}}\cdot\{x^i(t) - \xi_i(t)\}^2 \\ - \lambda_i(t)\cdot\{\dot{x}^i(t) - f_i(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega)\} \Big\} \Big] \, dt. \end{aligned} \quad \text{(a1)}$$

Then, the first derivatives $\partial L/\partial\omega$ are calculated as

$$\begin{aligned} \frac{\partial L}{\partial\omega} = \int_0^T \quad \Big[\quad & \sum_{i=1}^{N} \Big\{ \delta_{i\in\mathcal{V}}\cdot\{x^i(t) - \xi_i(t)\}\cdot\frac{\partial x^i}{\partial\omega}(t) \\ + \quad & \lambda_i(t)\cdot\sum_{j=1}^{N} \int_{-\tau}^{0} ds \, \frac{\partial f_i}{\partial x_t^j(s)}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega)\cdot\frac{\partial x^j}{\partial\omega}(t+s) \\ - \quad & \lambda_i(t)\cdot\frac{d}{dt}(\frac{\partial x^i}{\partial\omega}) \; + \; \lambda_i(t)\cdot\frac{\partial f_i}{\partial\omega}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega) \\ - \quad & \frac{\lambda_i}{\partial\omega}\cdot\{\dot{x}^i(t) - f_i(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega)\} \Big\} \Big] \, dt. \end{aligned} \quad \text{(a2)}$$

By the network dynamics of Eq. (2), the final term of Eq. (a2) vanishes. Since the second term of the Eq. (a2) can be written by the transformation $t' = t + s$ as

$$\begin{aligned} & \int_0^T \Big[\sum_{i=1}^{N} \Big\{ \lambda_i(t)\cdot\sum_{j=1}^{N} \int_{-\tau}^{0} ds \, \frac{\partial f_i}{\partial x_t^j(s)}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega)\cdot\frac{\partial x^j}{\partial\omega}(t+s) \Big\} \Big] \, dt \\ = \; & \int_0^T \Big[\sum_{i=1}^{N} \Big\{ \frac{\partial x^i}{\partial\omega}(t')\cdot\sum_{j=1}^{N} \int_{-\tau}^{0} ds \, \delta_{t'\in[0,T+s]}\cdot\lambda_j(t'-s)\cdot \\ & \frac{\partial f_j}{\partial x_{t'-s}^i(s)}(\boldsymbol{x}_{t'-s}, \boldsymbol{z}(t'-s), \Omega) \Big\} \Big] \, dt', \end{aligned} \quad \text{(a3)}$$

the first derivatives $\partial L/\partial\omega$ become

$$\begin{aligned}\frac{\partial L}{\partial \omega} = \int_0^T \quad \Big[\quad & \sum_{i=1}^{N} \Big\{ \frac{\partial x^i}{\partial \omega}(t) \cdot \{ \ \delta_{i \in \mathcal{V}} \cdot (x^i(t) - \xi_i(t)) \\ + \quad & \sum_{j=1}^{N} \int_{-\tau}^{0} ds \ \delta_{t \in [0,T+s]} \cdot \lambda_j(t-s) \cdot \frac{\partial f_j}{\partial x^i_{t-s}(s)}(\boldsymbol{x}_{t-s}, \boldsymbol{z}(t-s), \Omega) \ \} \\ - \quad & \lambda_i(t) \cdot \frac{d}{dt}(\frac{\partial x^i}{\partial \omega}) \ + \ \lambda_i(t) \cdot \frac{\partial f_i}{\partial \omega}(\boldsymbol{x}_t, \boldsymbol{z}(t), \Omega) \Big\} \Big] \ dt. \end{aligned} \tag{a4}$$

Suppose that the Lagrange multipliers $\boldsymbol{\lambda}(t)$ satisfy Eqs. (7) (see subsection 2. 4) with a boundary condition $\boldsymbol{\lambda}(T) = 0$. Then, the first, the second, and the third terms of Eq. (a4) become

$$\begin{aligned} & -\int_0^T \sum_{i=1}^{N} \{ \ \frac{d}{dt}\lambda_i(t) \cdot \frac{\partial x^i}{\partial \omega}(t) \ + \ \lambda_i(t) \cdot \frac{d}{dt} \ \frac{\partial x^i}{\partial \omega}(t) \ \} \ dt \\ = \quad & -\sum_{i=1}^{N} \{ \ \lambda_i(0) \cdot \frac{\partial x^i}{\partial \omega}(0) + \lambda_i(T) \cdot \frac{\partial x^i}{\partial \omega}(T) \ \}. \end{aligned} \tag{a5}$$

Since $\partial x^i / \partial \omega(0) = 0$ and $\boldsymbol{\lambda}(T) = 0$, the above terms also vanish. Therefore, the first derivatives $\partial E / \partial \omega$ can be calculated by integrating the final term of Eq. (a4) (see Eq. (8)).

Appendix B

In subsections 2. 3 and 2. 4, learning algorithms have been introduced for DRNN using a general form of RFDE (2). To be more practical, this appendix provides explicit equational forms for (I) the RTRL algorithm and (II) the TDRL algorithm for the Kleinfeld model (1).

(I) The main part of the RTRL algorithm is the computation of the first variational equations (6). For the Kleinfeld model (1), the variational equations can be written in explicit mathematical forms as

$$\begin{aligned}\frac{d}{dt}\Big(\frac{\partial x^i}{\partial W_{kl}}\Big)(t) &= \delta_{i=k}\cdot V_l(t) - \frac{1}{R}\cdot\frac{\partial x^i}{\partial W_{kl}}(t) + \sum_{j=1}^{N} W_{ij}\cdot G'(x^j(t))\cdot\frac{\partial x^j}{\partial W_{kl}}(t) \\ &+ \sum_{j=1}^{N} D_{ij}\cdot\int_{-\tau}^{0} G'(x^j(t+s))\cdot\frac{\partial x^j}{\partial W_{kl}}(t+s)\cdot\mathcal{D}_j(s)\,ds,\end{aligned} \tag{b1}$$

$$\begin{aligned}\frac{d}{dt}\Big(\frac{\partial x^i}{\partial D_{kl}}\Big)(t) &= \delta_{i=k}\cdot V_{\mathcal{D}_l}(t) - \frac{1}{R}\cdot\frac{\partial x^i}{\partial D_{kl}}(t) + \sum_{j=1}^{N} W_{ij}\cdot G'(x^j(t))\cdot\frac{\partial x^j}{\partial D_{kl}}(t) \\ &+ \sum_{j=1}^{N} D_{ij}\cdot\int_{-\tau}^{0} G'(x^j(t+s))\cdot\frac{\partial x^j}{\partial D_{kl}}(t+s)\cdot\mathcal{D}_j(s)\,ds,\end{aligned} \tag{b2}$$

$$\begin{aligned}\frac{d}{dt}\Big(\frac{\partial x^i}{\partial I_{kn}}\Big)(t) &= \delta_{i=k}\cdot z_l(t) - \frac{1}{R}\cdot\frac{\partial x^i}{\partial I_{kn}}(t) + \sum_{j=1}^{N} W_{ij}\cdot G'(x^j(t))\cdot\frac{\partial x^j}{\partial I_{kn}}(t) \\ &+ \sum_{j=1}^{N} D_{ij}\cdot\int_{-\tau}^{0} G'(x^j(t+s))\cdot\frac{\partial x^j}{\partial I_{kn}}(t+s)\cdot\mathcal{D}_j(s)\,ds,\end{aligned} \tag{b3}$$

for weight matrices $\{(W_{kl}, D_{kl}I_{kn}) : k, l = 1, \ldots, N, n = 1, \ldots, m\}$.

(II) In the TDRL algorithm, Lagrange multipliers $\boldsymbol{\lambda}(t)$ are calculated by the backward equations (7). For the Kleinfeld model (1), the backward equations can be written as

$$\begin{aligned}\frac{d}{dt}\lambda_i(t) &= \delta_{i\in\mathcal{V}}\cdot\{\xi_i(t) - x^i(t)\} + \frac{1}{R}\cdot\lambda_i(t) - G'(x_i(t))\cdot\sum_{j=1}^{N}\{W_{ji}\cdot\lambda_j(t) + \\ &D_{ji}\cdot\int_{-\tau}^{0}\delta_{t\in[0,T+s]}\cdot\lambda_j(t-s)\cdot\mathcal{D}_i(s)\,ds\}.\end{aligned} \tag{b4}$$

Then, the first derivatives are integrated as

$$\frac{\partial L}{\partial W_{kl}} = \int_0^T \lambda_k(t) \cdot V_l(t)\, dt, \tag{b5}$$

$$\frac{\partial L}{\partial D_{kl}} = \int_0^T \lambda_k(t) \cdot V_{\mathcal{D}_l}(t)\, dt, \tag{b6}$$

$$\frac{\partial L}{\partial I_{kn}} = \int_0^T dt\, \lambda_k(t) \cdot z_n(t). \tag{b7}$$

Appendix C

This appendix shows how the neural network model of Eq. (10) is transformed to the Kleinfeld model (1). First, using a time constant R, a new term $-x/R$ is added to Eq. (10) as

$$\frac{d}{dt}x(t) = -\frac{1}{R}\cdot x(t) + \sum_{k=1}^{h} \omega_k^{(1)}\cdot G(\omega_k^{(2)}\cdot x(t) + \sum_{j=1}^{d} \omega_{kj}^{(3)}\cdot x(t-\tau_j) + \omega_k^{(4)}). \qquad \text{(c1)}$$

The parameter R is chosen large enough so that the neural network (c1) exhibits qualitatively the same dynamics as the original equation (10). If we set

$$y_k(t) = \omega_k^{(2)}\cdot x(t) + \sum_{j=1}^{d} \omega_{kj}^{(3)}\cdot x(t-\tau_j) + \omega_k^{(4)} \qquad \text{(c2)}$$

for $k = 1, 2, \ldots, h$, then, the network (c1) becomes

$$\begin{aligned} \frac{d}{dt}x(t) &= -\frac{1}{R}\cdot x(t) + \sum_{j=1}^{h} \omega_j^{(1)}\cdot G(y_j(t)), & \text{(c3)} \\ \frac{d}{dt}y_k(t) &= \omega_k^{(2)}\cdot \frac{dx}{dt}(t) + \sum_{j=1}^{d} \omega_{kj}^{(3)}\cdot \frac{dx}{dt}(t-\tau_j) & \\ &= -\frac{1}{R}\cdot y_k(t) + \frac{1}{R}\cdot \omega_k^{(4)} + \sum_{j=1}^{h} \omega_k^{(2)}\cdot \omega_j^{(1)}\cdot G(y_j(t)) & \\ &\quad + \sum_{l=1}^{d}\sum_{j=1}^{h} \omega_{kl}^{(3)}\cdot \omega_j^{(1)}\cdot G(y_j(t-\tau_l)). & \text{(c4)} \end{aligned}$$

Since R is chosen large enough, we may drop the $(\omega_k^{(4)}/R)$ term in Eq. (c4). By setting

$$z^1 = x,\ z^2 = y_1,\ z^3 = y_2, \cdots,\ z^{h+1} = y_h, \qquad \text{(c5)}$$

$$W_{1\,j} = \omega_j^{(1)},\ W_{i\,1} = 0,\ W_{i+1\,j+1} = \omega_i^{(2)}\cdot \omega_j^{(1)}, \qquad \text{(c6)}$$

$$D_{1\,k\,j} = 0,\ D_{i\,k\,1} = 0,\ D_{i+1\,k\,j+1} = \omega_{ki}^{(3)}\cdot \omega_j^{(1)}, \qquad \text{(c7)}$$

for $i, j = 1, 2, \ldots, h$, $k = 1, 2, \ldots, d$, we obtain the following DRNN

$$\frac{d}{dt}z^i(t) = -\frac{1}{R}\cdot z^i(t) + \sum_{j=1}^{h+1} W_{ij}\cdot G(z^j(t)) + \sum_{k=1}^{d}\sum_{j=1}^{h+1} D_{ikj}\cdot G(z^j(t-\tau_k)), \qquad \text{(c8)}$$

which is in the form of the Kleinfeld network (1) with $(h+1)$ neurons, multiple delta function-type delays, and no external input.

Appendix D

This appendix briefly explains a maximal correlation function technique (Voss & Kurths, 1997) for detecting time delays $\tau_1, \ldots, \tau_d$ in time series $\{\xi(t)\}$ generated from the following delay-differential equation:

$$h(\frac{d\xi}{dt}(t)) = f_0(\xi(t)) + \sum_{i=1}^{d} f_i(\xi(t - \tau_i)). \tag{d1}$$

The functions h and $f_0, \ldots, f_d$ are assumed to be continuous.

The main point of this approach is to maximize the correlation coefficient

$$R(X, Y) = \frac{E[X \cdot Y] - E[X] \cdot E[Y]}{\sqrt{E[X^2] \cdot E[Y^2]}} \tag{d2}$$

between two variables

$$X \quad = \quad \Phi(\frac{d\xi}{dt}(t)) \tag{d3}$$

$$Y \quad = \quad \phi_0(\xi(t)) + \sum_{i=1}^{d} \phi_i(\xi(t - \hat{\tau}_i)) \tag{d4}$$

with respect to transformations Φ, $\phi_0, \ldots,$ ϕ_d, which are all one-to-one continuous nonlinear functions.

The alternating conditional expectation (ACE) algorithm provides an iterative technique (see Voss & Kurths, 1997 for technical details) to maximize the correlation $R(X, Y)$ by seeking for a set of optimal transformations Φ^*, $\phi_0^*, \ldots,$ ϕ_d^*. Based on the ACE algorithm, the maximal correlation function

$$R_{\max}(\hat{\tau}_1, \ldots, \hat{\tau}_d) = \max_{\Phi, \phi_0, \ldots, \phi_d} R(X, Y) \tag{d5}$$

can be obtained for a time series $\{\xi(t)\}$. If the time delays $\hat{\tau}_1, \ldots, \hat{\tau}_d$ are set to be the same as those of the original equation (d1), the optimal transformations are expected to coincide with the original functions as $\Phi^* =$

h, $\phi_i^* = f_i$ ($i = 0, \ldots, d$) and the maximal correlation becomes one, ı.e., $R_{\max}(\tau_1, \ldots, \tau_d) = 1$.

Since the time delays are supposed to be unknown in experimental situations, for various settings of time delays $\{\hat{\tau}_1, \ldots, \hat{\tau}_d\}$, we compute the maximal correlation function $R_{\max}(\hat{\tau}_1, \ldots, \hat{\tau}_d)$ and seek for a set of delays $\{\tau_1^*, \ldots, \tau_d^*\}$ that give rise to the maximal correlation of $R_{\max}(\tau_1^*, \ldots, \tau_d^*) \approx 1$. In Fig. 2, the result of applying the maximal correlation function technique to a time series generated from the Mackey-Glass equation is shown.

Appendix E

In subsections 3. 6 and 3. 7, (A) ORNN and (B) TDNN are compared with DRNN. This appendix provides details of the two neural networks.

(A) The ORNN is described as follows

$$\begin{aligned} \frac{d}{dt}x^i(t) &= F(\Omega, x^1(t), x^2(t), \ldots, x^N(t)) \qquad \text{(e1)} \\ &= \sum_{k=1}^{h} \omega_{ik}^{(1)} \cdot G(\sum_{j=1}^{N} \omega_{kj}^{(2)} \cdot x^j(t) + \omega_k^{(3)}). \qquad \text{(e2)} \end{aligned}$$

This network has N dynamical variables $x^1(t)$,. . .,$x^N(t)$ and K weight parameters $\Omega = \{\omega_{ik}^{(1)}, \omega_{kj}^{(2)}, \omega_k^{(3)}\}$ $(\in R^K)$. The function $F : R^K \times R^N \rightarrow R^N$ is based on the MLP that has N units in the input layer, h units in the middle layer, and N units in the output layer. Because of the universal approximation capability of the MLP, this neural network has been widely used to study learning capability of ORNN (Sato, Murakami, & Joe, 1990). In particular, Funahashi and Nakamura (1993) proved that the ORNN of Eq. (e2) is capable of approximating any dynamics generated from ordinary differential equations.

For the learning of a single time series $\{\xi(t)\}$ in subsections 3. 6 and 3. 7, delay-coordinate embedding technique was exploited. Namely, the time series was reconstructed in a delay coordinate space $\{(\xi(t), \xi(t-\tau), \ldots, \xi(t-(N-1)\cdot\tau)\}$, which were used as teacher signals for the dynamical variables $\{x^1(t)$,$x^2(t)$,. . ., $x^N(t)\}$, respectively. Since each unit has a corresponding teacher signal, there were no hidden units. To choose the time lag τ, autocorrelation function technique was used. For the learning of the Mackey-Glass equation (14) and the Rössler equations (15), the number of the dynamical units, the number of the middle units, and the time lag were set as (N, h, τ) $= (3, 10, 0.8)$ and $(3, 10, 7.5)$, respectively. Other learning conditions were set to be the same as those of the DRNN.

(B) The TDNN used in subsections 3. 6 and 3. 7 is described as

$$\begin{aligned} x((i+1)\cdot\Delta t) &= F(\Omega, x(i\cdot\Delta t), x((i-1)\cdot\Delta t), \ldots, x((i-N+1)\cdot\Delta t)) && \text{(e3)} \\ &= \sum_{k=1}^{h} \omega_k^{(1)} \cdot G(\sum_{j=1}^{N} \omega_{kj}^{(2)} \cdot x((i-j+1)\cdot\Delta t) + \omega_k^{(3)}). && \text{(e4)} \end{aligned}$$

This network has K weight parameters $\Omega = \{\omega_k^{(1)}, \omega_{kj}^{(2)}, \omega_k^{(3)}\}$ $(\in R^K)$ and N dynamical variables $x(i\cdot\Delta t)$, $x((i-1)\cdot\Delta t)$,..., $x((i-N+1)\cdot\Delta t)$ that change discretely in time with a discretization step of Δt. The function $F : R^K \times R^N \to R^N$ is again based the MLP that has N units in the input layer, h units in the middle layer, and N units in the output layer. Due to the universal approximation capability of the MLP, this neural network has been widely used to model complex dynamics such as speech (Tank & Hopfield, 1987; Anderson & van Essen, 1987; Mozer, 1989) and chaos (Lapedes & Farber, 1987). For the learning of the Mackey-Glass equation (14) and the Rössler equations (15), the number of the dynamical units, the number of the middle units, and the discretization step were set as $(N, h, \Delta t) = (5, 5, 4.1)$ and $(4, 10, 0.52)$, respectively. Other learning conditions were set to be the same as those of the DRNN.

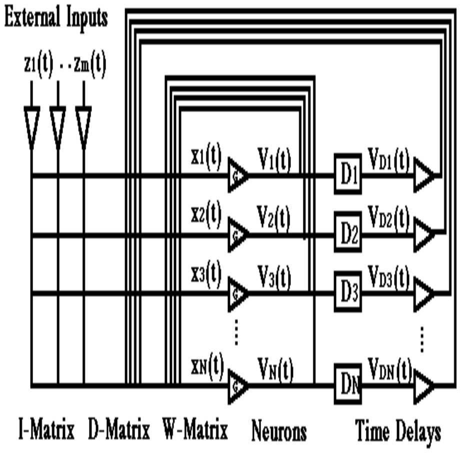

Figure 1: Schematic illustration of the Kleinfeld network of Eq. (1). Dynamical states of the N-neurons $\{x_1, \ldots, x_N\}$ are driven by the feedback connections from their outputs $\{V_1, \ldots, V_N\}$ and the delayed outputs $\{V_{\mathcal{D}_1}, \ldots, V_{\mathcal{D}_N}\}$. The network also receive m-external inputs $\{z_1, \ldots, z_m\}$.

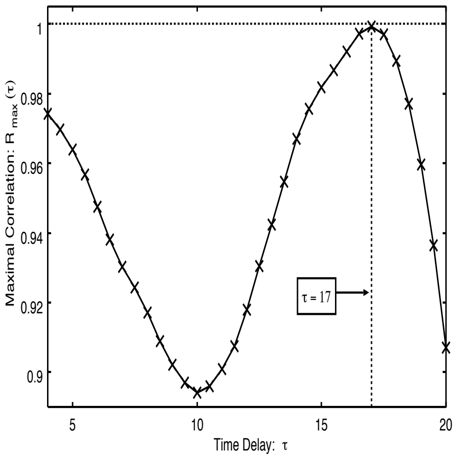

Figure 2: Maximal correlation function $R_{\max}(\tau)$ computed for the Mackey-Glass data with various time delay $\tau\in[4,20]$. The time delay of the original Mackey-Glass equation is identified at $\tau^* = 17$ with $R_{\max}(\tau^*)\approx 1$.

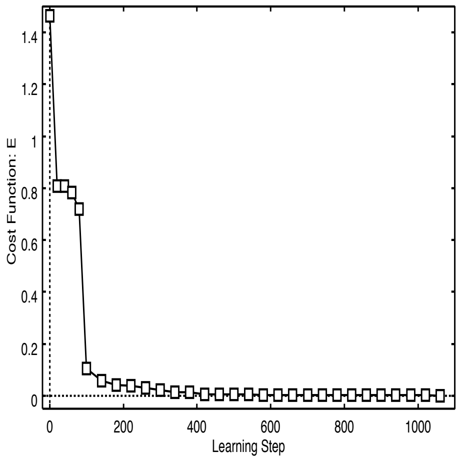

Figure 3: Cost function $E(\Omega)$ v.s. iterative steps of the learning algorithm.

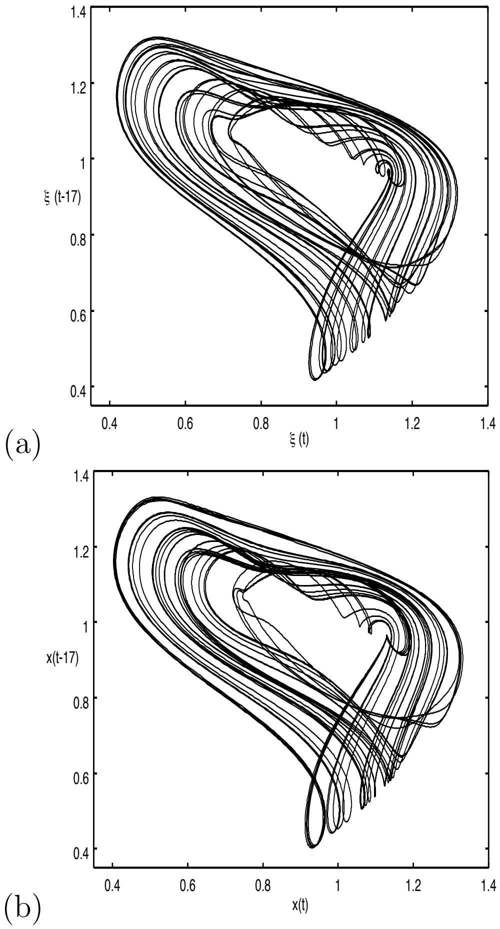

Figure 4: (a) Mackey-Glass dynamics of Eq. (14) in the $(\xi(t), \xi(t-17))$-space. (b) Dynamics of the DRNN (10) in the $(x(t), x(t-17))$-space after 1068-iterative learning. The network has one dynamical unit, one delayed feedback, and 5 middle units. 2-dimensional and 3-dimensional dynamics

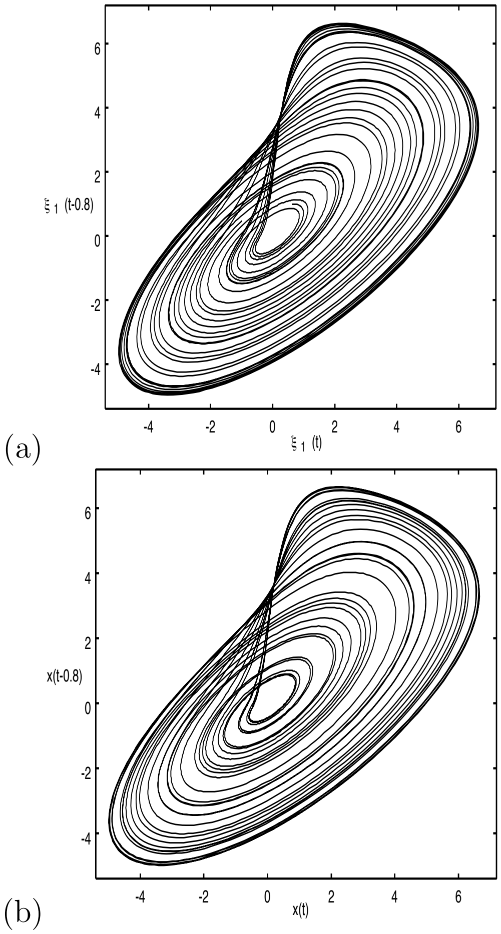

Figure 5: (a) Rössler dynamics of Eq. (15) in the $(\xi_1(t), \xi_1(t-0.8))$-space. (b) Dynamics of the DRNN (10) in the $(x(t), x(t-0.8))$-space after 2250-iterative learning. The network has one dynamical unit, two delayed feedbacks, and 8 middle units.

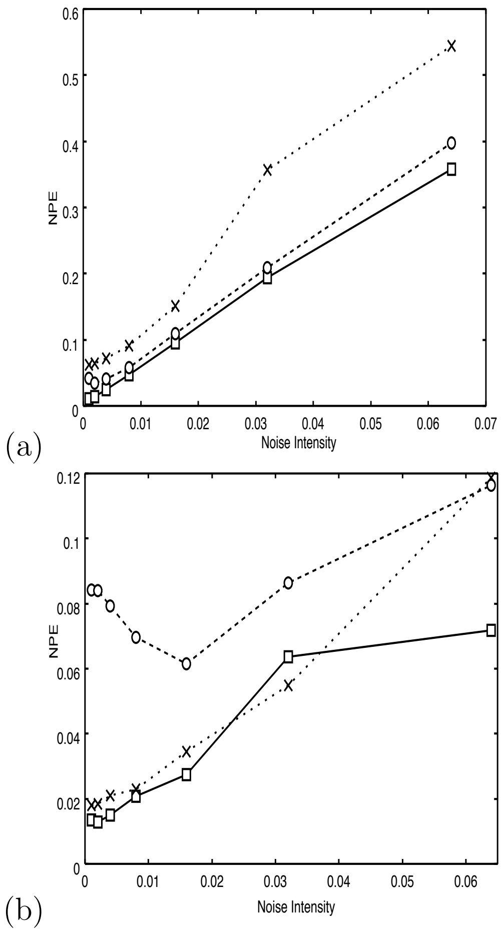

Figure 6: Nonlinear prediction errors of DRNN (solid line with squares), ORNN (dotted line with crosses), and TDNN (dashed line with circles) with an increasing noise level σ. As teacher signals, time series $\{\xi(t)\}$ generated from the Mackey-Glass equation (14) and the Rössler equations (15) are used for (a) and (b), respectively. The prediction intervals are fixed as $K = 4.1$ and 5.2 for (a) and (b), respectively.

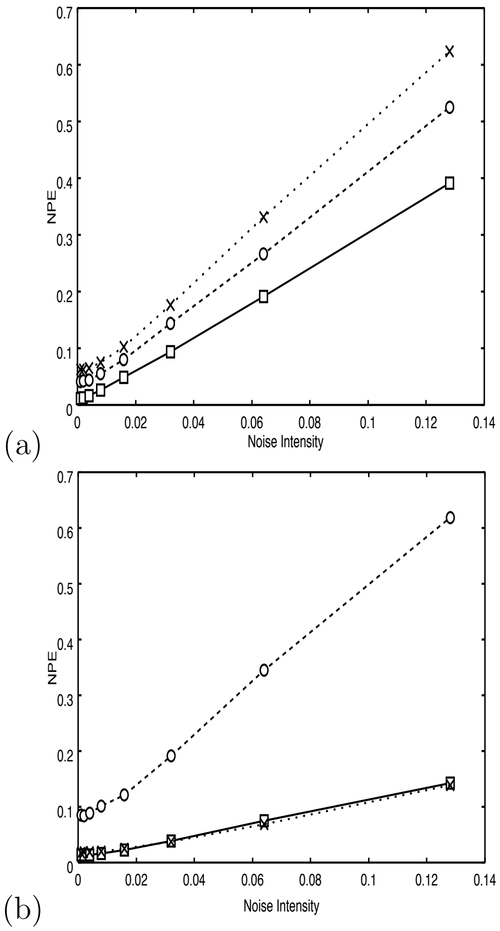

Figure 7: Nonlinear prediction errors of DRNN (solid line with squares), ORNN (dotted line with crosses), and TDNN (dashed line with circles) with an increasing noise σ on the initial condition. As teacher signals, time series $\{\xi(t)\}$ generated from the Mackey-Glass equation (14) and the Rössler equations (15) are used for (a) and (b), respectively.

Table 1: Comparison of the memory size and the computational cost required for learning algorithms of DRNN and ORNN (N: the number of neurons, K: the number of weight parameters, J_1: the division number of time interval τ, J_2: the division number of time interval T).

		DRNN	*ORNN*
RTRL	Memory Size	$N \cdot K \cdot J_1$	$N \cdot K$
Algorithm	Computation	$N \cdot K$ Sets of RFDEs	$N \cdot K$ Sets of ODEs
TDRL	Memory Size	$N \cdot J_2$	$N \cdot J_2$
Algorithm	Computation	N Sets of RFDEs	N Sets of ODEs